University of Granada

Department of Computer Science
and Artificial Intelligence

PhD Program in Information and
Communication Technologies

PhD THESIS DISSERTATION

TOWARDS GREEN AI: ALGORITHMIC STRATEGIES FOR COMPLEXITY REDUCTION IN DEEP LEARNING ARCHITECTURES

PhD STUDENT
LUIS BALDERAS RUIZ

PhD ADVISORS
José Manuel Benítez Sánchez
Miguel Lastra Leidinger

Granada, October 2025

Editor: Universidad de Granada. Tesis Doctorales
Autor: Luis Balderas Ruiz
ISBN: 978-84-1195-980-3
URI: https://hdl.handle.net/10481/110753

El doctorando / *The doctoral candidate* **Luis Balderas Ruiz** y los directores de la tesis / *and the thesis supervisors*: **José Manuel Benítez Sánchez** and **Miguel Lastra Leidinger**

Garantizamos, al firmar esta tesis doctoral, que el trabajo ha sido realizado por el doctorando bajo la dirección de los directores de la tesis y hasta donde nuestro conocimiento alcanza, en la realización del trabajo, se han respetado los derechos de otros autores a ser citados, cuando se han utilizado sus resultados o publicaciones.

Guarantee, by signing this doctoral thesis, that the work has been done by the doctoral candidate under the direction of the thesis supervisors and, as far as our knowledge reaches, in the performance of the work, the rights of the other authors to be cited (when their results or publications have been used) have been respected.

Lugar y fecha / *Place and date*: Granada, 3 de julio de 2025

The PhD student:	The PhD advisor:	The PhD advisor:
Sgd.: Luis Balderas Ruiz	Sgd.: José Manuel Benítez Sánchez	Sgd.: Miguel Lastra Leidinger

This doctoral thesis has been supported by the Industry Andalusian Council Research Project P18-TP-5168 and the Spanish National Research Projects TIN2016-81113-R, by Proyecto PID2020-118224RB-100 and Proyecto PID2023-151336OB-I00 financiado por MICIU/AEI /10.13039/501100011033 y por FEDER, UE.

To live a creative life,
we must lose our fear of being wrong.

JOSEPH CHILTON PEARCE

It is impossible for anyone to begin to learn
that which he thinks he already knows.
EPICTETUS, DISCOURSES, BOOK II

Agradecimientos

Esta tesis es el fruto de un esfuerzo compartido, y quiero expresar mi más sincero agradecimiento a quienes me han acompañado en este camino.

En primer lugar, a mis directores, José Manuel y Miguel. Vuestra inmensa dedicación e interés han sido fundamentales para que este proyecto se hiciera realidad. Gracias por la libertad que me habéis brindado y por enseñarme el oficio de la investigación, una vocación que siento desde niño y que he podido desarrollar plenamente en cada reunión y en cada paso que hemos dado juntos. Confío en que este sea solo el inicio de una colaboración duradera.

A mis compañeros de trabajo, gracias por compartir conmigo incontables horas dedicadas a un sentido del que todos hacemos patria. Gracias Elena, por tu apoyo, ayuda y comprensión infinita. A Jose, por ser un referente, un mentor y un amigo que siempre me impulsó cuando el desánimo se acentuaba; tenías razón, al final la lógica siempre se impone. Gracias infinitas a Manu, Jesús, José Antonio, José Fernando, Pablo, Alberto, Mario, Miriam, Nacho, Fernando y tantos otros. Sois la familia que se construye día a día para quienes estamos lejos de casa. En especial, gracias a vosotros, Manu y Jesús. Trabajar a vuestro lado es una fuente inagotable de risas y aprendizaje a partes iguales.

A mis queridos amigos, Rubén, Gema, Luis, Alberto y Pablo, gracias por enseñarme que el todo es mucho más que la suma de las partes. Y a Natalio, mi profesor, mi padre adoptivo, nunca habría sido matemático si nuestros caminos no se hubieran cruzado.

A mi familia, y en especial a mis padres, ejemplo de todo lo bueno que existe. Sois mi fuente de inspiración, de trabajo, de sacrificio y de cariño. Gracias por ser el refugio de valores en este mundo que a veces parece tambalearse. No habría conseguido nada de esto sin vosotros; un trocito de esta tesis doctoral será siempre vuestro.

Finalmente, el más profundo y sincero de los agradecimientos a Maripaz, mi pareja, mi compañera, mi amor incondicional. Has compartido conmigo cada etapa de este viaje, tanto los momentos de dificultad como los de alegría. Sé que en demasiadas ocasiones tuve que renunciar a nuestro tiempo juntos para avanzar en este sueño. Nunca podré devolverte toda la fuerza, energía, esperanza y apoyo que me has regalado para llegar hasta el final.

MUCHAS GRACIAS

Table of Contents

Chapter I

PhD dissertation

1 Resumen

1.1 Introducción

Las redes neuronales artificiales, epicentro de la inteligencia artificial contemporánea, son un paradigma computacional cuya arquitectura y funcionalidad se inspiran en la neurobiología del cerebro. Estas estructuras algorítmicas, compuestas por capas interconectadas de nodos —o "neuronas" artificiales—, son capaces de aprender representaciones jerárquicas de datos a través de un proceso de optimización iterativa. Mediante la propagación hacia adelante (inferencia) y la retropropagación de errores (entrenamiento) [1, 2], las redes neuronales ajustan sus pesos sinápticos para minimizar una función de pérdida, permitiéndoles modelar relaciones no lineales complejas en conjuntos de datos masivos. Su capacidad inherente para la extracción automática de características [3, 4] y su naturaleza de aproximadores universales [5] las ha posicionado como pilar fundamental en todos los campos del aprendizaje automático. De acuerdo a su topología y a las operaciones que realizan a los datos de entrada, las redes neuronales se pueden clasificar en varios tipos. A continuación, se presentan algunos de ellos.

Las redes neuronales densas feed-forward (FFNNs), a menudo denominadas perceptrones multicapa, constituyen la configuración más elemental de las redes neuronales profundas. Su arquitectura se organiza en distintas capas de neuronas artificiales: una capa de entrada que recibe los datos brutos, una o más capas ocultas donde se realizan los cálculos intermedios, y una capa de salida que produce el resultado final. La conectividad de estas redes es estrictamente unidireccional, fluyendo la información desde la entrada hacia la salida. Cada neurona de una capa se conecta a todas las neuronas de la capa siguiente.

A pesar de la gran importancia y presencia de las FFNNs en multitud de soluciones para problemas de la academia y la industria, se pudo comprobar que no son eficientes para procesar datos con una estructura espacial local inherente, como las imágenes. Dado que tratan cada píxel de una imagen como una entrada independiente, se ignoran las relaciones espaciales, siendo cruciales para encontrar patrones con píxeles cercanos. Además, esta estrategia conduce a un número muy elevado de parámetros, haciendo especialmente ineficiente su entrenamiento para imágenes de tamaño considerable. Esta limitación motivó el desarrollo de arquitecturas más especializadas. Concretamente, surgieron las redes neuronales convolucionales (CNNs) [6, 7, 8], particularmente bien adaptadas al procesamiento de datos visuales, dado que están formadas por componentes especializados en capturar de manera efectiva características espaciales. Las CNNs se han utilizado extensivamente y con éxito en problemas de visión por computador, como clasificación de imágenes [9, 10], detección de objetos en imagen y vídeo [11, 12, 13, 14], segmentación [15, 16] y en multitud de aplicaciones médicas [17, 18].

Para el procesamiento de datos secuenciales como el lenguaje o las series temporales, las redes neuronales recurrentes (RNN) [19, 20] y, en particular, las *Long Short-Term Memory* (LSTMs) [21] representaron un avance significativo, habida cuenta su capacidad para mantener información contextual a lo largo del tiempo, superando problemas de desvanecimiento de gradientes en secuencias largas [22]. Sin embargo, estas redes presentan limitaciones inherentes, como la dificultad de capturar dependencias a muy largo plazo de manera eficiente y, además, su naturaleza secuencial impedía una paralelización efectiva del entrenamiento, presentando requisitos muy altos en cuanto a tiempo de entrenamiento.

La arquitectura Transformers, introducida en 2017, revolucionó el campo del procesamiento del lenguaje natural (NLP) al abandonar la recurrencia y basarse en el mecanismo de atención [23]. Dicho mecanismo permite al modelo ponderar la importancia de diferentes partes de la secuencia de entrada al procesar cada elemento, capturando dependencias a largo plazo de manera eficiente y permitiendo el procesamiento de la secuencia en paralelo. En consecuencia, se acelera notablemente el proceso de entrenamiento e inferencia. La arquitectura, formada por un *encoder* (codificador) y un *decoder* (decodificador), da lugar a distintas funcionalidades. En el caso del *encoder*, han surgido modelos como BERT [24] y sus derivados (RoBERTa [25], DistilBERT [26], ALBERT [27] o ModernBERT [28] entre otros) destinados a tareas relacionadas con clasificación de textos, clustering o análisis semántico [29, 30]. Por el contrario, la parte del *decoder* se centra en la generación. Destaca la serie GPT (Generative Pre-trained Transformer) por su calidad en la generación de texto (GPT-3 [31]) y, en las siguientes versiones, por su carácter multimodal (GPT-4 [32]). Otras arquitecturas de gran capacidad generadora son Llama [33], Mistral [34] o DeepSeek [35]. Todas ellas, con sus diferencias, han mostrado notables capacidades de *few-shot learning* (capacidad de aprender de unos pocos ejemplos etiquetados), mejorando significativamente su versatilidad y utilidad tanto en la academia como en la industria.

El fenómeno de la aparición de capacidades inesperadas a través de la escala es un aspecto notable de los Transformers. El tamaño masivo de modelos ha permitido llevar a cabo una amplia gama de tareas de lenguaje natural sin un entrenamiento expreso, dando lugar a investigaciones relacionadas con *In-Context Learning of Representations* [36]. Los resultados sugieren que escalar el tamaño del contexto puede desbloquear nuevas capacidades. Esto implica una relación donde el aumento exponencial en el número de parámetros y la cantidad de datos de entrenamiento no solo conduce a mejoras lineales en el rendimiento, sino que puede dar lugar a la aparición de habilidades cualitativamente nuevas (como la capacidad de aprender de unos pocos ejemplos o de generar texto altamente coherente y contextualmente relevante) que no son predecibles a partir de modelos más pequeños. En consecuencia, cada nueva generación de modelos lleva aparejada un crecimiento considerable en el número de parámetros.

El número de parámetros de un modelo determina cuántas operaciones se realizan. Además, cada parámetro es una unidad de información generada a partir del entrenamiento, por lo que hay una relación directamente proporcional entre el número de parámetros y la cantidad de memoria requerida por el modelo. Tanto el número de operaciones como la memoria se traducen de forma directa en consumo energético. En consecuencia, en la medida en que los modelos no dejan de crecer en el número de parámetros a ritmo vertiginoso, el impacto medioambiental de su entrenamiento y puesta en producción es enorme en términos de consumo energético y huella de carbono [37, 38].

Para mitigar los efectos dañinos de este impacto medioambiental, surge el paradigma Green AI [39], con vocación de proponer modelos de inteligencia artificial más sostenibles por medio del uso de métricas que no sólo tengan en cuenta el acierto en predicción pero también el número de parámetros, el consumo de electricidad o el número de operaciones que lleva a cabo el modelo. Para ello, los algoritmos de simplificación o poda se erigen como una herramienta fundamental para reducir el tamaño de modelos preentrenados.

Amparados en el paradigma Green AI, en esta tesis nos centramos en la construcción de algoritmos de simplificación de redes neuronales. Como se ha podido comprobar a lo largo de esta sección, existe gran variedad de redes neuronales en función de su naturaleza, por lo que se requieren de estrategias específicas para la poda de cada una de ellas. Concretamente, en esta tesis se proponen un algoritmo de simplificación de redes neuronales densas feed-forward (ODF2NNA), un algoritmo de simplificación de redes neuronales convolucionales (OCNNA), un algoritmo de simplificación de Transformer-encoder basado en BERT (PBCE) y un algoritmo de simplificación de Transformer-decoder, concretamente para modelos de lenguajes con la estrategia Mixture-of-Experts o MoE-LLMs (MoEITS). Todas las técnicas tienen como objetivo determinar qué neuronas o conjuntos de neuronas son las más importantes, significativas o han alcanzado mayor relevancia en el entrenamiento y, por ende, son las fundamentales para que la predicción que genera la red sea precisa y de calidad. Una vez encontradas dichas neuronas, se pueden simplificar aquellas que son menos relevantes o útiles en la predicción, minimizando así la degradación que sufre la red al perder parte de sus componentes. Para ello, se utilizan diferentes técnicas de análisis de datos en cada caso: aproximaciones basadas en estadística, selección de características, teoría de homología o teoría de la información.

1.2 Objetivos

Esta tesis doctoral propone desarrollar algoritmos Green AI para la simplificación de modelos de deep learning, con el fin de reducir su impacto ambiental y facilitar su implementación en entornos con recursos de hardware limitados. Para ello, se realizará un análisis cualitativo y cuantitativo sobre la importancia de las neuronas o unidades individuales, buscando inspirar nuevas arquitecturas. Los objetivos específicos son los siguientes:

- **Objetivo (O1): Diseñar e implementar un algoritmo de simplificación para redes neuronales densas feed-forward.** Se propondrá un método para simplificar FFNNs, tomando como referencia redes y problemas de aprendizaje de distinto tamaño y complejidad.

- **Objetivo (O2): Diseñar e implementar un algoritmo de simplificación para redes neuronales convolucionales.** Este objetivo se centra en desarrollar una metodología capaz de identificar los filtros convolucionales cruciales para simplificar redes preentrenadas, como las utilizadas en clasificación de imágenes, y comparar su rendimiento con el estado del arte.

- **Objetivo (O3): Diseñar e implementar un algoritmo de simplificación para la arquitectura Transformer.** Dada la relevancia actual de los Transformers, se desarrollarán algoritmos de simplificación específicos para sus componentes codificador y decodificador, incluyendo modelos de gran escala como los basados en Mixture-of-Experts (MoE-LLMs).

- **Objetivo (O4): Desplegar redes simplificadas para resolver problemas complejos de aprendizaje automático.** Se aplicarán las versiones simplificadas (y a menudo más eficaces) de las redes neuronales generadas a partir de los algoritmos de simplificación desarrollados en los objetivos anteriores para abordar problemas complejos en campos como la biomedicina, la economía, la energía y las ciudades inteligentes.

1.3 Resultados

Los resultados de esta tesis doctoral se plasman en un conjunto de artículos científicos que dan respuesta a los objetivos presentados anteriormente. Son los siguientes:

- Balderas, L., Lastra, M., & Benítez, J. M. (2023). Optimizing dense feed-forward neural networks. *Neural Networks*, 171, 229-241. https://doi.org/10.1016/j.neunet.2023.12.015 (O1).
- Balderas, L., Lastra, M., & Benítez, J. M. (2024). Optimizing Convolutional Neural Network Architectures. *Mathematics*, 12(19), 3032. https://doi.org/10.3390/math12193032 (O2).
- Balderas, L., Lastra, M., & Benítez, J. M. (2025). A Green AI Methodology Based on Persistent Homology for Compressing BERT. *Applied Sciences*, 15(1), 390. https://doi.org/10.3390/app15010390 (O3)
- Balderas, L., Lastra, M., & Benítez, J. M. (2025). MoEITS: A Green AI approach for simplifying MoE-LLMs. Submitted (O3).
- Balderas, L., Lastra, M., & Benítez, J. M. (2024). An Efficient Green AI Approach to Time Series Forecasting Based on Deep Learning. *Big Data and Cognitive Computing*, 8(9), 120. https://doi.org/10.3390/bdcc8090120 (O4).
- Balderas, L., Pérez, F.J., Moreno de Castro, M., Lastra, M., Martínez, J.P., Lainez-Ramos, A.J., Arauzo, A., & Benítez, J.M. (2025). On MRI-Based Central Nervous System Tumor Differentiation using Deep Neural Networks and Conformal Predictors (O4).

A continuación, se describe en detalle cada uno de los algoritmos y se presentan los principales resultados obtenidos. Se diseñaron e implementaron cuatro algoritmos de simplificación, cada uno adaptado a una arquitectura específica de red neuronal. En la Tabla I.1 se recoge un resumen detallado de cada uno de los algoritmos, descripción, ámbito de aplicación y objetivo con el que se relaciona.

- **ODF2NNA (Optimizing Dense Feed-Forward Neural Network Algorithm, Objetivo O1):** Este algoritmo se enfoca en la simplificación de redes neuronales densas feed-forward para tareas de clasificación y regresión. La selección de neuronas se basa en la variabilidad de sus salidas, cuantificada mediante la desviación estándar. Las neuronas con baja variabilidad se podan, y su contribución se transfiere al *bias* para mitigar la pérdida de rendimiento. ODF2NNA demostró una reducción significativa de parámetros (hasta un 98% en arquitecturas como LeNet300-100) y una mejora en la precisión predictiva (entre un 15% y un 40% en ciertos casos) en conjuntos de datos de pequeña, mediana y gran escala, superando a técnicas de poda existentes. Un hallazgo relevante es que los modelos simplificados por ODF2NNA superan en precisión a redes concebidas inicialmente con arquitecturas de tamaño reducido, sugiriendo una transferencia de conocimiento efectiva entre el modelo original y el modelo simplificado.
- **OCNNA (Optimizing Convolutional Neural Network Architectures, Objetivo O2):** Diseñado para la simplificación de redes neuronales convolucionales (CNNs), OCNNA emplea una combinación de Análisis de Componentes Principales (PCA), la norma de Frobenius y el Coeficiente de Variación (CV) para cuantificar la importancia de los filtros convolucionales. Se trata de un algoritmo con un parámetro k que modula la intensidad de la poda. Su efectividad

se validó en arquitecturas preentrenadas como ResNet-50, VGG-16, DenseNet-40 y MobileNet, y en conjuntos de datos de referencia (CIFAR-10, CIFAR-100, ImageNet). OCNNA logró una reducción sustancial de parámetros (hasta un 86.68% en VGG-16 para CIFAR-10, con una mejora de precisión del 0.42%), manteniendo o incrementando la precisión en todos los casos. Se observó que un valor de $k = 40$ (percentil de filtros a retener) ofrecía el mejor equilibrio entre reducción de tamaño y precisión, demostrando la versatilidad del algoritmo para diversas arquitecturas y su competitividad frente a más de 20 técnicas del estado del arte.

- **PBCE (Persistent BERT Compression and Explainability, Objetivo O3):** Este algoritmo se desarrolló para la simplificación de *encoders* Transformer de la familia BERT, utilizando homología persistente de dimensión cero para evaluar, por medio del sus características topológicas de las salidas de las neuronas, la relevancia de las mismas. PBCE logró una compresión notable (hasta un 47% de los parámetros originales en BERT Base y un 43% en BERT Large) mientras mejoraba el rendimiento en la mayoría de las tareas del GLUE Benchmark. Este método no solo optimiza la eficiencia del modelo, sino que también contribuye a la explicabilidad al identificar las unidades neuronales más significativas.

- **MoEITS (Mixture of Expert Information Theory Simplifier, Objetivo O3):** Enfocado en la simplificación de decodificadores Transformer, específicamente MoE-LLMs, MoEITS se basa en la teoría de la información, utilizando la Información Mutua Normalizada (NMI) para cuantificar la redundancia entre expertos. El algoritmo selecciona iterativamente a los expertos más informativos, logrando una reducción de hasta el 43% en el conteo de parámetros en el modelo Mixtral 8×7B, superando a las técnicas del estado del arte en tareas de razonamiento y comprensión. Se destaca la distinción entre la simplificación física (eliminación de componentes) que realiza MoEITS y la simplificación lógica (inducción de escasez) de otras técnicas, siendo la primera crucial para entornos con restricciones de hardware. El estudio del parámetro τ reveló que es útil para conseguir un control preciso en el nivel de compresión, identificando un equilibrio precisión-simplificación muy efectivo en $\tau = 0.75$ para el benchmark BoolQ.

- **Aplicaciones de las Redes Simplificadas en Problemas Complejos, Objetivo O4:** Los algoritmos de simplificación propuestos se aplicaron con éxito en la resolución de problemas complejos del mundo real. A continuación, se introducen dos aplicaciones:

  – En primer lugar, se presenta la metodología GreeNNTSF, que integra ODF2NNA como modelo de simplificación para construir modelos de predicción de series temporales. Se evaluó rigurosamente en diversos conjuntos de datos de series temporales (SARS-CoV-2, Brazilian Weather, PhysioNet, WTI, tráfico y consumo eléctrico) obteniendo resultados sobresalientes. En concreto, GreeNNTSF igualó e incluso superó el rendimiento de las técnicas de vanguardia, generando modelos significativamente más compactos y precisos. Por ejemplo, en la predicción del SARS-CoV-2, logró una reducción del 19% en MAE y del 81% en RMSE. En el pronóstico de consumo eléctrico, superó a un modelo de *ensemble* complejo, reduciendo sus parámetros en un 35% y mejorando métricas clave como MAPE (5%), RMSE (20%) y MdAPE (54%).

  – En segundo lugar, se desarrolló un sistema de clasificación en cascada para diferenciar gliomas de bajo y alto grado (LGG/HGG) y sus subtipos según la Organización Mundial de la Salud (OMS), utilizando curvas de perfusión derivadas de imágenes de resonancia magnética (MRI). En la etapa inicial HGG/LGG, una MCDCNN simplificada con OCNNA demostró un rendimiento óptimo. Para la clasificación de LGG (Grado OMS I/II), el modelo 1NN-DTW fue el más efectivo. Finalmente, para la clasificación de HGG

(Grado OMS III/IV), se seleccionó una red neuronal densa simplificada con ODF2NNA debido a su rendimiento superior en F1-score, Brier Loss y Log Loss. La incorporación de *conformal predictors* [40] mejoró significativamente la confianza y la precisión del sistema, rectificando errores de clasificación y proporcionando información crucial para la toma de decisiones clínicas usando modelos de inteligencia artificial confiable.

Algoritmo	**Descripción**	**Casos de uso de aplicación**	**Objetivo**
ODF2NNA	Algoritmo de simplificación de redes neuronales densas. Cuantifica la relevancia de las neuronas por medio de la desviación estándar de sus salidas.	Problemas de clasificación y regresión tabulares. Metodología aplicada para la construcción de modelos de predicción de series temporales (GreeNNTSF) y diferenciación de gliomas HGG para los Grados OMS III/IV.	O1, O4
OCNNA	Algoritmo de simplificación de redes neuronales convolucionales. Combina técnicas de análisis de datos como PCA, norma de Frobenius y CV para determinar los canales convolucionales relevantes.	Clasificación de imágenes con modelos convolucionales. ImageNet, CIFAR-10 y CIFAR-100 para las arquitecturas ResNet-50, VGG-16, DenseNet-40 y MobileNet. Metodología aplicada para la simplificación de modelos convolucionales en la diferenciación de gliomas HGG y LGG.	O2, O4
PBCE	Algoritmo de simplificación de codificadores Transformers. Uso de homología persistente de dimensión cero para evaluar la importancia de las neuronas por medio de las características topológicas presentes en sus salidas.	Evaluado en la familia BERT (Base y Large) empleando el GLUE Benchmark, incluyendo diferentes tareas de NLP.	O3
MoEITS	Algoritmo de simplificación de decodificadores Transformers, concretamente MoE-LLMs. Basado en teoría de información, emplea la Información Mutua Normalizada para cuantificar la redundancia de los expertos.	Aplicado al modelo Mixtral 8×7B, se evalúa en distintos benchmarks de razonamiento y comprensión como son BoolQ, HelloSWag, WinoGrande y ARC.	O3

Table I.1: Algoritmos propuestos en la tesis doctoral, su descripción, casos de uso evaluados y objetivos a los que se asocian.

Estos resultados demuestran la efectividad y versatilidad de los algoritmos de simplificación propuestos, validando su potencial para generar modelos de deep learning más eficientes, precisos y sostenibles para una amplia gama de aplicaciones prácticas y científicas.

1.4 Conclusiones

Las redes neuronales profundas, en sus múltiples formas y topologías, han sido durante mucho tiempo fundamentales en el desarrollo de soluciones innovadoras de aprendizaje automático, experimentando una reciente explosión de popularidad en la industria, la academia y el dominio público. A medida que el tamaño, la complejidad y las demandas computacionales de estas redes continúan escalando en la búsqueda de mayores capacidades de resolución de problemas, el paradigma Green AI y los algoritmos de simplificación de redes se han vuelto indispensables. Estos enfoques son cruciales para grupos de investigación, pequeñas empresas y usuarios individuales, ya que les permite aprovechar la inteligencia computacional avanzada sin requerir los colosales recursos informáticos disponibles solo para las grandes corporaciones tecnológicas. Esta tesis doctoral aborda de manera integral este desafío, explorando arquitecturas clave de redes neuronales y proponiendo un algoritmo de simplificación específico para cada una.

El primer objetivo de esta tesis introdujo ODF2NNA, un algoritmo diseñado específicamente para simplificar redes neuronales densas feed-forward. Su sencilla definición y facilidad de uso contrastan con su notable versatilidad, ya que la experimentación ha demostrado su utilidad en

problemas que involucran conjuntos de datos y modelos pequeños, medianos y grandes. Esta amplia aplicabilidad subraya el potencial de ODF2NNA como una herramienta fundamental para optimizar diversas aplicaciones de aprendizaje profundo.

El segundo objetivo se centró en la simplificación de redes neuronales convolucionales. Para este propósito, se propuso OCNNA, un método novedoso que aprovecha ingeniosamente herramientas clásicas de estadística y análisis de datos, incluyendo el Análisis de Componentes Principales (PCA), la norma matricial de Frobenius y el Coeficiente de Variación. Al utilizar estas robustas técnicas para cuantificar la importancia de los filtros convolucionales, OCNNA genera versiones simplificadas de modelos preentrenados ampliamente adoptados, validados en conjuntos de datos de referencia para clasificación de imágenes.

Posteriormente, el tercer objetivo profundizó en el estudio y la simplificación de la arquitectura Transformer, una innovación fundamental en la Inteligencia Artificial del siglo XXI. Esta arquitectura, que comprende distintos componentes llamados *encoder* y *decoder*, presentó desafíos de simplificación únicos. Para el *encoder*, se introdujo PBCE, una metodología que aplica la teoría de la homología persistente de dimensión cero para evaluar la relevancia de las neuronas y componentes dentro de los modelos de la familia BERT. Una evaluación exhaustiva en el ampliamente reconocido GLUE Benchmark demostró consistentemente la capacidad de PBCE para producir modelos simplificados sin comprometer la capacidad predictiva. Abordando el componente del *decoder*, se desarrolló un novedoso método de simplificación para Large Language Models basados en Mixture-of-Experts, fundamentado en la teoría de la información. Específicamente, se empleó la información mutua normalizada (NMI) para cuantificar la importancia de los expertos individuales. Este enfoque produjo modelos notablemente reducidos, que fueron rigurosamente evaluados en los benchmarks estándar para tareas de generación de texto y razonamiento.

Finalmente, el cuarto objetivo de esta tesis se centró en aplicar los métodos de simplificación propuestos para resolver problemas complejos del mundo real utilizando versiones reducidas de redes neuronales profundas. En primer lugar, dentro del dominio de la predicción de series temporales, se presentó la metodología GreeNNTSF. Este marco, construido sobre el algoritmo de poda ODF2NNA, construye modelos que superan consistentemente a las técnicas de vanguardia existentes en diversos problemas que abarcan la economía, las ciudades inteligentes, la biomedicina y la climatología. En segundo lugar, se abordó un problema biomédico crítico: la diferenciación de glioblastomas utilizando curvas de perfusión como datos de entrada. En este caso, los algoritmos ODF2NNA y OCNNA se aplicaron de forma sinérgica para simplificar redes complejas para la clasificación de series temporales. Esta integración dio como resultado el desarrollo de un sistema eficiente y eficaz de apoyo a la decisión para radiólogos, mejorado además por la incorporación de elementos de explicabilidad basados en *conformal predictors*. Esta aplicación final no solo demuestra la utilidad práctica de los algoritmos de simplificación propuestos, sino que también resalta su potencial para contribuir significativamente en contextos clínicos reales.

2 Introduction

Artificial Neural Networks (ANNs), a cornerstone of contemporary artificial intelligence, represent a computational paradigm whose architecture and functionality draw inspiration from the neurobiology of the brain. These algorithmic structures, comprising interconnected layers of nodes —or artificial "neurons"— are capable of learning hierarchical representations of data through an iterative optimization process. By forward propagation (inference) and error backpropagation (training) [1, 2], neural networks adjust their synaptic weights to minimize a loss function, enabling them to model complex nonlinear relationships in massive datasets. Their inherent capacity for automatic feature extraction [3, 4] and their nature as universal approximators [5] have positioned them as a fundamental pillar across all fields of machine learning. According to their topology and the operations performed on input data, neural networks can be classified into various types. Several of these will be discussed subsequently.

Feed-forward neural networks (FFNNs), often referred to as multilayer perceptrons, constitute the most elemental configuration of deep neural networks. Their architecture is organized into distinct layers of artificial neurons: an input layer that receives raw data, one or more hidden layers where intermediate computations are performed, and an output layer that produces the final result. The connectivity within these networks is strictly unidirectional, with information flowing from the input towards the output. Each neuron in one layer connects to all neurons in the subsequent layer.

Despite being the earliest neural network architecture, dense feed-forward neural networks remain remarkably prevalent and valuable across numerous academic and industrial applications. Evidence of their enduring significance is found in over 58,000 articles and 51,000 patents related to FFNNs within Scopus, the widely utilized multidisciplinary bibliographic and citation database launched by Elsevier. This underscores their continued importance in contemporary research and development.

However, FFNNs have been shown to be inefficient for processing data with inherent local spatial structures, such as images. Their approach, which treats each pixel of an image as an independent input, overlooks crucial spatial relationships necessary for identifying patterns among proximate pixels. Furthermore, this strategy leads to an exceptionally high number of parameters, rendering their training particularly inefficient for considerably sized images. This limitation motivated the development of more specialized architectures, notably Convolutional Neural Networks (CNNs) [6, 7, 8]. CNNs are particularly well-suited for visual data processing due to their specialized components designed for effectively capturing spatial features.

CNNs have been extensively and successfully employed in a wide array of computer vision problems. These include, but are not limited to, image classification [9, 10], object detection in images and video [11, 12, 13, 14], segmentation [15, 16], and numerous medical applications [17, 18].

For the processing of sequential data, such as language or time series, Recurrent Neural Networks (RNNs) [19, 20], and particularly Long Short-Term Memory (LSTMs) [21], represented a significant advancement. These architectures could maintain contextual information over time, thereby overcoming vanishing gradient problems in long sequences [22]. However, these networks exhibited inherent limitations, including difficulty in efficiently capturing very long-term dependencies. Furthermore, their sequential nature precluded effective parallelization of training, rendering them slow for large volumes of data.

The Transformer architecture, introduced in 2017, revolutionized the field of natural language processing (NLP) by abandoning recurrence and relying on the attention mechanism [23]. This mechanism allows the model to weigh the importance of different parts of the input sequence when processing each element, efficiently capturing long-term dependencies and enabling parallel processing of the sequence. Consequently, the training and inference processes are notably accelerated.

The Transformer architecture, comprising an encoder and a decoder, gives rise to distinct functionalities. On the encoder side, models such as BERT [24] and its derivatives (e.g., RoBERTa [25], DistilBERT [26], ALBERT [27], and ModernBERT [28]) have emerged, primarily aimed at tasks related to text classification, clustering, and semantic analysis [29, 30]. Conversely, the decoder component focuses on generation. The GPT (Generative Pre-trained Transformer) series stands out for its quality in text generation (GPT-3 [31]) and, in subsequent versions, for its multimodal capabilities (GPT-4 [32]). Other highly capable generative architectures include Llama [33], Mistral [34], and DeepSeek [35]. All these models, despite their differences, have demonstrated remarkable few-shot learning capabilities (approach that enables a model to learn a task with only a small number of labeled training examples), significantly enhancing their versatility and utility in both academia and industry.

The emergence of unexpected capabilities through scaling is a notable aspect of Transformer models. The substantial size of these models has facilitated the execution of a wide range of natural language tasks without explicit training, leading to research in areas such as In-Context Learning of Representations [36]. Results suggest that scaling context size can unlock novel capabilities. These new capabilities, such as the capacity for few-shot learning or the generation of highly coherent and contextually relevant text, are not predictable from smaller models. Consequently, each new generation of models entails a substantial growth in the number of parameters.

The number of parameters within a model dictates the volume of operations performed. Furthermore, each parameter represents a unit of information generated through training; consequently, a direct proportional relationship exists between the number of parameters and the memory capacity required by the model. Both the number of operations and memory consumption directly correlate with energy consumption. As models continue to expand at an accelerating rate in terms of parameter count, the environmental impact of their training and deployment becomes enormous in terms of energy consumption and carbon footprint [37, 38].

To mitigate the detrimental effects of this environmental impact, the Green AI paradigm [39] has emerged. This paradigm advocates for more sustainable artificial intelligence models by incorporating metrics that consider not only predictive accuracy but also the number of parameters, electricity consumption, and the number of operations performed by the model. To this end, model simplification or pruning algorithms are posited as fundamental tools for reducing the size of pre-trained models.

Operating within the Green AI paradigm, this thesis focuses on the development of neural network simplification algorithms. As evidenced throughout this section, there is a wide variety of neural networks based on their nature, thus requiring specific pruning strategies for each type. Concretely, this thesis proposes an algorithm for simplifying dense feed-forward neural networks (ODF2NNA), a simplification algorithm for convolutional neural networks (OCNNA), a simplification algorithm for Transformer-encoders based on BERT (PBCE), and a simplification algorithm for Transformer-decoders, particularly for Mixture-of-Experts (MoE) Large Language Models (MoEITS).

All these techniques aim to determine which neurons or sets of neurons are most important, significant, or have achieved greater relevance during training. These are considered fundamental for ensuring the accuracy and quality of the network's predictions. Once these crucial neurons are identified, less relevant or useful neurons can be simplified, thereby minimizing the degradation experienced by the network as it loses some of its components. To achieve this, various data analysis techniques are employed in each case, including statistics-based approaches, feature selection, homology theory, and information theory.

Following the introduction presented in this section, the structure of the current document is outlined. This dissertation is divided into two main parts: the doctoral thesis itself and its constituent publications.

In the first part, Section 2 provides a detailed description of the necessary theoretical background, encompassing both model-centric perspectives and data analysis tools relevant to the development of the algorithms. Section 3 describes the objectives established to tackle the identified problems. Section 4 details the methodology employed throughout the development of the doctoral thesis. Section 5 offers a summary of the research articles that comprise the thesis. Subsequently, Section 6 showcases the primary results achieved. Finally, Section 7 synthesizes the fundamental conclusions and proposes avenues for future research. The second part of the dissertation comprises six research papers. Of these, four have been published in indexed international journals, and two are currently under review. These works are as follows:

- Balderas, L., Lastra, M., & Benítez, J. M. (2023). Optimizing dense feed-forward neural networks. *Neural Networks*, 171, 229-241. https://doi.org/10.1016/j.neunet.2023.12.015.
- Balderas, L., Lastra, M., & Benítez, J. M. (2024). Optimizing Convolutional Neural Network Architectures. *Mathematics*, 12(19), 3032. https://doi.org/10.3390/math12193032.
- Balderas, L., Lastra, M., & Benítez, J. M. (2025). A Green AI Methodology Based on Persistent Homology for Compressing BERT. *Applied Sciences*, 15(1), 390. https://doi.org/10.3390/app15010390.
- Balderas, L., Lastra, M., & Benítez, J. M. (2025). MoEITS: A Green AI approach for simplifying MoE-LLMs. Submitted.
- Balderas, L., Lastra, M., & Benítez, J. M. (2024). An Efficient Green AI Approach to Time Series Forecasting Based on Deep Learning. *Big Data and Cognitive Computing*, 8(9), 120. https://doi.org/10.3390/bdcc8090120
- Balderas, L., Pérez, F.J., Moreno de Castro, M., Lastra, M., Martínez, J.P., Lainez-Ramos, A.J., Arauzo, A., & Benítez, J.M. (2025). On MRI-Based Central Nervous System Tumor Differentiation using Deep Neural Networks and Conformal Predictors. Submitted.

3 Preliminaries

3.1 Deep Neural Networks: A brief review

Neural networks have undergone significant evolution, leading to specialized architectures for diverse machine learning tasks. The following section introduces some of the network architectures addressed in this doctoral thesis, for which an algorithm for their simplification is proposed.

3.1.1 Dense Feed-Forward Neural Networks

Dense feed-forward networks represent the primordial form of artificial neural networks. Their structure is characterized by each neuron in one layer connecting to all neurons in the subsequent layer. Information flows unidirectionally from the input layer to the output layer. Mathematically, the operation of a dense layer is defined as:

$$h^{(l+1)} = f(W^{(l)}h^{(l)} + b^{(l)}) \tag{I.1}$$

where:

- $h^{(l)}$ is the activation vector of layer l
- $W^{(l)}$ is the weight matrix connecting layer l to layer $l+1$
- $b^{(l)}$ is the bias vector of layer l
- $f()$ is a non-linear activation function

As introduced previously, feed-forward neural networks have achieved considerable success in the scientific literature and are well-suited for learning tasks involving tabular data where relationships between features are complex but do not exhibit an explicit temporal structure. An example of an FFNN is presented in Figure 1.

3.1.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are specifically designed to process data with local structure, particularly for detecting patterns such as edges, textures, and ultimately objects in images. Key components of CNNs include:

- Convolutional Layers: These layers apply filters to small regions of the input to detect local and hierarchical features, such as edges, textures, or shapes. The convolutional operation can be mathematically represented as a weighted sum of the input region's pixels by the filter's values. These layers are crucial for detecting translation-invariant features, meaning the network can recognize a feature regardless of its position within the image.
- Pooling Layers: Pooling layers reduce the spatial dimensionality of representations, for instance, by selecting the maximum (max pooling) or average (average pooling) value from a region. This contributes to making the network more robust to minor variations in feature position and reduces computational complexity.

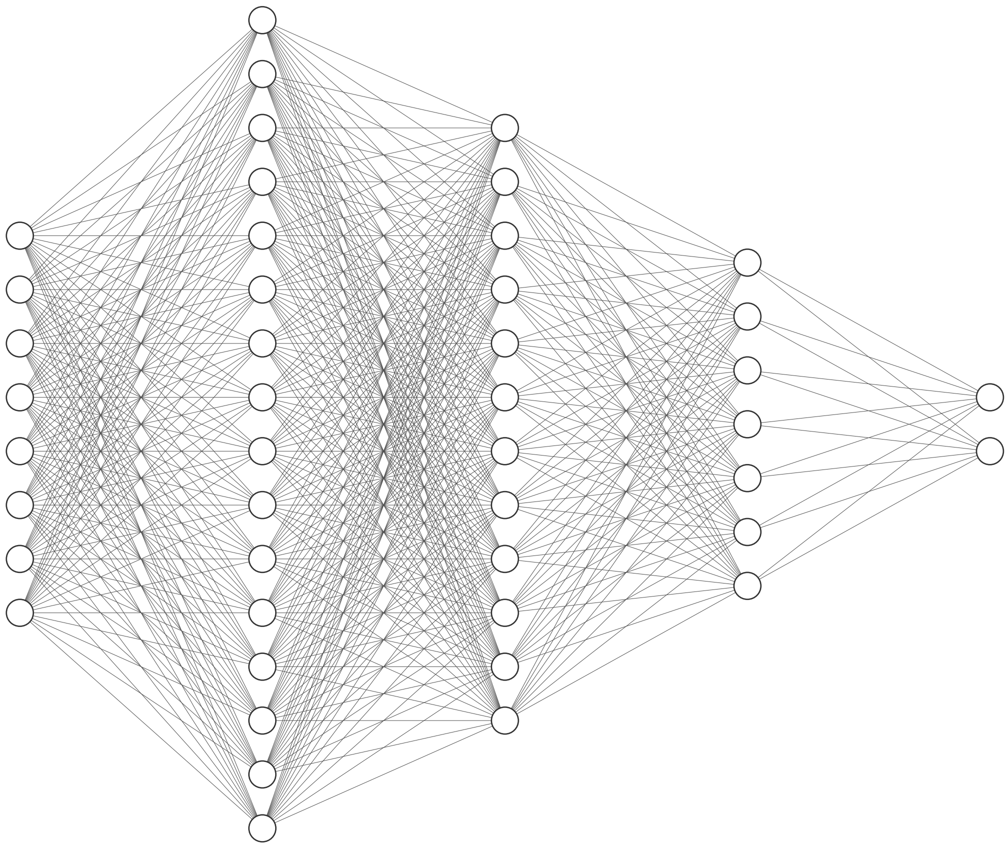


Figure 1: Example of feed-forward neural network

- Fully Connected Layers: At the end of the network, the high-level features extracted by the convolutional and pooling layers are flattened and connected to a set of dense layers to perform the network's learning task.

The primary operation, known as convolution, occurs via a filter that slides across the image, calculating the dot product between the filter and local portions of the image. This process is applied across the width and height of the image, generating feature maps [41]. The mathematical operation of a 2D discrete convolution between an input image I and a filter or kernel K can be expressed as:

$$S(i,j) = (I * K)(i,j) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} I(i-m, j-n)K(m,n) \tag{I.2}$$

where:

- I is the input image
- K is a convolutional filter of size $M \times N$
- b^l is the bias vector of layer l
- $S(i,j)$ is the feature map at position (i,j), represeting the coordinates of the output pixel

Other criteria involved in convolution include the stride, which defines how many positions the filter moves across the image, and padding, which adds extra pixels around the edges to complete the operation. An example of this type of network can be found in Figure 2.

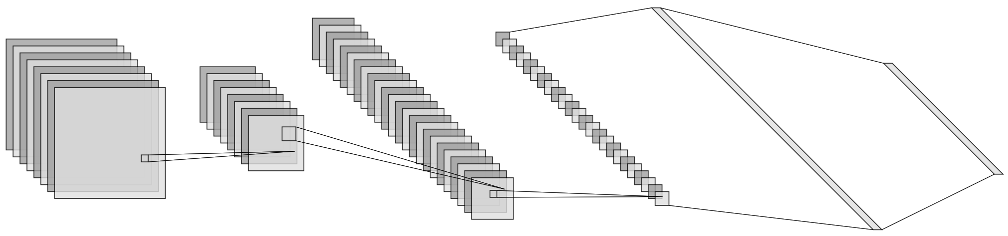


Figure 2: Example of convolutional neural network

3.1.3 Transformer architecture

The Transformer architecture (Figure 3) has become a fundamental component in all advancements related to Natural Language Processing (NLP) and multimodal generation. It operates as a neural network composed of two main stacks: an encoder and a decoder. The encoder processes an input sequence and produces a contextualized vector representation or embedding. The decoder, conversely, generates an output sequence from an input, often in an autoregressive manner. The key to its success lies in the Attention mechanism, which is capable of capturing dependencies within long sequences. Two primary types of Attention methods exist: Attention, which considers relationships between input and output elements, and Self-attention, which is applied exclusively to input data.

Self-attention (Figure 4), the mechanism prominently featured in the Transformer, draws inspiration from human visual perception. It focuses on quantifying interdependencies to amplify the association between words within the same phrase, while disregarding less relevant or noisy elements [42]. Self-attention [43] is applied to sequences after the addition of positional embeddings and processing through several linear layers to generate the Query (Q), Key (K), and Value (V) matrices. Self-attention can be computed as:

$$\text{Self-Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{D_k}}\right)V \tag{I.3}$$

Mixture of Experts (MoE) architectures [44] represent an advanced extension of Transformers, particularly relevant in the decoders of very large models, such as Large Language Models (LLMs). Architecturally, MoEs diverge from traditional monolithic neural networks by decomposing a complex problem into a collection of simpler sub-problems, each addressed by a specialized “expert” network. A critical component of this framework is the “gating network”, called router, which dynamically learns to route incoming data instances to the most appropriate expert or combination of experts. This conditional computation mechanism allows the model to selectively activate only a subset of its parameters for any given input, leading to substantial computational savings during inference while maintaining or even improving performance.

Mathematically, given N experts $E_1, E_2, \ldots, E_N$, and a gating function $g(x)$, the output of a MoE layer can be expressed as

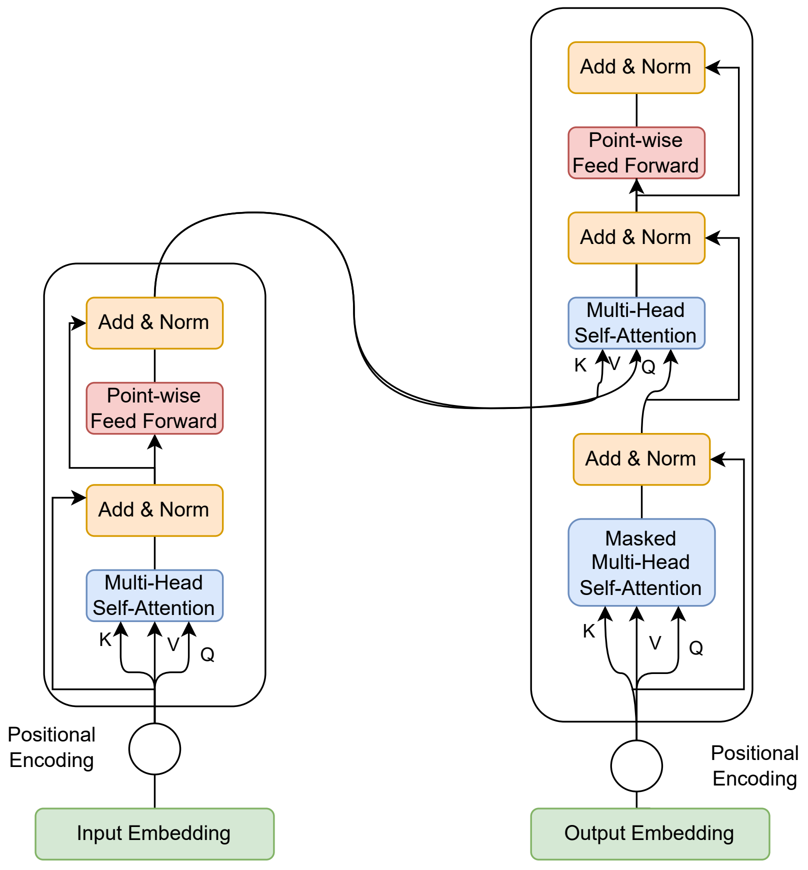


Figure 3: Transformer architecture [23]

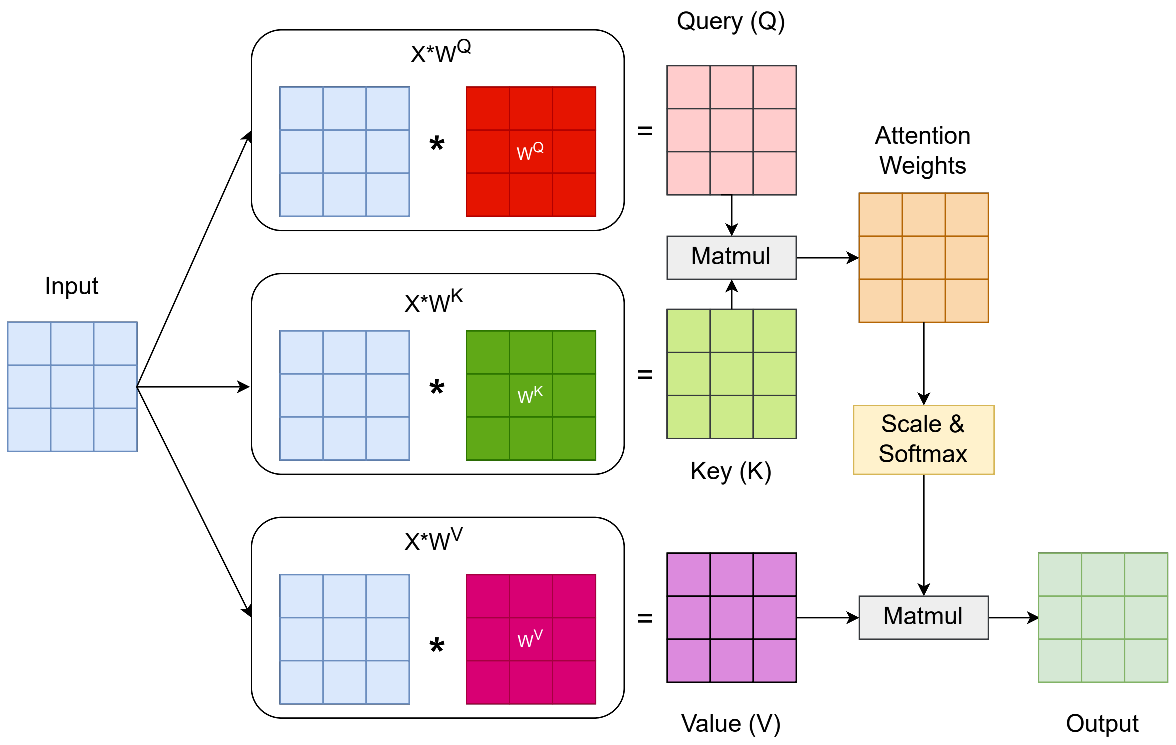


Figure 4: Self-attention [23, 42]

$$\mathrm{MoE}(x) = \sum_{i=1}^{N} g_i(x) E_i(x) \tag{I.4}$$

where $g_i(x)$ is the output of the gate network for expert i. As mentioned earlier, MoE enable models to scale to a massive number of parameters (billions or trillions) without proportionally increasing computational costs during inference. This is because only a subset of experts is activated for each input. The inherent modularity and sparsity induced by the gating mechanism contribute to MoEs' scalability and their ability to model highly complex, multi-modal relationships within data, making them a powerful tool for a wide range of applications from natural language processing to computer vision [45, 46]. Nonetheless, they present a significant challenge in terms of memory consumption, taking into account that, even if only a fex experts are active at any given time, all experts parameters need to be loaded into memory.

3.2 Green AI

Artificial Intelligence (AI) has emerged as a transformative force across all productive sectors. Specifically, Large Language Models (LLMs) are being integrated at a vertiginous pace into myriad facets of daily life for citizens, academia, and industrial activity. AI is disruptive, introducing a revolution in efficiency and the capacity for complex problem-solving. This growth has been driven by the increasing availability of computational resources, leading to significant advances in deep learning. However, this technological progress is not without environmental costs, given that the infrastructure supporting machine learning models consumes enormous quantities of resources.

Among these, the demand for electricity stands out, which already represents a notable increase in greenhouse gas emissions.

The deployment and use of deep neural networks entail a very considerable energy consumption, which manifests in two principal phases: training and inference. Firstly, the training phase is extremely intensive in computational and, consequently, energy resources. This process necessitates the simultaneous use of a considerable number of GPUs operating for extended periods. Pioneering models such as GPT-3 illustrate the magnitude of this consumption, having required 1287 MWh for training. Other models, including BLOOM [47], Gopher [48], and OPT [49], required 433 MWh, 1066 MWh, and 324 MWh, respectively [50]. It is estimated that the training of more advanced models, such as GPT-4, required approximately 1750 MWh. In the case of Llama 3.2, 581 MWh were consumed, an amount comparable to the energy consumed by a long-haul aircraft on a 7-hour flight [51]. This dynamic in energy consumption during the training phase highlights a growing demand for the resources necessary for the creation of these models and, consequently, efforts are required to develop models that are not only more capable from a generation or reasoning standpoint but also more sustainable and optimized.

Although model training is intensive, the inference phase—in other words, the daily use of these models to generate responses to user queries—becomes an even more significant energy factor due to the massive scale of their global adoption. Large technology companies have reported that inference workloads constitute a substantial portion of their energy consumption: in the case of Meta, 70% of energy consumption; in the case of Google, approximately 60% [52]. The company SemiAnalysis suggests that for ChatGPT, daily energy consumption amounts to 564 MWh [53, 54]. Other sources compare ChatGPT's annual consumption to the energy of over 35000 electric vehicles traveling 15000 km/year. In any case, the trend in consumption is increasing given that models are becoming progressively larger and have been democratized at a rapid pace. It is clear that optimizations in inference are crucial because this is the phase of greatest continuous use throughout a model's lifetime, and its cumulative impact is what truly pressures energy infrastructures.

In contrast to the prevailing trend within academia and industry, which prioritizes the unbridled scaling of model size for improved performance without considering environmental impact —a trend coined as Red AI— Green AI emerges as a crucial counter-paradigm in AI research [39]. This paradigm explicitly incorporates computational costs into the equation, advocating for a reduction in the resources expended during model training and deployment. To achieve this, Green AI expands beyond conventional machine learning metrics of efficacy to include efficiency-focused measures such as carbon emissions, electricity consumption, inference time, floating-point operations, and the number of parameters of the model. The latter metric, the number of parameters, is particularly valuable as it correlates directly with the computational workload of the model and is hardware-agnostic, representing an intrinsic characteristic of the models themselves. Furthermore, it exhibits a complete correlation with memory consumption.

As models continue to grow in size, specifically in their number of parameters, and given that inference for large and scalable models demonstrably consumes significantly more resources than training, network simplification methods have become indispensable tools for combating the environmental impact and excessive resource consumption associated with AI without sacrificing its advancements. These techniques, methods, or algorithms embody the principles of Green AI, ultimately enabling the deployment of highly powerful models in resource-constrained hardware environments, thereby facilitating broader access and research. Moreover, in the case of pruning methods, the simplification process inherently involves an analysis of the network architecture and its constituent neurons, which can yield valuable insights into optimal neural network design. Additionally, a model's explainability is enhanced by a deeper understanding of its internal processes during prediction [55]. As introduced by Gilpin *et al.*[56], comprehending the role of individual units in a neural network's inference and learning processes offers a viable pathway to endowing deep learning models with explainability. Consequently, neural network simplification methods are instrumental in constructing models aligned with the Green AI paradigm and hold significant potential in demystifying the prevalent "black box" perception of neural networks [57].

3.3 Deep Learning Model Simplification Techniques

The simplification of deep learning models constitutes a broad and active area of research, encompassing a wide array of strategies. While this doctoral thesis focuses on developing pruning algorithms, other notable alternatives include neuroevolution, neural architecture search, quantization, and knowledge distillation, among others. A description of each of these techniques follows.

3.3.1 Neuroevolution

Neuroevolution involves the application of evolutionary algorithms to optimize the parameters, connections, or architecture of neural networks. In contrast to gradient-based optimization, neuroevolutionary approaches explore an expansive search space, leading to the discovery of novel and efficient network configurations. These techniques necessitate the definition of a fitness function that considers both model efficacy and complexity, guiding the evolutionary process toward neural network architectures that are simultaneously simple and effective. However, these techniques are computationally intensive due to the vastness of their search space. Neuroevolution has been successfully employed to simplify convolutional neural networks such as Inception, DenseNet, and ResNet [58].

3.3.2 Neural Architecture Search

Neural Architecture Search (NAS) aims to automatically identify the optimal neural network architecture for a specific problem. NAS algorithms can be broadly categorized into two types: (1) micro-level search, which focuses on identifying the best operations at the neuron and layer level; and (2) macro-level search, which seeks to determine the optimal number of filters for each layer [59]. These techniques have been extensively applied to dense, convolutional, and recurrent networks [60], and more recently, versions adapted for LLMs such as Llama-nas have been proposed [61].

3.3.3 Quantization

Quantization is a technique that reduces the precision of the numerical representations used for neural network weights. Instead of employing high-precision floating-point numbers (e.g., 32-bit), quantization maps these numbers to lower-precision versions, such as 8-bit integers or even smaller representations [62]. This significantly reduces memory consumption and computational operations within the network, although it does entail a noticeable loss of information. Furthermore, it is a relatively straightforward technique as it does not require discerning the importance of network components or searching for alternative architectures. Consequently, quantization has been successfully adopted for reducing LLMs [63], even leading to highly popular fine-tuning methods for these models, such as QLoRA [64].

3.3.4 Knowledge Distillation

Knowledge Distillation [65, 66] is a method that facilitates the transfer of knowledge from one neural network to another. Specifically, a smaller “student” network learns from a larger, more complex “teacher” model. The objective is for the smaller model to achieve a performance level similar to that of the larger model, considering that the student is significantly smaller in size and complexity, thereby achieving network simplification. As documented in [67], this technique has been particularly applied to convolutional models, although initiatives also exist for Graph Neural Networks [68] and LLMs [69].

3.3.5 Pruning

Pruning algorithms represent one of the most effective tools for model compression. The underlying principle is to eliminate redundant connections, neurons, or filters. The hypothesis is that neural networks exhibit plasticity, and knowledge is distributed across their architecture, implying a significant number of parameters whose contribution to the network's inference process is negligible. Consequently, these parameters can be removed logically (by setting their weight values to zero, thus preventing their contribution to decisions) or physically (by eliminating these units, connections, or filters from the network entirely). The latter approach yields a true reduction in the number of parameters and, by extension, in the number of operations and memory required for network deployment. The literature presents a vast array of pruning approaches applicable to all types of networks, including feed-forward networks [70, 71, 72], convolutional networks [73, 74, 75], and Transformer architectures, encompassing both the encoder [76, 77, 78] and decoder components [79, 80]. Furthermore, a wide variety of techniques specifically target specialized decoders, such as Mixture of Experts (MoE) models [81, 82].

3.4 Mathematical tools to assess the most important neurons

For the construction of neural network simplification algorithms, it is necessary to employ mathematical tools that facilitate the assessment of neuron relevance, specifically identifying those neurons that contribute significant predictive information and, consequently, those that can be disregarded. Throughout this thesis, various strategies have been devised, spanning statistical methods, algebraic topology, and information theory. Herein, each utilized metric is described.

3.4.1 Statistical Approach

The statistical approach, employed in the CNN simplification method (Objective 2), called OCNNA, leverages three statistical or data analysis tools.

Firstly, it applies Principal Component Analysis (PCA) [83], an algorithm designed for dimensionality reduction. This method utilizes the covariance matrix, which quantifies the variance among distinct variables, and subsequently extracts eigenvalues via the Spectral Theorem [84]. The eigenvectors associated with the largest eigenvalues represent the principal components of the input data, enabling the projection of data into the principal component space and thereby reducing dimensionality.

Subsequently, the Frobenius norm [85] is employed, calculated as:

$$||A||_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{N}|a_{ij}|^2} \tag{I.5}$$

Finally, the Coefficient of Variation is utilized, which establishes the relationship between the mean and the standard deviation of a data distribution D [86]. Specifically, it is calculated as:

$$CV_D = \frac{\sigma_D}{\mu_D} \tag{I.6}$$

This proves to be a valuable tool as it enables the dimensionless comparison of variables, independent of their scales.

3.4.2 Persistent homology-based Approach

Homology theory, a branch of algebraic topology, plays a crucial role in data analysis. Persistent homology, in particular, focuses on identifying and quantifying topological features within data sets across varying spatial scales and resolutions [87]. To conduct this analysis, data are represented as a simplicial complex, from which a filtration is subsequently generated. A filtration is defined as a nested sequence of these simplicial complexes [88]. This sequence is parameterized, and as the parameter evolves, topological features emerge and vanish, a phenomenon typically visualized through a Birth-Death diagram [89]. Persistent homology has become a potent tool within Topological Data Analysis (TDA), proving valuable for the detection of patterns and structures in complex datasets.

For the simplification of the Transformer-encoder (Objective 3), the PBCE method was designed. This network simplification approach is grounded in zero-dimensional persistent homology, a tool utilized to discriminate the importance of each component within the neural network. This discrimination allows for the application of varying pruning levels based on component redundancy.

3.4.3 Information theory-based Approach

Information theory [90] constitutes a branch of mathematics dedicated to the quantification and communication of information. Entropy is among the most prominent metrics within this field. It quantifies the amount of uncertainty or information associated with a random variable [91]. If $p(x)$ is the probability of the event $x \in X$, the entropy of a discrete random variable X is calculated as

$$H(X) = -\sum_{x} p(x)\log p(x) \tag{I.7}$$

In other words, entropy indicates the amount of information contained within a vector. Similarly, joint entropy, denoted as $H(X,Y)$ can be defined as the inherent uncertainty within two random variables. Another fundamental metric is mutual information, which quantifies the statistical information shared between two distributions. It is calculated as:

$$I(X,Y) = H(X) + H(Y) - H(X,Y) \tag{I.8}$$

From mutual information, Normalized Mutual Information (NMI) is derived, which offers enhanced interpretability and comparability. Following the notation proposed in [92], we define NMI as

$$\mathrm{NMI}(X,Y) = \frac{I(X,Y)}{\sqrt{H(X)H(Y)}} \tag{I.9}$$

Therefore, given random variables X, Y, a high $\mathrm{NMI}(X,Y)$ value indicates that X and Y share a significant amount of information, implying they contain redundant information. Conversely, if $\mathrm{NMI}(X,Y)$ is close to 0, it can be asserted that X and Y have little in common. Leveraging these mathematical definitions, NMI was employed to construct a simplification method for the Transformer-decord, specifically for MoE-LLMs (Objective 3).

4 Objectives

Following the introduction of key concepts from the state of the art, the objectives of this Ph.D. thesis are presented. As previously discussed, the ultimate aim of this doctoral research is to develop simplification algorithms for deep learning models. These algorithms seek to reduce the environmental impact generated by these networks when deployed in production systems, thereby enabling their implementation in hardware-constrained environments. Furthermore, this work aims to conduct a qualitative and quantitative analysis of the role and importance of individual neurons or units within these networks, with the potential to inspire novel architectures and designs. The specific objectives are detailed below:

1. **Design and Implementation of a Simplification Algorithm for Dense Feed-Forward Neural Networks:** As a starting point for this doctoral thesis, a method for simplifying dense feed-forward neural networks will be proposed. This involves establishing various classes of networks as benchmarks, differentiated by both size and the specific learning tasks they are designed to solve.

2. **Design and Implementation of a Simplification Algorithm for Convolutional Neural Networks:** Prior to the advent of Large Language Models, Convolutional Neural Networks represented the largest established deep learning models. Major competitions, such as the ImageNet Large Scale Visual Recognition Challenge, led to the development of pre-trained convolutional architectures that serve as benchmarks for problems like image classification. This objective aims to construct a method capable of identifying the most crucial convolutional filters to simplify these pre-trained networks and compare the resulting performance with the current state of the art.

3. **Design and Implementation of a Simplification Algorithm for the Transformer architecture:** While the initial research plan for this thesis proposed developing a pruning technique for Recurrent Neural Networks, the landscape of specialized sequential neural networks (e.g., natural language, time series, speech) has undergone a significant transformation during the course of this doctoral work. The explosion of Transformers as the absolute vanguard in this domain has led to a strategic pivot. Consequently, a method for simplifying the Transformer architecture, encompassing both its encoder and decoder components, is now deemed to generate substantially greater impact. This shift is particularly pertinent given that the natural evolution of these models, as previously noted, involves a massive increase in their parameter count to tackle increasingly complex problems. Therefore, this objective is now dedicated to an in-depth study of the Transformer architecture to propose distinct simplification algorithms for both a Transformer-encoder and a Transformer-decoder. This focus extends specifically to even larger model versions, such as Mixture-of-Experts Large Language Models, aligning with the cutting edge of current research.

4. **Deployment of Simplified Networks for Solving Complex Machine Learning Problems:** Neural networks are extraordinarily useful for solving supervised machine learning problems. Following the development of simplified versions through the pruning methods defined in the preceding objectives, this aim focuses on applying these simplified (and often more effective) versions to address complex problems across disciplines such as biomedicine, economics, energy, and smart cities.

5 Methodology

This thesis rigorously adheres to the principles of the traditional scientific method, integrating both theoretical exposition and empirical investigation throughout its development. The research methodology, particularly concerning the design and execution of experiments, is structured around the following systematic stages:

1. **Observation and Problem Identification:** The initial phase involved a comprehensive analysis of extant challenges within large-scale neural networks, specifically the computational and environmental costs associated with their immense parameter counts and extensive training data. Concurrently, an in-depth study of Green AI paradigms and their proposed metrics was undertaken to understand their potential in addressing these sustainability concerns. A particular focus was placed on identifying the inherent inefficiencies and redundancy within complex neural network architectures, recognizing that simplification is a fundamental step towards more sustainable and efficient AI. This phase aimed to precisely delineate the limitations of current over-parameterized models when confronted with practical deployment constraints and environmental considerations.

2. **Hypothesis Formulation and Algorithm Design:** Based on the identified needs for more efficient and sustainable AI, the central hypothesis of this research was formulated: that novel neural network simplification algorithms can be designed and developed to effectively reduce model complexity without significant degradation in performance. This stage involved the conceptualization and architectural design of pruning and simplification techniques specifically tailored for various neural network types, including dense feed-forward networks, convolutional neural networks, and Transformer architectures (both encoder and decoder components). Emphasis was placed on proposing solutions that offer both theoretical soundness in identifying redundant components and practical applicability in real-world scenarios.

3. **Data Collection and Experimental Setup:** This stage focused on the systematic gathering of empirical data through controlled experimentation. The proposed simplification algorithms were deployed and evaluated on diverse, large-scale datasets and pre-trained neural network models. A comprehensive set of performance measures was employed to quantify their effectiveness in terms of simplification ratio and the resulting impact on model performance and efficiency. Primary metrics included, but were not limited to, predictive accuracy for assessing maintained model efficacy and reduction in parameter count for quantifying simplification for evaluating efficiency gains. Rigorous control over experimental conditions was maintained to ensure the reliability and reproducibility of the obtained results across different network architectures.

4. **Hypothesis Contrasting and Comparative Analysis:** The results obtained from the proposed simplification algorithms were subjected to a stringent comparative analysis against established pruning techniques and other model compression methods documented in the literature. This phase critically evaluated the quality of the proposed solutions in terms of both efficiency and effectiveness. To ensure a robust comparison, widely adopted machine learning frameworks and benchmark datasets were utilized, providing a standardized basis for evaluation across different neural network types.

5. **Hypothesis Validation, Refinement, or Refutation:** The culmination of the experimental and comparative analysis led to the formal validation or refutation of the initial hypothesis regarding the efficacy of the proposed simplification methods. If the empirical evidence

consistently supported the proposed hypothesis, it was deemed validated. Conversely, if the results did not align with the hypothesis, a systematic rejection of the hypothesis was undertaken, followed by a meticulous re-evaluation of the underlying assumptions, algorithm designs, and experimental protocols. This iterative process of refinement and re-testing was crucial to ensure the quality and robustness of the final research outcomes in neural network simplification.

6. **Dissemination and Scientific Contribution:** The final stage involved the extraction and formal acceptance of conclusions derived from the entire research process concerning neural network simplification. This included the comprehensive documentation of the methodologies, experimental procedures, results, and insights gained from developing and applying these algorithms. The findings were subsequently compiled into journal publications and the complete doctoral dissertation.

6 Summary

This section summarizes the primary proposals and studies conducted in the publications associated with the thesis. Subsequently, Section 6 presents the main results obtained from each article. The journal publications are listed below:

- Balderas, L., Lastra, M., & Benítez, J. M. (2023). Optimizing dense feed-forward neural networks. *Neural Networks*, 171, 229-241. https://doi.org/10.1016/j.neunet.2023.12.015 (Objective 1).
- Balderas, L., Lastra, M., & Benítez, J. M. (2024). Optimizing Convolutional Neural Network Architectures. *Mathematics*, 12(19), 3032. https://doi.org/10.3390/math12193032 (Objective 2).
- Balderas, L., Lastra, M., & Benítez, J. M. (2025). A Green AI Methodology Based on Persistent Homology for Compressing BERT. *Applied Sciences*, 15(1), 390. https://doi.org/10.3390/app15010390 (Objective 3)
- Balderas, L., Lastra, M., & Benítez, J. M. (2025). MoEITS: A Green AI approach for simplifying MoE-LLMs. Submitted (Objective 3).
- Balderas, L., Lastra, M., & Benítez, J. M. (2024). An Efficient Green AI Approach to Time Series Forecasting Based on Deep Learning. *Big Data and Cognitive Computing*, 8(9), 120. https://doi.org/10.3390/bdcc8090120 (Objective 4).
- Balderas, L., Pérez, F.J., Moreno de Castro, M., Lastra, M., Martínez, J.P., Lainez-Ramos, A.J., Arauzo, A., & Benítez, J.M. (2025). On MRI-Based Central Nervous System Tumor Differentiation using Deep Neural Networks and Conformal Predictors (Objective 4).

The remainder of this summary is organized according to the publications and objectives outlined in Section 3, which were achieved through the application of the methodology introduced in Section 4. Firstly, Section 5.1 presents the algorithm for simplifying dense feed-forward neural networks. Subsequently, Section 5.2 introduces the method for simplifying convolutional neural networks. Following this, Section 5.3 includes methods for simplifying the Transformer architecture, encompassing both the encoder and the decoder. Finally, in Section 5.4, the applications of several techniques presented in the preceding sections for solving complex data science problems are discussed.

6.1 Design and Implementation of a Simplification Algorithm for Dense Feed-Forward Neural Networks

This section introduces ODF2NNA (Optimizing Dense Feed-Forward Neural Network Algorithm), the dense feed-forward neural network simplification algorithm related to the first objective of the thesis. ODF2NNA is applicable to both classification and regression problems and comprises three distinct phases. In the initial phase, a model with an equal number of neurons across all layers is constructed and subsequently trained to address the classification task. Following this, the network undergoes pruning, and the resulting simplified architecture is refined through re-training for a limited number of epochs. The algorithm incorporates a parameter, ϵ, which dictates the tolerance level for pruning (a smaller ϵ implies more intense pruning).

As with any pruning methodology, the criterion for retaining or eliminating neurons is paramount to the algorithm's efficacy. The procedure for neuron selection is detailed as follows. Initially, a designated dataset, D, distinct from the training set, is utilized to assess the importance of individual neurons. For each neuron within each layer of the network, D is evaluated, and the output of the respective neuron is analyzed. Subsequently, the standard deviation of these outputs is computed and compared against the ϵ parameter. If the standard deviation exceeds ϵ, the neuron is deemed to generate sufficient variability in its output, thus warranting its retention within the network. Conversely, if the standard deviation is lower, the neuron is marked as redundant and is consequently excluded from the simplified network. To mitigate a significant impact of pruning on the network's overall performance, the mean of the outputs from the removed neurons is stored in the bias term. This value closely approximates the actual output of the neuron, given its assessed redundancy. Consequently, a portion of the contribution from the eliminated neurons persists within the simplified model. Before finishing the pruning process, the network undergoes re-training for a small number of epochs (typically between 10 and 20) to fine-tune its final output. This systematic approach enables the construction of highly simplified versions of feed-forward neural networks, tailored to desired levels of parsimony.

To evaluate the proposed method's quality, a thorough experimentation was conducted using well-known datasets from the UCI repository (for both classification and regression tasks) and MNIST, where ODF2NNA's performance was compared against state-of-the-art techniques. Furthermore, the algorithm was applied to established architectures such as LeNet300-100. Finally, large-scale datasets, including Hepmass and Higgs, were also utilized for evaluation.

The research article presenting ODF2NNA is as follows:

Balderas, L., Lastra, M., & Benítez, J. M. (2023). Optimizing dense feed-forward neural networks. *Neural Networks*, 171, 229-241. https://doi.org/10.1016/j.neunet.2023.12.015

6.2 Design and Implementation of a Simplification Algorithm for Convolutional Neural Networks

This section, related to the second objective of the thesis, introduces OCNNA (Optimizing Convolutional Neural Networks Architectures), an algorithm developed for the simplification of Convolutional Neural Networks (CNNs). The algorithm takes a parameter, k, which specifies the percentile of filters to be retained in the simplified model based on their importance. Consequently, a higher value of k results in a more aggressively simplified network.

The algorithm's operational procedure is as follows: A dedicated dataset, D, is established for the sole purpose of assessing the importance of convolutional filters. Each convolutional layer and its respective filters are then iteratively processed. For a given filter, its output is generated by evaluating it on dataset D. Subsequently, a triad of statistical tools is applied to this output. First, Principal Component Analysis (PCA) is performed, retaining 95% of the variability to reduce dimensionality. Next, the Frobenius norm is applied to each of these lower-dimensional representations, corresponding to individual images in dataset D, to obtain a concise summary of the information. Finally, the Coefficient of Variation (CV) of these matrix norms is calculated to yield a numerical score representing the importance of the evaluated filter. In essence, OCNNA assigns a low importance score to a convolutional filter if, when applied to a set of images, its output's internal structure (extracted via PCA) —after being summarized (Frobenius norm)— exhibits low variability (measured by CV).

The empirical evaluation of OCNNA was conducted using established benchmarks for image classification, encompassing both evaluation datasets and well-known pre-trained convolutional neural network models. Specifically, the datasets employed include CIFAR-10, CIFAR-100, and ImageNet. The architectures utilized comprise ResNet-50, VGG-16, DenseNet-40, and MobileNet. The results obtained with OCNNA are rigorously compared against over 20 state-of-the-art CNN simplification techniques.

The research article presenting OCNNA is as follows:

Balderas, L., Lastra, M., & Benítez, J. M. (2024). Optimizing Convolutional Neural Network Architectures. *Mathematics*, 12(19), 3032. https://doi.org/10.3390/math12193032

6.3 Design and Implementation of a Simplification Algorithm for the Transformer architecture

This section, related to Objective 3, presents the methodologies devised for simplifying Transformer architectures. Primarily, focusing on the encoder, the Persistent BERT Compression and Explainability (PBCE) algorithm is proposed. This algorithm centers on simplifying models within the BERT family through the application of persistent homology.

Specifically, PBCE is a Green AI methodology that leverages zero-dimensional persistent homology to extract the topological features of neuron outputs, thereby enabling an assessment of their importance. This approach facilitates the identification of neurons that are most useful, contributing more information, and those that can be pruned due to their lesser relevance. PBCE systematically executes the following steps: First, a corpus is selected to evaluate neuron importance. Given its focus on BERT-based models, the Wikipedia dataset [93] serves as the chosen corpus. Subsequently, for each layer and neuron, zero-dimensional persistent homology is computed. The underlying principle is to derive a ranking of the most relevant neurons for each component by evaluating the richness of the topological features generated by their outputs. Once ordered, the topological information is analyzed at the component level, and pruning is applied at three intensity levels, determined by whether the network component contains more or fewer neurons with significant topological features. Finally, upon construction of the simplified model, its performance is evaluated against the General Language Understanding Evaluation (GLUE) Benchmark [94], a reference benchmark formed by eight distinct Natural Language Processing (NLP) tasks. PBCE has been applied to both BERT Base and BERT Large models and its performance has been compared against more than ten state-of-the-art techniques.

The research article presenting PBCE is as follows:

Balderas, L., Lastra, M., & Benítez, J. M. (2025). A Green AI Methodology Based on Persistent Homology for Compressing BERT. *Applied Sciences*, 15(1), 390. https://doi.org/10.3390/app15010390

Regarding the simplification of the Transformer-decoder, MoEITS (Mixture of Expert Information Theory Simplifier) is introduced. This algorithm is designed for the compression of MoE-LLMs and is based on information theory. Specifically, the Normalized Mutual Information (NMI) metric is utilized to quantify the redundancy level of experts within each block. The algorithm takes a parameter τ, which dictates the intensity of the pruning performed.

The procedure is as follows: for each block, the NMI is computed for all pairs of experts, constructing a redundancy matrix R that encompasses all these values. From this matrix, MoEITS applies an iterative algorithm to select the experts that contribute the most information and should therefore be retained in the simplified model. Given a redundancy level ρ derived from τ, the algorithm iterates over R, searching for values that exceed ρ. If such a value exists at position (i, j), it implies that experts e_i and e_j exhibit a redundancy level surpassing the defined limit; consequently, one of them must be eliminated. To decide which expert to prune, it becomes necessary to measure their respective redundancy levels with the remaining experts. This is accomplished by calculating the average redundancy levels of e_i and e_j with the other experts. Should e_i possess the higher average, it is pruned from the model. The information associated with e_i is then removed from R, and the iterative process continues until no pair of experts exceeds the redundancy limit ρ.

For evaluation, the Mixtral 8×7B model serves as the reference. Following simplification, the model is evaluated across various datasets, including BoolQ, HelloSwag, WinoGrande, ARC-e, and ARC-c. Furthermore, MoEITS is compared against five state-of-the-art techniques in MoE-LLM simplification.

The research article presenting MoEITS is as follows:

Balderas, L., Lastra, M., & Benítez, J. M. (2025). MoEITS: A Green AI approach for simplifying MoE-LLMs. Submitted

6.4 Deployment of Simplified Networks for Solving Complex Machine Learning Problems

The fourth objective of this thesis focuses on solving complex data science problems by applying neural networks enhanced with the proposed simplification algorithms. This section begins by detailing the GreeNNTSF methodology, which is designed for constructing simplified deep learning models specifically for time series forecasting.

GreeNNTSF utilizes ODF2NNA (Objective 1) as its primary simplification method. This allows for the development of neural networks capable of addressing diverse problems across various domains, including economics, smart cities, medicine, and climate. To demonstrate its effectiveness, GreeNNTSF is applied to a selection of real-world datasets that serve as established benchmarks for time series forecasting: Sars-Cov-2, Brazilian Weather, PhysioNet, WTI, Traffic, and Electricity.

For rigorous evaluation, GreeNNTSF is compared against various state-of-the-art techniques. Performance is assessed using standard metrics such as Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE), as well as specialized metrics like DILATE, which are prevalent in the current literature for time series forecasting.

Balderas, L., Lastra, M., & Benítez, J. M. (2024). An Efficient Green AI Approach to Time Series Forecasting Based on Deep Learning. *Big Data and Cognitive Computing*, 8(9), 120. https://doi.org/10.3390/bdcc8090120

This subsequent application focuses on a research project aimed at characterizing glioblastomas. This endeavor begins by constructing a perfusion curve from the perfusion sequence of a Magnetic Resonance Imaging (MRI) scan. This effectively transforms an image-based problem into a time series classification task with several key objectives:

- Differentiating gliomas between low-grade (LGG) and high-grade (HGG).
- For low-grade gliomas, distinguishing between WHO Grades I and II.

- For high-grade gliomas, differentiating between WHO Grades III and IV.

To address this complex learning problem, cascaded classification systems incorporating conformal predictors [40] are developed to enhance trustworthiness. These classification systems feature an initial MCDCNN (Multi-Channel Deep Convolutional Neural Network), simplified using the OCNNA algorithm, to differentiate between high-grade and low-grade gliomas. Subsequently, for high-grade cases, a feed-forward neural network, simplified with ODF2NNA, is integrated to distinguish between WHO Grades III and IV.

This work represents a multidisciplinary effort where the pruning algorithms proposed in this thesis directly impact the efficiency and quality of the predictive systems. The journal article related to this project is as follows:

Balderas, L., Pérez, F.J., Moreno de Castro, M., Lastra, M., Martínez, J.P., Lainez-Ramos, A.J., Arauzo, A., & Benítez, J.M. (2025). On MRI-Based Central Nervous System Tumor Differentiation using Deep Neural Networks and Conformal Predictors. Submitted.

7 Discussion of results

This section summarizes the main results obtained for each thesis objective, in relation to the publications previously outlined.

7.1 Design and Implementation of a Simplification Algorithm for Dense Feed-Forward Neural Networks

As previously described, ODF2NNA, a simplification algorithm for dense feed-forward neural networks, was designed and implemented. This algorithm has been applied to classification and regression problems, demonstrating its capacity to reduce model complexity and enhance predictive performance. The primary results are analyzed below.

Firstly, experiments demonstrate superior performance on small-scale datasets, where ODF2NNA outperforms other pruning techniques, achieving higher accuracy and reducing model complexity. When applied to reference network architectures, such as LeNet 300-100, the proposed method significantly reduces the number of parameters (by up to 98% for this specific architecture). Such reduction is crucial for computational efficiency and implementation in resource-constrained environments. Indeed, experimentation indicates that ODF2NNA is also effective with large-scale tasks, where a substantial reduction in the number of parameters (between 20% and 40%) is achieved without compromising predictive capability. Consequently, ODF2NNA proves versatile for application across small, medium, and large network contexts.

Moreover, ODF2NNA not only simplifies models but also enhances prediction accuracy. Experiments reveal that for certain values of the ϵ parameter, the simplified models achieve accuracy improvements ranging from 15% to 40% compared to the original model. This indicates that the method generates models that are not only lighter but also more effective.

Finally, an interesting experimental finding is that ODF2NNA not only guides the construction of more efficient models but also generates more accurate neural networks compared to those conceived with the simplified architecture from the very beginning. Experiments demonstrated that models directly constructed with a similar number of parameters and form to the architecture found did not achieve the same level of performance as the models simplified by ODF2NNA. This suggests that the technique facilitates a transfer of knowledge between the original and the simplified model, which is not easily attainable when starting with a reduced in size model.

7.2 Design and Implementation of a Simplification Algorithm for Convolutional Neural Networks

The OCNNA algorithm is presented as a robust method for the simplification of CNNs. It offers an effective solution, achieving a significant reduction in the complexity of well-known pre-trained models with minimal, negligible, or even positive impact on the accuracy of the resulting model. OCNNA is noteworthy for its versatility, being applicable across diverse network architectures such as ResNet-50, VGG-16, DenseNet-40, and MobileNet. The principal findings are detailed below.

Experiments conducted on the CIFAR-10, CIFAR-100, and ImageNet datasets, which serve as established benchmarks for image classification, demonstrate the superiority of OCNNA. Specifically, its performance on the CIFAR datasets is outstanding, consistently achieving optimal results in terms of both parameter reduction and accuracy.

- For ResNet-50, OCNNA retained approximately 45% of the original parameters with a very small loss in accuracy. This highlights the algorithm's efficiency in compressing large models while preserving performance.
- In the case of VGG-16, a notoriously complex network, the algorithm achieved a remarkable reduction of up to 86.68% of the parameters for CIFAR-10, notably resulting in a 0.42% improvement in accuracy. For CIFAR-100, the accuracy remained almost unchanged, underscoring OCNNA's capacity to significantly prune even highly redundant architectures without performance degradation.
- With DenseNet-40, OCNNA not only reduced complexity but also improved accuracy by 0.39% for CIFAR-10 and 0.23% for CIFAR-100. Such improvements are particularly valuable, as they indicate that the pruning process can also act as a regularization mechanism, enhancing generalization.
- For MobileNet, OCNNA excelled in both parameter reduction ratio and accuracy, further improving test accuracy by 0.65% for CIFAR-10. This demonstrates its effectiveness on lightweight, mobile-oriented architectures, which are critical for deployment on edge devices.

Furthermore, while OCNNA did not always achieve the highest absolute compression on ImageNet, its balance between size reduction and accuracy retention was notable. Specifically, for ResNet-50, OCNNA yielded a comparable parameter reduction to other methods but with a significantly lower accuracy loss. For MobileNet, the proposed method surpassed most existing approaches, solidifying its competitive standing in large-scale image classification tasks.

OCNNA is characterized by a single parameter, k, which represents the percentile of filters with the highest importance, effectively controlling the intensity of the pruning process. A sensitivity analysis of k was conducted, yielding the following conclusions:

- Reduction Control: As the value of k increases, OCNNA enhances the reduction in the number of filters, leading to a substantial decrease in parameters. This provides a direct mechanism for controlling model size.
- Accuracy-Compression Balance: Conversely, an excessively high k value can lead to a drastic drop in accuracy. The study revealed that a value of $k = 40$ was the best found, offering the highest possible accuracy while achieving a significant reduction in parameters.

These findings are crucial as it enables users to customize network simplification based on specific application requirements, balancing computational resource demands with accuracy prerequisites. This adaptability is a key advantage for deploying pruned models in diverse operational environments.

7.3 Design and Implementation of a Simplification Algorithm for the Transformer architecture

PBCE algorithm emerges as a highly effective technique for the simplification of Transformer encoder models, such as BERT. Its primary strength lies in its ability to substantially reduce the number of parameters in the original model while maintaining, and in some cases even improving, performance across a wide array of NLP tasks. This notable achievement is accomplished with a relatively concise fine-tuning process, requiring only 40 epochs.

Comprehensive experiments conducted on BERT Base, utilizing the widely accepted GLUE Benchmark for language model evaluation, unequivocally demonstrate PBCE's superiority over most state-of-the-art techniques.

Specifically, PBCE achieved a substantial parameter reduction, compressing the model to just 47% of its original parameter count (from 110M to 52M). Despite this significant compression, PBCE consistently enhanced the performance of the original model across the majority of tasks, registering notable improvements in QQP (+20), QNLI (+1.2), CoLA (+10.86), STS-B (+3.72), MRPC (+2.19), and RTE (+5.58). While it did not surpass all existing techniques on SST-2 and RTE, its results remained highly competitive, underscoring its generalizability. Furthermore, the accuracy results of PBCE indicate that, despite other techniques achieved higher parameter reductions, PBCE offers the best balance between model accuracy and size, a crucial trade-off in practical applications.

For BERT Large, where fewer experimental results are documented in the existing literature, PBCE similarly exhibited outstanding performance. The algorithm achieved an even more significant compression, reducing the model to only 43% of its original parameters (from 340M to 146.2M). This greater reduction is attributed to the increased inherent redundancy typically found in larger models. Furthermore, PBCE outperformed the RPP technique across most tasks (with the exceptions of MRPC and RTE) and improved the performance of the original model in MNLI (0.4/0.33), QNLI (0.45), and CoLA (1.28).

These findings substantiate that persistent homology is a valuable tool for the analysis and selection of the most relevant neurons in the predictive process. Since PBCE identifies the most significant units based on the topological characteristics of their outputs, it significantly contributes to the explainability of neuronal roles within the Transformer architecture. This enhanced interpretability facilitates a deeper human understanding of the internal decision-making processes within the model, addressing a critical aspect of complex deep learning systems.

On the other hand, for Transformer-decoder simplification, MoEITS is introduced. Empirical results demonstrate that MoEITS represents a highly effective and superior solution compared to existing state-of-the-art proposals, particularly in achieving an optimal balance between model accuracy and size reduction.

Comparative experiments rigorously conducted revealed that MoEITS consistently achieves superior performance across several key reasoning and comprehension tasks. The method exhibited higher accuracy results than other leading approaches on the ARC-c, HelloSwag, and ARC-e benchmarks. While MoE-Pruner demonstrated superior accuracy on WinoGrande and BoolQ, it is critical to note that this method relies on sparsity-inducing techniques, which, as further elaborated, do not inherently reduce the physical size of the model. Consequently, such approaches are less applicable in hardware-constrained environments. In contrast, MoEITS achieves a remarkable reduction in network size, successfully compressing the model to only 43% of its original parameter count.

A significant contribution of this work lies in its detailed discussion and rigorous comparison between simplification methods based on parameter pruning and those employing sparsity-inducing techniques. Parameter pruning methods, exemplified by MoEITS, perform a "physical simplification" of the network. This involves the explicit removal of redundant or inconsequential neurons and connections from the model's architecture. Such a structural modification directly translates to a tangible reduction in floating-point operations (FLOPS) and memory footprint during inference. This approach is fundamental to the Green AI initiative, as it demonstrably decreases the computational resources and energy required. Conversely, sparsity-inducing techniques, such as MoE-Pruner or

STUN, operate by encouraging a subset of network weights to converge to zero, resulting in a sparse weight matrix. However, it is emphasized that this represents a “logical simplification” rather than a physical one.

Besides, an in-depth ablation study was conducted to analyze the impact of the parameter τ, which governs the level of simplification applied to the model (a smaller τ value corresponds to more aggressive pruning).

A monotonic relationship was observed between τ and the number of parameters, where a smaller τ consistently led to greater parameter reduction. However, model accuracy did not follow a direct monotonic relationship. Specifically, on the BoolQ benchmark, a peak accuracy value was identified around $\tau = 0.75$, at which the model achieved a notably superior performance. For τ values less than 0.75, the model's performance drastically decreased. This comprehensive analysis demonstrates that parametric pruning algorithms like MoEITS are particularly valuable for identifying the optimal trade-off between model performance and size. Such an ability enables the generation of models that can effectively meet the stringent hardware constraints of a target deployment device while maximizing accuracy.

7.4 Deployment of Simplified Networks for Solving Complex Machine Learning Problems

First of all, the GreeNNTSF methodology results are introduced. GreeNNTSF's effectiveness was rigorously evaluated across several real-world datasets, consistently demonstrating superior or highly competitive performance:

- SARS-CoV-2: Unlike a single-model approach, GreeNNTSF constructs individual models for each of 188 countries, leading to more tailored and accurate predictions. This ensemble approach achieved a remarkable 19% reduction in Mean Absolute Error (MAE) and an 81% reduction in Root Mean Squared Error (RMSE) compared to the reference REGENN model, showcasing a substantial improvement in forecast accuracy for critical public health data. GreeNNTSF also yielded competitive results for Mean Squared Log Error (MSLE).

- Brazilian Weather: For this dataset, involving 253 different sensors, GreeNNTSF built a dedicated model for each sensor. The methodology achieved the best results across all metrics, securing a 13.5% reduction in MAE, a 40% reduction in RMSE, and a 17% reduction in MSLE. This highlights GreeNNTSF's ability to capture complex weather patterns with enhanced precision and efficiency.

- PhysioNet: In analyzing data from 11,988 ICU patients, GreeNNTSF generated an ensemble of models, one for each patient. The method delivered more competitive results than the reference algorithm, achieving a 19% reduction in MAE, a 14% reduction in RMSE, and a 36% reduction in MSLE. This demonstrates the method's potential for critical short-term physiological predictions.

- WTI Crude Oil: When predicting WTI crude oil prices, GreeNNTSF outperformed existing approaches in RMSE, Mean Absolute Percentage Error (MAPE), and R2 metrics. Although a VMD-GRU configuration yielded a slightly better MAE, GreeNNTSF still proved highly effective, particularly when compared to dense neural network approaches that exhibited significantly higher error rates.

- Traffic: For hourly road occupancy prediction in California, GreeNNTSF achieved better results for both MSE and DILATE, a specialized forecasting metric. This indicates GreeNNTSF's strong capability in capturing dynamic traffic patterns.
- Electricity: In the electricity consumption forecasting problem, GreeNNTSF's methodology not only outperformed the complex ES-dRNNe ensemble model across all three metrics (5% reduction for MAPE, 20% for RMSE, and 54% for MdAPE) but also achieved this with a simplified model, reducing its parameter count by 35% (from 280000 to 184207 parameters). This outcome strongly aligns with the principles of Green AI, demonstrating that superior performance can be achieved with significantly more efficient models.

The consistent results across these diverse real-world problems underscore GreeNNTSF's efficacy in generating significantly more compact and efficient models for time series forecasting

Finally, the results of the AI-based system to differentiate between glioblastomas are introduced. For the initial classification stage, which distinguishes between High-Grade Glioma (HGG) and Low-Grade Glioma (LGG), the Multi-Channel Deep Convolutional Neural Network (MCDCNN), simplified using OCNNA, emerged as the optimal choice. This model consistently demonstrated the best performance across all four evaluation metrics considered in the study. A notable finding in this stage was that the application of conformal prediction significantly benefited most considered techniques, with the exception of Random Forest. Furthermore, the absence of conformalization would have led to a different selection for the best-performing model in this critical initial stage, highlighting the transformative impact of this calibration technique.

Transitioning to the second stage, focused on classifying LGG into Grade 1 and Grade 2, the 1NN-DTW (1-Nearest Neighbor with Dynamic Time Warping) model proved to be the most effective. The integration of conformal prediction further boosted the performance of 1NN-DTW, allowing it to achieve superior results by a larger margin than observed in the preceding stage.

Finally, for the HGG classification into Grade 3 and Grade 4, a Dense Neural Network simplified with ODF2NNA was selected. This decision was predicated on its superior performance in terms of F1-score, Brier Loss, and Log Loss. While its accuracy was marginally lower than the top-performing Shapelet and SVC techniques, its overall strong showing across multiple critical metrics made it the preferred choice for this challenging classification task.

The impact of conformalization on the classification process was further elucidated through detailed visualization. For instance, in the initial HGG/LGG classification stage, conformalization notably rectified several misclassifications. Samples that were incorrectly predicted by the non-calibrated model were correctly classified after applying conformal prediction, as evidenced by the alignment of the calibrated probability with the true label. However, the analysis also revealed instances where, even with conformal probabilities, some samples remained misclassified. In such cases, the system could alert the user about low prediction confidence, indicated by a large probability range, which is crucial for clinical applications. Conversely, certain misclassified samples by the calibrated model exhibited high confidence levels, suggesting a potential limitation, possibly attributable to the scarcity of available data for optimal conformalization. This underscores the intricate interplay between data availability, model performance, and the effectiveness of calibration techniques in achieving robust and reliable predictions for brain tumor differentiation.

8 Conclusions and future work

This section presents the main conclusions of the thesis and provides some future research lines.

8.1 Conclusions

Deep neural networks, in their myriad forms and topologies, have long been instrumental in the development of innovative machine learning solutions, experiencing a recent surge in popularity across industry, academia, and the public domain. As the scale, complexity, and computational demands of these networks continue to escalate in the pursuit of ever-greater problem-solving capabilities, the Green AI paradigm and network simplification algorithms have become indispensable. These approaches are crucial for research groups, small companies, and individual users to leverage advanced computational intelligence without requiring the colossal computing resources available only to large technology corporations. This doctoral thesis comprehensively addresses this challenge by exploring key neural network architectures and proposing a dedicated simplification algorithm for each.

The first objective of this thesis introduced ODF2NNA, an algorithm specifically designed for simplifying dense feed-forward neural networks. Its straightforward definition and ease of use contrasts with its its remarkable versatility, as experimental validation has demonstrated its efficacy across problems involving small, medium, and large datasets and models. This broad applicability underscores ODF2NNA's potential as a foundational tool for optimizing diverse deep learning applications.

The second objective focused on the simplification of convolutional neural networks. For this purpose, OCNNA was proposed, a novel method that ingeniously leverages classical statistical and data analysis tools, including Principal Component Analysis (PCA), the Frobenius matrix norm, and the Coefficient of Variation. By utilizing these robust techniques to quantify the importance of convolutional filters, OCNNA effectively generates simplified versions of widely adopted pre-trained models, validated on benchmark image classification datasets.

Subsequently, the third objective delved into the study and simplification of the Transformer architecture, a pivotal innovation in 21st-century Artificial Intelligence. This architecture, comprising distinct encoder and decoder components, presented unique simplification challenges. For the encoder, PBCE was introduced, a methodology that applies zero-dimensional persistent homology theory to assess the significance of neurons and components within BERT-family models. Extensive evaluation on the widely recognized GLUE Benchmark consistently demonstrated PBCE's ability to produce simplified models without compromising predictive capacity. Addressing the decoder component, a novel simplification method for Mixture-of-Experts Large Language Models (MoE-LLMs) was developed, grounded in information theory. Specifically, normalized mutual information was employed to quantify the importance of individual experts. This approach yielded notably reduced models, which were rigorously evaluated across standard benchmarks for text generation and reasoning tasks.

Finally, the fourth objective of this thesis centered on applying these proposed simplification methods to address complex real-world problems using reduced versions of deep neural networks. Firstly, within the domain of time series forecasting, the GreeNNTSF methodology was presented. This framework, built upon the ODF2NNA pruning algorithm, constructs models that consistently outperform existing state-of-the-art techniques across diverse problems spanning economics, smart cities, biomedicine, and climate science. Secondly, a critical biomedical challenge —the differentiation of glioblastomas using perfusion curves as input data— was tackled. Here, the ODF2NNA and OCNNA algorithms were synergistically applied to simplify complex time series classification networks. This integration resulted in the development of an efficient and effective radiologist decision-support system, further enhanced by the incorporation of explainability elements based on conformal predictors. This final application not only showcases the practical utility of the proposed simplification algorithms but also highlights their potential to contribute meaningfully to real-world clinical contexts.

8.2 Future work

The findings presented in this doctoral thesis lay the groundwork for novel research trajectories, concurrently posing new challenges concerning the simplification and profound understanding of neural networks. This section outlines several prospective research problems that naturally extend from the investigations conducted herein.

- **Simplification of Foundation Models for Time Series-related tasks.** Future work can focus on simplifying foundation models across various time series tasks, including classification, clustering, anomaly detection, and time series forecasting. Given the expanding application of the Transformer architecture beyond natural language processing, particularly within time series analysis, it is proposed that novel strategies be developed for model simplification. These strategies should intrinsically account for the unique characteristics and inherent complexities of time series data. This endeavor aims to yield more efficient models without compromising performance, thereby facilitating their deployment in resource-constrained environments.
- **Compression and Interpretation of Multimodal Models.** The analytical insights gained from scrutinizing the Transformer-decoder simplification method open avenues for deeper investigation into other generative neural networks. Of particular interest is the comprehensive study of multimodal models and how to simplify them. The internal states of neurons and their intermediate outputs within these models often amalgamate concepts spanning disparate domains. Consequently, extracting meaningful patterns from these complex vectorial representations and identifying the most salient computational units presents a vast and largely undefined research domain. This research direction seeks to enhance the interpretability of these sophisticated models.
- **Simplification of Ensemble and Mixture-of-Experts Models.** Further research will explore the simplification of ensemble models and Mixture-of-Experts architectures, which integrate various neural networks. Investigating the learning paradigms and inter-model knowledge transfer within collaborative multi-model systems holds significant promise for simplification. This extends beyond mere parameter reduction to a more abstract level, aiming to eliminate redundancy among models. The objective is to foster efficient and effective collaboration, thereby optimizing the overall model architecture and reducing computational overhead while maintaining or improving predictive power.
- **Application of Model Compression in Robotics.** A crucial future direction involves the practical application of model compression techniques to the field of robotics. This entails the deployment of simplified models onto robotic systems, thereby enabling the full potential of machine learning to be harnessed in environments characterized by limited computational capabilities. This research is critical for developing autonomous robotic systems that are both intelligent and resource-efficient, facilitating their operation in real-world scenarios where computational constraints are prevalent.

Bibliography

[1] Yann LeCun, D Touresky, G Hinton, and T Sejnowski. A theoretical framework for back-propagation. In *Proceedings of the 1988 connectionist models summer school*, volume 1, pages 21–28, 1988.

[2] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, Oct 1986.

[3] Jianchang Mao and A.K. Jain. Artificial neural networks for feature extraction and multivariate data projection. *IEEE Transactions on Neural Networks*, 6(2):296–317, 1995.

[4] Suresh Dara and Priyanka Tumma. Feature extraction by using deep learning: A survey. In *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, pages 1795–1801, 2018.

[5] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[6] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

[7] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[8] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. 25, 2012.

[10] Waseem Rawat and Zenghui Wang. Deep convolutional neural networks for image classification: A comprehensive review. *Neural Computation*, 29(9):2352–2449, 2017.

[11] Haojie Zhao, Gang Yang, Dong Wang, and Huchuan Lu. Deep mutual learning for visual object tracking. *Pattern Recognition*, 112:107796, 2021.

[12] Roberto Olmos, Siham Tabik, and Francisco Herrera. Automatic handgun detection alarm in videos using deep learning. *Neurocomputing*, 275:66–72, 2018.

[13] Farhana Sultana, Abu Sufian, and Paramartha Dutta. A review of object detection models based on convolutional neural network. *Intelligent computing: image processing based applications*, pages 1–16, 2020.

[14] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, 30(11):3212–3232, 2019.

[15] Qian Yu, Yang Gao, Yefeng Zheng, Jianbing Zhu, Yakang Dai, and Yinghuan Shi. Crossover-net: Leveraging vertical-horizontal crossover relation for robust medical image segmentation. *Pattern Recognition*, 113:107756, 2021.

[16] GV Sivanarayana, K Naveen Kumar, Y Srinivas, and GVS Raj Kumar. Review on the methodologies for image segmentation based on cnn. In *Communication Software and Networks: Proceedings of INDIA 2019*, pages 165–175. Springer, 2021.

[17] Lei Cai, Jingyang Gao, and Di Zhao. A review of the application of deep learning in medical image classification and segmentation. *Ann. Transl. Med.*, 8(11):713, June 2020.

[18] Ahmad Waleed Salehi, Shakir Khan, Gaurav Gupta, Bayan Ibrahimm Alabduallah, Abrar Almjally, Hadeel Alsolai, Tamanna Siddiqui, and Adel Mellit. A study of cnn and transfer learning in medical imaging: Advantages, challenges, future scope. *Sustainability*, 15(7):5930, 2023.

[19] J J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982.

[20] Larry R Medsker, Lakhmi Jain, et al. Recurrent neural networks. *Design and Applications*, 5(64-67):2, 2001.

[21] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[22] Sepp Hochreiter. Recurrent neural net learning and vanishing gradient. *International Journal Of Uncertainity, Fuzziness and Knowledge-Based Systems*, 6(2):107–116, 1998.

[23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[24] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[25] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[26] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020.

[27] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations, 2020.

[28] Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, Nathan Cooper, Griffin Adams, Jeremy Howard, and Iacopo Poli. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference, 2024.

[29] Qian Li, Hao Peng, Jianxin Li, Congying Xia, Renyu Yang, Lichao Sun, Philip S Yu, and Lifang He. A survey on text classification: From traditional to deep learning. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 13(2):1–41, 2022.

[30] Alvin Subakti, Hendri Murfi, and Nora Hariadi. The performance of bert as data representation of text clustering. *Journal of big Data*, 9(1):15, 2022.

[31] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, ..., and Dario Amodei. Language models are few-shot learners, 2020.

[32] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, ..., and Brooke Chan. Gpt-4 technical report, 2024.

[33] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, ..., and Hu Xu. The llama 3 herd of models, 2024.

[34] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023.

[35] DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, ..., and Ruizhe Pan. Deepseek-v3 technical report, 2025.

[36] Core Francisco Park, Andrew Lee, Ekdeep Singh Lubana, Yongyi Yang, Maya Okawa, Kento Nishi, Martin Wattenberg, and Hidenori Tanaka. Iclr: In-context learning of representations. In *The Thirteenth International Conference on Learning Representations*, 2025.

[37] Tina Vartziotis, Maximilian Schmidt, George Dasoulas, Ippolyti Dellatolas, Stefano Attademo, Viet Dung Le, Anke Wiechmann, Tim Hoffmann, Michael Keckeisen, and Sotirios Kotsopoulos. Carbon footprint evaluation of code generation through llm as a service. In André Casal Kulzer, Hans-Christian Reuss, and Andreas Wagner, editors, *2024 Stuttgart International Symposium on Automotive and Engine Technology*, pages 230–241, Wiesbaden, 2024. Springer Fachmedien Wiesbaden.

[38] Yi Ding and Tianyao Shi. Sustainable llm serving: Environmental implications, challenges, and opportunities : Invited paper. In *2024 IEEE 15th International Green and Sustainable Computing Conference (IGSC)*, pages 37–38, 2024.

[39] Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. Green ai. *Commun. ACM*, 63(12):54–63, November 2020.

[40] Glenn Shafer and Vladimir Vovk. A tutorial on conformal prediction. *Journal of Machine Learning Research*, 9(3), 2008.

[41] Jake Bouvrie. Notes on convolutional neural networks. 2006.

[42] Krishna Teja Chitty-Venkata, Murali Emani, Venkatram Vishwanath, and Arun K Somani. Neural architecture search for transformers: A survey. *IEEE Access*, 10:108374–108412, 2022.

[43] Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. A survey of transformers. *AI Open*, 3:111–132, 2022.

[44] Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, Quoc V Le, James Laudon, et al. Mixture-of-experts with expert choice routing. *Advances in Neural Information Processing Systems*, 35:7103–7114, 2022.

[45] Omar Sanseviero, Lewis Tunstall, Philipp Schmid, Sourab Mangrulkar, Younes Belkada, and Pedro Cuenca. Mixture of experts explained, 2023.

[46] Sheng Shen, Le Hou, Yanqi Zhou, Nan Du, Shayne Longpre, Jason Wei, Hyung Won Chung, Barret Zoph, William Fedus, Xinyun Chen, Tu Vu, Yuexin Wu, Wuyang Chen, Albert Webson, Yunxuan Li, Vincent Zhao, Hongkun Yu, Kurt Keutzer, Trevor Darrell, and Denny Zhou. Mixture-of-experts meets instruction tuning:a winning combination for large language models, 2023.

[47] BigScience Workshop:, Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, ..., and Francesco De Toni. Bloom: A 176b-parameter open-access multilingual language model, 2023.

[48] Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, ..., and Geoffrey Irving. Scaling language models: Methods, analysis and insights from training gopher, 2022.

[49] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. Opt: Open pre-trained transformer language models, 2022.

[50] Shashikant Ilager, Lukas Florian Briem, and Ivona Brandic. Green-code: Learning to optimize energy efficiency in llm-based code generation, 2025.

[51] Siddharth Samsi, Dan Zhao, Joseph McDonald, Baolin Li, Adam Michaleas, Michael Jones, William Bergeron, Jeremy Kepner, Devesh Tiwari, and Vijay Gadepally. From words to watts: Benchmarking the energy costs of large language model inference. In *2023 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–9, 2023.

[52] David Patterson, Joseph Gonzalez, Urs Hölzle, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David R. So, Maud Texier, and Jeff Dean. The carbon footprint of machine learning training will plateau, then shrink. *Computer*, 55(7):18–28, 2022.

[53] The inference cost of search disruption – large language model cost analysis., 2023.

[54] Alex de Vries. The growing energy footprint of artificial intelligence. *Joule*, 7(10):2191–2194, 2023.

[55] Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis. Explainable ai: A review of machine learning interpretability methods. *Entropy*, 23(1), 2021.

[56] Leilani H. Gilpin, David Bau, Ben Z. Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining explanations: An overview of interpretability of machine learning. pages 80–89, 2018.

[57] José Manuel Benítez, Juan Luis Castro, and Ignacio Requena. Are artificial neural networks black boxes? *IEEE Transactions on neural networks*, 8(5):1156–1164, 1997.

[58] Kenneth O. Stanley, Jeff Clune, Joel Lehman, and Risto Miikkulainen. Designing neural networks through neuroevolution. *Nature Machine Intelligence*, 1(1):24–35, Jan 2019.

[59] Wenxiao Wang, Zhengxu Yu, Cong Fu, Deng Cai, and Xiaofei He. Cop: customized correlation-based filter level pruning method for deep cnn compression. *Neurocomputing*, 464:533–545, 2021.

[60] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21, 2019.

[61] Anthony Sarah, Sharath Nittur Sridhar, Maciej Szankin, and Sairam Sundaresan. Llama-nas: Efficient neural architecture search for large language models. In *European Conference on Computer Vision*, pages 67–74. Springer, 2025.

[62] Tailin Liang, John Glossner, Lei Wang, Shaobo Shi, and Xiaotong Zhang. Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing*, 461:370–403, 2021.

[63] Jiedong Lang, Zhehao Guo, and Shuyu Huang. A comprehensive study on quantization techniques for large language models. In *2024 4th International Conference on Artificial Intelligence, Robotics, and Communication (ICAIRC)*, pages 224–231. IEEE, 2024.

[64] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115, 2023.

[65] Cristian Buciluă, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541, 2006.

[66] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[67] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, 2021.

[68] Yijun Tian, Shichao Pei, Xiangliang Zhang, Chuxu Zhang, and Nitesh V Chawla. Knowledge distillation on graphs: A survey. *ACM Computing Surveys*, 57(8):1–16, 2025.

[69] Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. A survey on knowledge distillation of large language models. *arXiv preprint arXiv:2402.13116*, 2024.

[70] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

[71] Yiwen Guo, Anbang Yao, and Yurong Chen. Dynamic network surgery for efficient dnns. *Advances in neural information processing systems*, 29, 2016.

[72] Franco Manessi, Alessandro Rozza, Simone Bianco, Paolo Napoletano, and Raimondo Schettini. Automated pruning for deep neural network compression. In *2018 24th International conference on pattern recognition (ICPR)*, pages 657–664. IEEE, 2018.

[73] Yajun Liu, Kefeng Fan, and Wenju Zhou. Fpwt: Filter pruning via wavelet transform for cnns. *Neural Networks*, 179:106577, 2024.

[74] Xu Geng, Jinxiong Gao, Yonghui Zhang, and Dingtan Xu. Complex hybrid weighted pruning method for accelerating convolutional neural networks. *Scientific Reports*, 14(1):5570, 2024.

[75] Youzao Lian, Peng Peng, Kai Jiang, and Weisheng Xu. Cross-layer importance evaluation for neural network pruning. *Neural Networks*, 179:106496, 2024.

[76] Shaokun Zhang, Xiawu Zheng, Guilin Li, Chenyi Yang, Yuchao Li, Yan Wang, Fei Chao, Mengdi Wang, Shen Li, and Rongrong Ji. You only compress once: Towards effective and elastic bert compression via exploit–explore stochastic nature gradient. *Neurocomputing*, 599:128140, 2024.

[77] Zhou Zhang, Yang Lu, Tengfei Wang, Xing Wei, and Zhen Wei. Ddk: Dynamic structure pruning based on differentiable search and recursive knowledge distillation for bert. *Neural Networks*, 173:106164, 2024.

[78] Dashun Zheng, Jiaxuan Li, Yunchu Yang, Yapeng Wang, and Patrick Cheong-Iao Pang. Microbert: Distilling moe-based knowledge from bert into a lighter model. *Applied Sciences*, 14(14):6171, 2024.

[79] Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702–21720, 2023.

[80] Max Marion, Ahmet Üstün, Luiza Pozzobon, Alex Wang, Marzieh Fadaee, and Sara Hooker. When less is more: Investigating data pruning for pretraining llms at scale. *arXiv preprint arXiv:2309.04564*, 2023.

[81] Yanyue Xie, Zhi Zhang, Ding Zhou, Cong Xie, Ziang Song, Xin Liu, Yanzhi Wang, Xue Lin, and An Xu. Moe-pruner: Pruning mixture-of-experts large language model using the hints from its router. *arXiv preprint arXiv:2410.12013*, 2024.

[82] Cheng Yang, Yang Sui, Jinqi Xiao, Lingyi Huang, Yu Gong, Yuanlin Duan, Wenqi Jia, Miao Yin, Yu Cheng, and Bo Yuan. Moe-i2: Compressing mixture of experts models through inter-expert pruning and intra-expert low-rank decomposition. *arXiv preprint arXiv:2411.01016*, 2024.

[83] Andrzej Maćkiewicz and Waldemar Ratajczak. Principal components analysis (pca). *Computers & Geosciences*, 19(3):303–342, 1993.

[84] Henry Helson. The spectral theorem. *The Spectral Theorem*, pages 23–41, 2006.

[85] G.H. Golub and C.F. Van Loan. *Matrix Computations*. 3rd edition edition, 1996.

[86] B.S. Everitt and A. Skrondal. *The Cambridge Dictionary of Statistics B.S. Everitt Aut; A. Skrondal Aut*. Cambridge University Press, 2011.

[87] Herbert Edelsbrunner and John Harer. *Computational Topology: An Introduction*. 01 2010.

[88] Felix Hensel, Michael Moor, and Bastian Rieck. A survey of topological machine learning methods. *Frontiers in Artificial Intelligence*, 4, 2021.

[89] Yuriy Mileyko, Sayan Mukherjee, and John Harer. Probability measures on the space of persistence diagrams. *Inverse Problems*, 27(12):124007, nov 2011.

[90] Claude Shannon. The lattice theory of information. *Transactions of the IRE professional Group on Information Theory*, 1(1):105–107, 1953.

[91] Claude E Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

[92] Alexander Strehl and Joydeep Ghosh. Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3:583–617, 01 2002.

[93] legacy-datasets/wikipedia · Datasets at Hugging Face — huggingface.co. `https://huggingface.co/datasets/wikipedia`. [Accessed 08-09-2024].

[94] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. pages 353–355, November 2018.

Chapter II

Publications

1 Optimizing Dense Feed-Forward Neural Networks

- Balderas, L., Lastra, M., & Benítez, J. M. (2023). Optimizing dense feed-forward neural networks. *Neural Networks*, 171, 229-241. https://doi.org/10.1016/j.neunet.2023.12.015
  - Status: **Published**.
  - Impact Factor (JCR 2023): **6**
  - Subject Category: **Computer Science, Artificial Intelligence**
  - Rank: **38/197**
  - Quartile: **Q1**

Optimizing Dense Feed-Forward Neural Networks

Luis Balderas
Department of Computer Science
and Artificial Intelligence
DiCITS, iMUDS, DaSCI
University of Granada, Granada, Spain 18071
luisbalru@ugr.es

Miguel Lastra
Department of Software Engineering
DiCITS, iMUDS, DaSCI
University of Granada, Granada, Spain 18071
mlastral@ugr.es

José M. Benítez
Department of Computer Science
and Artificial Intelligence
DiCITS, iMUDS, DaSCI
University of Granada, Granada, Spain 18071
J.M.Benitez@decsai.ugr.es

Abstract

Deep learning models have been widely used during the last decade due to their outstanding learning and abstraction capacities. However, one of the main challenges any scientist has to face using deep learning models is to establish the network's architecture. Due to this difficulty, data scientists usually build over complex models and, as a result, most of them result computationally intensive and impose a large memory footprint, generating huge costs, contributing to climate change and hindering their use in computational-limited devices. In this paper, we propose a novel dense feed-forward neural network constructing method based on pruning and transfer learning. Its performance has been thoroughly assessed in classification and regression problems. Without any accuracy loss, our approach can compress the number of parameters by more than 70%. Even further, choosing the pruning parameter carefully, most of the refined models outperform original ones. Furthermore, we have verified that our method not only identifies a better network architecture but also facilitates knowledge transfer between the original and refined models. The results obtained show that our constructing method not only helps in the design of more efficient models but also more effective ones.

The code implementing the method will be available at https://github.com/luisbalru/ODF2NNA.

***K*eywords** neural network pruning, dense feed-forward neural network optimization

1 Introduction

Over the last decade, deep neural networks have become the state-of-art technique on several challenging tasks such as computer vision [1], speech recognition [2] or natural language processing [3]. Due to the huge amount of data and hardware development and innovation in the field, larger and deeper learning models have been employed to face more complex problems and learn more elaborated patterns from data. Concretely, dense feed-forward neural networks, which represented the beginning of the era of neural networks, are still an extraordinary and widely used tool in solving machine learning problems on various dataset types not only in the industry but also in academia. More than 58000 articles and more than 51000 patents can be found in Scopus related with this type of neural networks.

One of the main challenges any scientist has to face when using deep learning models is to establish the network's topology and its hyper parameters. Assuming that we have millions of parameters, it can be intricate to design a suitable structure for profitable learning, as we would need to choose the number of layers, the number of units and their distribution. Generally, machine learning engineers come up with enormous neural networks to assure their models are complex enough to find the correct patterns in the learning tasks they are facing. Many problems can be addressed by this methodology and modern hardware developments facilitate it (it is known that when training deep learning neural networks, graphical processing units, GPUs, can be more than two orders of magnitude faster than CPUs). However, this solution might not be the most efficient one, due to the fact that these high capacity networks have significant inference and energetic costs [4]. In 2019, researchers at the University of Massachusetts found that training several large Artificial Intelligence models (including neural architecture search) can emit more than 284019 kilograms of carbon dioxide, in other words, nearly five times the lifetime emissions of the average American car (including the manufacture of the car itself) [5].

There are some ways of designing efficient deep neural networks and generating effective solutions for a given learning task. For example, using memetic algorithms to find a good architecture to fit the task or, provided an architecture, using alternative optimization algorithms to fine-tune the connection weights. However, pruning is one of the most used methods to reduce neural networks complexity. In fact, pruning techniques have been extensively studied for model compression since 1990, when Optimal Brain Damage (OBD) [9] and Optimal Brain Surgery (OBS) [10] where designed. Along the past years, many other approaches have been presented in order to generate more efficient and effective neural networks in all their representations (dense, convolutional or recurrent).

Motivated by that general interest, we have developed a new dense feed-forward neural network constructing method called *Optimizing Dense Feed-Forward Neural Network Algorithm* (**ODF2NNA**), based on pruning and transfer learning which requires minimal tuning. For each supervised learning problem, either classification or regression, we build a model to address the learning task but with a very general topology (the same number of neurons for each layer). Then, inspired by [7], we define our pruning algorithm. This algorithm is designed to find an optimal subnet embedded in the original one, reducing the number of parameters and maintaining, or even improving, the accuracy results. The presented technique has only one parameter ϵ, which represents the pruning level depending on the variability of each unit in a layer. A refining phase is carried out afterwards. Our experiments show that this method is effective: it produces neural networks with improved learning and generalization capabilities through the use of knowledge transfer from large networks to the pruned and refined ones.

The rest of this paper is structured as follows: In Section 2, we introduce the state-of-art of different approaches for designing efficient deep neural networks. In Section 3, we describe our proposal. In Section 4 our methodology is experimentally analyzed In Section 5 we discuss the results and Section 6 highlights the conclusions.

2 Previous work

Designing a well-generalized architecture for deep neural networks is an important task. One approach that must be considered is neuroevolution. Neuroevolution enables important capabilities such as including learning neural network building blocks, hyperparameters, architectures and even the algorithms for learning themselves. Besides, it enables extreme exploration and massive parallelization due to the fact that neuroevolution maintains a population of solutions during the search ([15]). In [24] NEAT algorithm is described as a classical neuroevolution approach. More recently, [25] took inspiration from NEAT and evolved deep neural networks by starting small and adding complexity through mutations. In consequence, entire layers of neurons are added achieving impressive performance on the CIFAR dataset. In [26] we find a variant approach which improved performance by evolving small neural network modules that are repeatedly used in larger hand-coded blueprint, which consists in stacking the same layer module to make a DNN, like Inception, DenseNet and ResNet architectures ([15]). Another approach to be considered is Neural Architecture Search (NAS), which aims to automate the architecture designs of Deep Neural Networks. Mathematically, NAS can be modeled by an optimization problem. Based on the optimizer used, the existing NAS algorithms can be classified into three categories: reinforcement learning based NAS algorithms, gradient-based NAS algorithms and evolutionary NAS algorithms (ENAS). ENAS algorithms are particularly effective and they might be used in many different real-world applications, such as image classification, speech recognition, language modeling or traffic flow forecasting between others. In [27] we find a complete ENAS algorithms survey. In order to enhance the hyper-parameter selection process in Deep Neural Networks, biologically-inspired approaches should be considered. In particular, [28] deploys a Particle Swarm Optimization method as a wrapper to the training process to retrieve hyper-parameters that minimize the classification error. In fact, they present an extensive experimental study involving convolutional neural networks and LeNet-4 network on the MNIST dataset and some very small DNN models optimized using PSO for the CIFAR-10 dataset. Hardware optimizations for accelerating Deep Neural Networks should be considered too ([29])

Although the aforementioned methods show very promising results, they demand heavily computational resources ([30], [31]). Pruning approaches are generally more efficient for deep neural networks and, as a result, they are extensively used for convolutional neural networks and dense feed-forward neural network optimization. More details about CNN pruning and quantization methods can be found in [32]. Nonetheless, as ODF2NNA is designed only for feed-forward networks, CNN optimization methods are not comparable with ODF2NNA, so they are out of the scope of this paper. We will focus on dense feed-forward neural network optimization methods. Sensitivity-based methods ([8]) try to understand how important each weight or node is in the network to prune those which have the least effect on the objective function. The most important pruning algorithms of this kind are OBD and OBS. The basic idea of OBD ([9]) is to use second-derivative information to find a trade-off between network complexity and training set error. The OBD procedure can be carried out as follows: First, choose a network architecture and train it until an accurate solution is obtained. Then, compute the second derivatives and the salience for each parameter. Finally, sort the parameters by salience, delete the lowest ones and start training again. On the other hand, OBS ([10]) trains a neural network minimizing the error. After that, computes the inverse Hessian matrix, in order to find the parameter which gives the smallest salience. If this increment in the candidate error is much smaller than the Tailor's error function (E), then this weight should be deleted and the remaining weights are updated. Otherwise, we may have found that no more weights can be deleted without a large increase in E, so it may be desirable to retrain the network. Consequently, OBD permits pruning more weights than other methods and thus yields better generalization. Dong et al. [11] borrow ideas from OBS and OBD presenting L-OBS (Layer-wise OBS), whose intention is restrict the computation on second order derivative, i.e., the Hessian matrix is only computed for the parameters of an specific layer, making the computations tractable (resp. OBD, where the inverse of the Hessian matrix is calculated over all the parameters). Besides, they reduce computational complexity of the inverse operation of the Hessian matrix by using characteristics of back-propagation for dense layers in deep networks.

LWC [4] begins learning which connections are important via regular network training. Then, they prune the low-weight connections, in other words, connections with weights below a threshold are removed from the network. Finally, the network is retrained to learn the final weights for the remaining connections. Guo et al. [12] propose to sever redundant connections by what they call dynamic network surgery (DNS). This method involves pruning and splicing. The pruning operation is made to compress the network but over pruning or incorrect pruning might be responsible for accuracy loss. In order to compensate it, they incorporate the splicing operation so as to recover connections once the pruned connections are found to be important any time.

Engelbrecht [13] proposed a modified approach to perform sensitivity analysis. Instead of directly using the sensitivity value, Engelbrecht uses the average sensitivity of the network parameters which is computed over all the patterns and then a new measure called variance nullity is applied. As a result, the Variance Nullity Pruning (VNP) method allows pruning both nodes and weights. Hagiwara [14] presented a Magnitude Based Pruning (MBP) method suggesting three strategies called Goodness factor, Consuming energy and Weights power for detecting redundant hidden neurons. Augasta et al. [16] proposed the Neural Network Pruning by Significance (N2PS) pruning method. N2PS is based on a measure which is calculated by the sigmoidal activation value of the node and all the weights of its outgoing connections. Every node whose significance value is below a threshold is considered insignificant and therefore eliminated. Xing et al. [17] proposed a two-phase construction approach for pruning input and hidden units of dense neural networks based on mutual information.

Han et al. [18] presented a three-stage pipeline to reduce the storage required by neural networks. First, they prune the network by removing redundant connections. Next, the weights are quantized so that multiple connections share the same weights. Finally, they apply Huffman coding to take advantage of the biased distribution of effective weights. Manessi et al. [19] introduced the following compressing method: given a neural network N, they build a sibling network N_s that is explicitly able to shrink the trainable weights of N. Then, they train N_s by means of a gradient descent-based technique introducing a regularization term; and finally, they build N_p from N_s, (pruned version of the original network N). Guo et al. [20] proposed a learning automata-based method to train deep neural networks prepared to prune the weakly connected units. For all models, they allocate 1000 units for each layer, and use the ReLU activation function and Gaussian initialization.

Despite the great number of existing research papers focused on Neural Networks pruning, the state of the literature is such that it is not possible yet to answer crucial questions such as 'which techniques achieve the best accuracy/efficiency trade off?' or 'are there strategies that work best on specific architectures or datasets?' The reason is that we are suffering a lack of standardized benchmarks and metrics [21].

3 Proposal

In this paper, we address the challenge of establishing the network's topology. Thus, we introduce Optimizing Dense Feed-Forward Neural Network Algorithm (**ODF2NNA**), a new dense feed-forward neural network construction method based on pruning and transfer learning. In this section we pay special attention to the process that performs the identification and extraction of relevant units (called *useful units*) and finally the construction of the refined model.

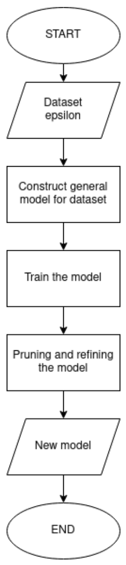

START
Dataset
epsilon
Construct general model for dataset
Train the model
Pruning and refining the model
New model
END

Figure 1: Flowchart of Algorithm 1: Starting with a dataset and a value of ϵ, a general model is constructed, with the same number of neurons per layer, to solve the learning problem associated with the dataset. After training, the model is pruned and refined, resulting in a new one.

3.1 ODF2NNA

ODF2NNA is the neural network construction method we propose. It can be used for classification and regression problems and employs a three-step process: construct a general model, train it and build a new one pruning and refining the general model (Algorithm 1, Figure 1). It receives two arguments: a dataset, which feeds a supervised learning task to address, in our case, building dense feed-forward neural networks, and a pruning tolerance parameter, $\epsilon > 0$.

Algorithm 1 ODF2NNA

```
function ODF2NNA(dataset, $\epsilon$)
    Construct a general model for addressing the dataset learning task.
    Train the general model
    Build a new model by pruning and refining the general model using $\epsilon$.
end function
```

Our procedure begins by receiving a dataset, which represents a classification or regression problem. Once we know the number of examples available in the dataset, we establish the number of parameters as the number of examples and build a general model composed by layers and the same number of neurons per layer. There are no restrictions imposed for the number of layers. The number of parameters of the general model depends on the number of layers chosen. Each problem requires a different number of layers to acquire an accurate model. We have used between three and ten of them.

The number of neurons per layer is determined as follows: For any hidden layer, the number of parameters is equal to the sum of the connections between layers and the bias values in every layer. In consequence, if we define

- NL: number of layers.

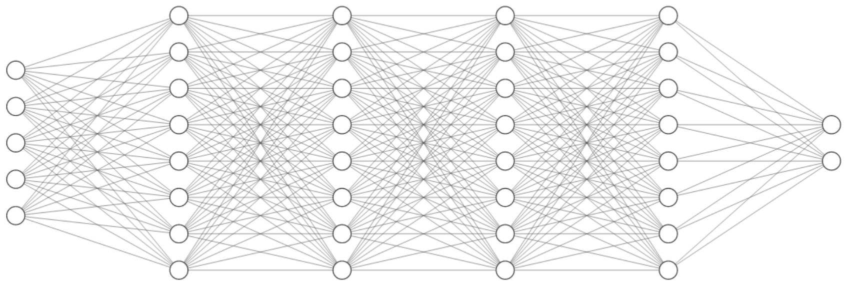


Figure 2: Example of rectangular model

- NP: number of parameters. (1)
- NU: number of units. (2)
- D: input dimension.
- NC: number of classes

We know that

$$NP = D \times NU + NL(NU \times NC) + NL \times NU + NC \tag{1}$$

As a result, we can determine the number of neurons per layer:

$$NU = \frac{NP - NC}{D + NL(NC + 1)} \tag{2}$$

Note that when addressing a regression problem, $NC = 1$, for unidimensional output regression problems. It is straightforward to generalize this for multidimensional regression problems. As a result, we build a rectangular model, in other words, with a given number of layers and the same number of neurons in each layer (Figure 2), whose complexity, in terms of the number of parameters, is enough to successfully complete the learning task.

After the aforementioned model (with NU units per layer) has been trained, based on [7], we apply a pruning method which employs a three-step process (Algorithm 2, Figure 3). It receives two arguments: a dense feed-forward neural network, which has already been trained and whose predictions are accurate for a given problem, and a pruning tolerance parameter, $\epsilon > 0$.

The process starts with the extraction of the *useful units* or neurons from the model, in other words, those units which are indispensable for the network to acquire the real nature and patterns from data.

Algorithm 2 Neural network refining procedure

```
function RefiningNN(model, $\epsilon$)
    Extract the useful units per layer from model applying $\epsilon$.
    Construct the new model with the useful units and layers.
    Model refinement.
end function
```

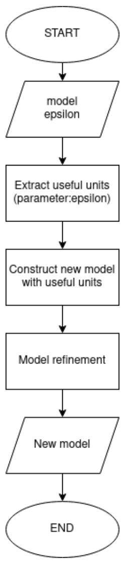


Figure 3: Flowchart of Algorithm 2: Given a model and a value of ϵ, the useful units are extracted and used to build a new model. Subsequently, a refinement process is carried out with slight retraining, resulting in the refined model.

The second step is to construct a new model integrating all the layers, composed only of useful units, to evaluate the prediction capacities of the pruned neural network. The obtained model may under-perform the results compared to the original one. To overcome this problem, the pruned model is lightly retrained using a low number of epochs compared to the number of epochs employed by the training process of the original model. In consequence, the pruned model obtains competitive results and, in some cases, it may even outperform the original model.

3.2 Extracting useful units

The key idea behind the process that performs the extraction of useful units is to find the "important" neurons for the learning task. This process is completed to enhance the model by reducing its size and raising the accuracy level. The resulting model, due to its reduced size, will be more efficient, not only in terms of memory footprint but also runtime.

Algorithm 3 (Figure 5) shows the extraction process if performed layer-wise examining each neuron. The inputs are a model, an evaluation dataset and a pruning tolerance level ϵ.

To better expose the aforementioned algorithm, let us present an example. Let us consider the network shown in Figure 4. For the selected unit, its implicit subnet is bounded by the red pentagon.

Supposing that we are keen on evaluating the relevance of the unit in red, it is necessary to define a strategy to isolate the unit and its implicit subnet, in other words, the neural network formed by every unit belonging to the previous layers and whose output layer is the unit in red.

Once we have a subnet which characterizes a neuron, we need to measure its importance within the whole model. The importance of a subnet is measured by its outputs, which are evaluated for different input examples (*evalData*). If the output of the subnet varies sufficiently for different input examples, then the subnet is considered to contribute valuable insights to the overall information flow across the network. Conversely, if the output of the subnet is uniform for different input examples, then the neuron is deemed to be unimportant for the final prediction of the network. Thus, a metric is needed to encapsulate the variability

Algorithm 3 Extracting useful units

```
function USEFULUNITS(model, evalData, $\epsilon$)
    layers = []
    biasExtra = []
    Append to *layers* the original input layer from *model*
    for each hidden layer in *model*, feed-forward wise, do
        new-layer = []
        bias = 0
        accumulated-bias = 0
        for each unit in layer do
            Create a subnet with $unit$ as its output layer
            result $\leftarrow$ Evaluate subnet on $evalData$
            if Standard-deviation(*result*) $> \epsilon$ then
                Append the unit to *new-layer*
            else
                *bias* = *bias* + mean(*result*)
            end if
        end for
        Append *new-layer* to *layers*
        if *new-layer* is not empty then
            Append *bias*+*accumulated-bias* to *biasExtra*
            *accumulated-bias* = 0
        else
            Append 0 to *biasExtra*
            *accumulated-bias* = *bias*
        end if
    end for
    Append to *layers* the original output layer from *model*
    return *layers*, *biasExtra*
end function
```

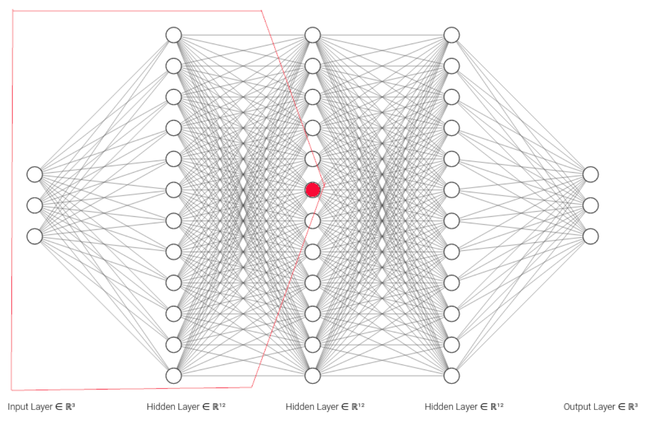


Figure 4: Sample of an implicit subnet for the candidate unit (highlighted in red). This subnet is processed as an independent neural network, and the output of the red neuron is used to evaluate the importance of that unit.

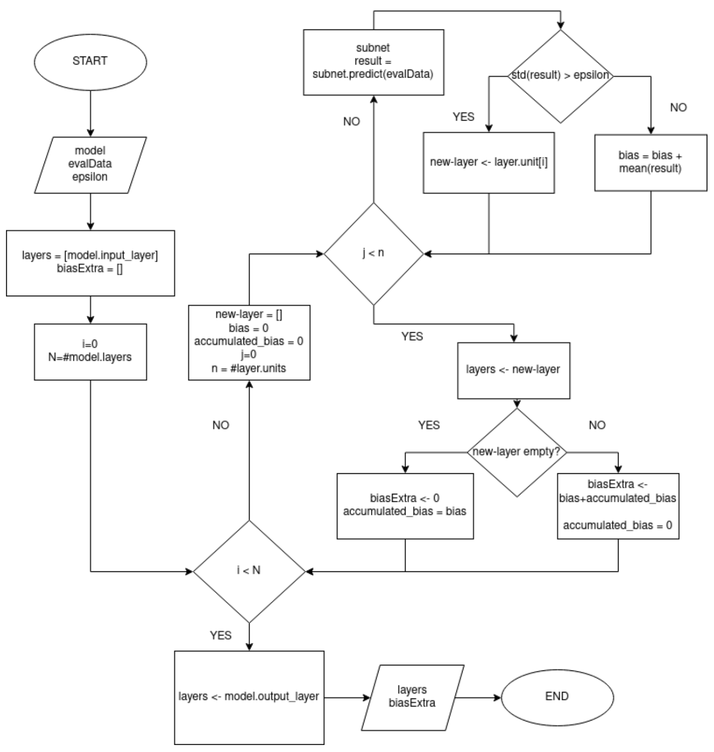


Figure 5: Flowchart of Algorithm 3: Starting with a model, an evaluation dataset, and a value of ϵ, the process to extract useful units from each layer is initiated. This involves comparing the standard deviation of the output of each neuron with ϵ. If it is greater, the neuron is considered useful. If not, it is discarded. To avoid losing all information from the discarded neurons, the mean of the outputs of those neurons is added to the layer's bias.

of the output of a subnet. The standard deviation is chosen as the metric due to its simplicity and widespread use in statistics. The standard deviation over the *evalData* outputs' is measured and if it is greater than ϵ, the unit is classified as significant (useful unit). In consequence, it will form part of the refined model. Otherwise, this unit will not appear in the new model.

However, even if the variation of the predictions in a given subnet might be subtle, it could have an important and global effect on the neural network. Therefore, these results should be taken into consideration. Bearing that in mind, for those units that will be discarded in the new model, we add the mean of their outputs to the layer's bias. As a result, we take into account all the information from the neurons but we only compute those which produce enough variability.

In Algorithm 4 (Figure 6), we can see how a subnet is built using a specific neuron as the only output unit. First, we create a new model and add the input layer from the original model. Then, we add to the new model all the previous layers before the one which contains the neuron being considered for evaluation. Once we find the neuron, we add a layer with this single neuron as the output layer to the subnet. No possible conflict arises in the construction since the *useful units* are selected feed-forward wise.

Each of these sub networks gather a part of the knowledge captured by the initial large network. Actually, a rather relevant part of the overall knowledge since it has been selected as useful. By reusing this part of the network the user is achieving knowledge transfer.

Algorithm 4 Creating a subnet with a specific output final unit

```
function SubNet(model, final-layer, final-unit)
    Create a new model (new-model)
    Add the input layer from the original model
    for each layer in model previous to final-layer do
        Add layer to new-model and set weights to new-model
    end for
    Add a final layer with an unique output unit (final-unit from final-layer) to new-model
    return new-model
end function
```

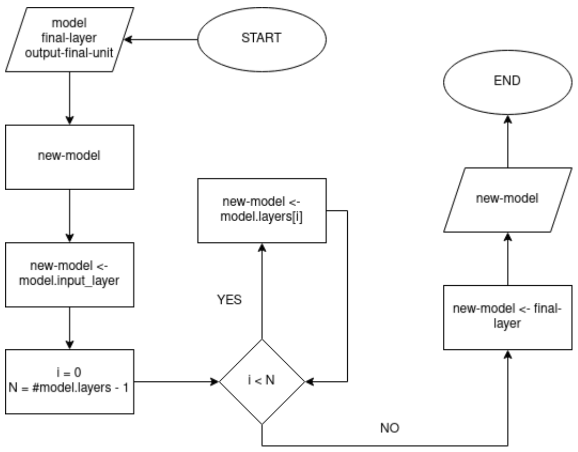


Figure 6: Flowchart of Algorithm 4: Starting with a model, the last layer, and the unit to evaluate, we build a new model that includes all the layers preceding the one that contains the neuron to be evaluated. Finally, a last layer is added, containing only the neuron that needs to be evaluated.

3.3 Building and retraining the refined model

Once we have defined the useful unit list, we build the refined model using those units which are in the list (whose outputs present enough variability) and adding them to the new network respecting the original topology related to the useful units. Careful update of the weights and shape is crucial in this phase, due to the fact that our method may detect the inoperability of a complete layer in the original model and thus, the layer would be completely disregarded in the process of building the refined model. In this case, if the nth layer is discarded, the $(n-1)$th and the $(n+1)$th layers will be connected. Therefore, an adjustment of shapes is needed. The only layers which are kept immutable are the input and the output layers. Hidden layers from the original model may change completely.

However, as we have said above, the refined model may underperform the results compared to the original one. Therefore, a retraining phase (model refinement) is necessary to finish the process. We have tried to retrain as less as possible so as to carry out small adjustments and avoid overfitting. In other words, we set the number of epochs to 15% of the original number of epochs at the beginning of the process. Hyper parameters like activation functions, optimizer or batch size are maintained constant in both learning tasks.

Our method could be at first considered only as an effective way to only find a neural network optimal topology for a specific learning task. Nonetheless, we have found that building a network with the topology identified by our algorithm and then setting random initial weights and training (with any of the currently available learning algorithms) is not enough to reach the same performance as that achieved with the refined model. Therefore, our method is not only guiding us in the selection of the neural network topology but also it is transferring knowledge from the original model to the refined one. In short, our methodology implements a particular kind of transfer learning. Thus, we could see this method simply as a neural network design procedure characterized by transfer learning.

4 Empirical Evaluation

We have thoroughly evaluated our neural network building scheme, considering the two broad classes of problems —classification and regression—, detailing the study throughout diverse categories. ODF2NNA has been compared with the principal state-of-art approaches mentioned in Section II using the datasets proposed in the research papers which presented those techniques. Additionally, two additional standard classification problems (Spambase and Poker) and one large-scale dataset (Hepmass) from the UCI Machine Learning Repository have been utilized to evaluate the performance and quality of our dense neural network optimization algorithm. Finally, we consider three regression datasets (Ailerons, Compactiv and Pole) to evaluate the effectiveness of our method not only in classification but also in regression. In each experiment, several values for ϵ were used to evaluate the learning capacity of the refined model obtained and its effectiveness. We measured the prediction performance with accuracy (ACC) metrics for classification and mean squared error (MSE) for regression. In addition, we registered the number of parameters and the number of layers of the model to assess the efficiency in terms of memory requirements and runtime (the lower the number of parameters, the higher the efficiency). We compared the accuracy (or MSE, respectively) in three different stages: original model, pruned model without retraining and refined model (retrained pruned model). As a result, we are able to understand if the original model was too complex. In other words, if the original model had more parameters than actually needed. Finally, we can notice the pruned model's learning capacity by comparing the accuracy (or MSE) before and after retraining. The process of retraining is very light in terms of epochs (10% - 15% of the original number or epochs) and, thus, we avoid overfitting.

In summary, we compared ODF2NNA with 15 different techniques for pruning dense feed-forward neural networks. Additionally, we extended the experimentation to problems that have not been addressed, to the best of our knowledge, in the literature, such as dividing the experimentation according to the size of the dataset (medium-scale datasets, large-scale datasets) and facing regression problems.

We implemented the proposal using Keras-Tensorflow 2.1. We carried out the experiments on the following pieces of hardware: AMD Ryzen 7 3800X 8-Core Processor, NVIDIA GeForce GTX 1650 SUPER and NVIDIA GeForce RTX 2080 ti.

4.1 Classification experiments

To illustrate the generality of our method we test it on a core set of common benchmark datasets and fully connected network architectures. In all of these experiments, we measure the accuracy (or the classification error rate) before and after applying the optimization method and the complexity reduction in terms of the remaining number of parameters or layers. Nonetheless, we have adapted our metrics (and the way we show them) to the ones used in the state-of-art references to achieve an accurate comparison between ODF2NNA and the state-of-art references.

4.1.1 Experiment 1: Well-known datasets

In this section, the proposed algorithm is evaluated on four well known datasets from UCI Machine Learning Repository and compared to other pruning methods such as VNP [13], Xing-Hu's method [17], MBP [14], OBD [9], OBS [10] and N2PS [16]. Besides, we have expanded our experimentation with two additional datasets from the UCI Machine Learning Repository. The datasets used in this experiment are:

- Iris dataset (iris): 150 instances, four attributes and three categories: setosa, versicolor and virginica.
- Wisconsin-breast-cancer dataset (cancer): It contains 699 instances with 9 real value attributes and two classes.
- Hepatitis domain dataset (hepatitis): 155 instances, 19 attributes. Binary classification.
- Pima Indians Diabetes dataset (diabetes): Binary classification problem with 500 instances and 8 attributes.
- Wholesale customers dataset: 440 instances, 8 attributes. Binary classification.
- Ecoli: 336 instances, 8 attributes. Multi classification.

Table 1 shows the comparison results of our method on the four datasets compared to the other pruning methods:

Table 1: Comparison results of ODF2NNA on iris, cancer, hepatitis and diabetes. NN: Neural Network Architecture. Best accuracy results are highlighted in bold.

Dataset	Original		N2PS		VNP		Xing-Hu		OBD		OBS		MBP		ODF2NNA	
	NN	ACC	NN	ACC	NN	ACC	NN	ACC	NN	ACC	NN	ACC	NN	ACC	NN	ACC
Iris	4-10-3	96	3-3-3	98.67	2-2-3	97.7	3-2-3	**98.67**	4-4-3	98	4-4-3	98	4-4-3	98	3-4-3	98.5
Cancer	9-10-2	95.4	3-2-2	97.1	9-1-2	97.8	3-3-2	96.78	9-8-1	92.5	9-7-1	90	9-7-1	90	5-3-2	**97.86**
Hepatitis	19-25-2	80.2	2-3-2	86.4	4-4-2	83.3	3-8-2	84.62	19-9-1	78.7	19-16-1	73.8	19-18-1	80.3	7-11-2	**87.21**
Diabetes	8-40-2	68.6	5-3-2	70.3	6-8-2	69.1	6-8-2	**74.22**	8-16-1	68.6	8-26-1	65.4	8-26-1	68.9	5-17-2	73.62
Wholesale customers	8-25-2	75.9	-	-	-	-	-	-	-	-	-	-	-	-	9-12-2	**82.62**
Ecoli	8-22-8	87.9	-	-	-	-	-	-	-	-	-	-	-	-	6-10-8	**94.63**

As we can see, ODF2NNA outperforms the accuracy results for Cancer and Hepatitis compared to the other state-of-art methods reducing the architecture widely.

Table 2 shows the F1-Score obtained for Iris, Wisconsin-breast-cancer, Hepatitis, Diabetes, Wholesale customers and Ecoli datasets. We cannot compare F1-Score with the other state-of-art methods because it is not included in their results.

Table 2: F1-Score value

Dataset	Iris	Cancer	Hepatitis	Diabetes	Wholesale	Ecoli
F1-Score	0.98	0.86	0.79	0.66	0.81	0.92

4.1.2 Experiment 2: Modifying the number of layers

We tested our method on deep neural networks with different numbers of hidden layers (from 2 to 6) on the MNIST dataset. The MNIST dataset consists of 28×28 pixel handwritten digit images from 0 to 9. The task is to classify the images into ten digit classes. There are 60 000 examples in the training set and 10 000 in the test set. For all models, we allocate 1000 units and compare the results with the Learning Automata-Based SGD [20]. Table 3 shows the classification error percentage with different number of hidden layers (columns 2, 3 and 4). The Sparseness column denotes the rates of remaining weights after pruning with respect to the

Table 3: Classification error and sparseness. Best results are highlighted in bold.

Hidden layers	SGD	SGD+LA	ODF2NNA	Sparseness - Guo et al.	ODF2NNA sparseness	ODF2NNA F1-Score
1000×2	1.45	1.43	1.87	43%	**5%**	0.79
1000×3	1.56	1.52	**1.49**	62%	**32%**	0.94
1000×4	1.61	1.49	1.51	60%	**27%**	0.81
1000×5	1.79	1.47	1.68	44%	**13%**	0.84
1000×6	2.4	1.58	**1.52**	54%	**19.8%**	0.87

initial number of weights (less is better). F1-Score value is also added. As mentioned before, we cannot compare F1-Score with the other state-of-art methods because it is not included in their investigation results.

The least error measure values obtained are for 1000×3 and 1000×6 hidden layers. Besides, we are able to reduce the sparseness in each architecture compared to Guo et al. [20].

4.1.3 Experiment 3: LeNet300-100

In this experiment we verify the effectiveness of our optimization method on MNIST using a deep neural network architecture called LeNet-300-100, which is a fully connected neural network made of four layers: the input layer, the output layer and two hidden layers with 300 and 100 neurons respectively.

First of all, we compare our method with Han et al. [18] and Manessi et al. [19]. Table 4 shows the percentage prediction error (Δ_1) for the pruned networks and their respective non-pruned implementation. We also show the pruning performance, measured by the number of weights' compression. Similarly, Han et al. [18] and Manessi et al. [19] results are considered. Note that there are N/A values, which mean that the original research paper did not specified any value for this item.

Table 4: LeNet-300-100 (I) Comparison with [18] and [19]. Pr. Performance refers to Pruning Performance. Best results are highlighted in bold.

Method	Δ_1	Weights	Pr. Performance
LeNet-300-100 reference	N/A	266K	N/A
LeNet-300-100 pruned (Han et al.)	0.1%	21K	12x
LeNet-300-100 pruned (Manessi et al.)	0.1%	14K	19x
LeNet-300-100 pruned (ODF2NNA)	0.2%	**5K**	**40x**

As we can see, the number of weights is widely reduced, improving the pruning performance.

Finally, we compare our method with randomly pruning [11], Layer-wise OBS [11], OBD [9], LWC [22] and DNS [12]. In this case, we contemplate the original classification error, the error rate after pruning, the error rate after retraining (Re-Error), the number of iterations to complete the retraining phase (#Re-Iters) and the compression ratio (CR), which is the ratio of the number of preserved parameters to that of original parameters (lower is better) [11]. Table 5 shows the obtained results:

Even if ODF2NNA's compression rate is higher than L-OBS (iterative) and DNS, the error rate after the retraining phase is lower. In addition, a significant difference can be observed in the number of epochs required by ODF2NNA to fine-tune the pruned optimized network (50) compared to its closest competitor in error rate after retraining, L-OBS (iterative), which required 643 retraining epochs. This indicates not only that ODF2NNA is a more effective method for optimizing the original network but also that it is much more efficient in terms of runtime and energy consumption. Additionally, F1-Score obtained is 0.77. Again, we cannot compare it with the other state-of-art methods because it is not included in their investigation results.

Table 5: LeNet-300-100 (II). Comparison with [11], [9], [22] and [12]. Best results are highlighted in bold.

Method	Original Error	CR	Err. After Pruning	Re-Error	#Re-Iters
Random	1.76%	8%	85.72%	2.25%	3.5e5
OBD	2.76%	8%	86.72%	1.96%	8.1e4
LWC	3.76%	8%	81.32%	1.95%	1.4e5
DNS	4.76%	1.8%	N/A	1.99%	3.4e4
L-OBS	5.76%	7%	3.1%	1.82%	510
L-OBS (iterative)	6.76%	1.55%	2.43%	1.96%	643
ODF2NNA	2%	2.46%	92%	**1.78%**	**50**

4.1.4 Experiment 4: Other UCI datasets

In order to complete our empirical evaluation, we chose two extra datasets from the UCI Machine Learning Repository which are more complex than the datasets used in Experiment 1. These datasets are Spambase (4597 examples, 57 real attributes, and binary classification) and Poker (1025010, 10 integer attributes, multi classification). To the best of our knowledge, these datasets have not been used to evaluate dense neural network optimization algorithms. Table 6 (Spambase) and Table 7 (Poker) show the original accuracy rate (OC), the accuracy rate after pruning without retraining (PWR) and the accuracy rate after retraining (PR). Each table shows the original number of parameters (OR), the number of preserved parameters after pruning (P) and the reduction percentage.

Table 6: Spambase dataset results (OR = original accuracy rate, PWR = Pruning without retraining accuracy rate, PR = Pruned and retrained accuracy rate). For the number or parameters, OR refers to the original number of parameters, P refers to the number of parameters of the pruned model (the lower the better) and % Reduc to the reduction of the number of parameters (the larger the better)

ϵ	**ACCURACY**			**N. PARAM**		
	OR	PWR	PR	OR	P	% Reduc
0.1	75%	76%	91%	1262	993	21.32%
0.3	75%	77%	93%	1262	861	31.77%
0.5	75%	39%	93%	1262	836	33.76%
0.7	75%	39%	93%	1262	824	34.71%
0.9	75%	39%	92%	1262	810	35.82%

It should be remarked that the pruned and retrained model produces more accurate predictions than the original model in every case. Besides, the reduction in the number of parameters is enormous for the Poker dataset (99% of reduction). The best Spambase F1-Score obtained is 0.89 ($\epsilon = 0.7$) and the best Poker F1-Score obtained is 0.65 ($\epsilon = 0.5$).

4.1.5 Experiment 5: Large-scale dataset

The experiments above show that our method can be successfully applied to small and medium-scale datasets and deep neural architectures. After thoroughly reviewing the literature, we have not found any fully connected neural network optimization method evaluated on large-scale datasets. We have selected Hepmass from the UCI Machine Learning Repository to verify that ODF2NNA is also effective for large-scale datasets (10,500,000 instances, 28 attributes).

Table 7: Poker dataset results (OR = original accuracy rate, PWR = Pruning without retraining accuracy rate, PR = Pruned and retrained accuracy rate). For the number or parameters, OR refers to the original number of parameters, P refers to the number of parameters of the pruned model (the lower the better) and % Reduc to the reduction of the number of parameters (the larger the better)

ϵ	**ACCURACY**			**N. PARAM**		
	OR	PWR	PR	OR	P	% Reduc
0.1	53%	50%	59%	374261	137141	63.36%
0.3	53%	42%	66%	374261	55791	85.09%
0.5	53%	0%	74%	374261	21776	94.18%
0.7	53%	0%	41%	374261	9477	97.47%
0.9	53%	0%	86%	374261	3837	98.97%

Table 8: Hepmass dataset results. The number of training epochs for the original model is set to 1000. In contrast, for the different values of ϵ, the number of epochs in the refinement phase is 100 (10% of the total).

ϵ	#Layers	#Parameters	Accuracy	%Reduction
Original	17	3773002	81%	0%
5e-5	17	2287537	82%	39.37%
1e-5	17	2283005	**85.6%**	**39.49%**
0.0001	17	2224246	83.49%	41.05%
0.0004	17	2114143	80.35%	43.96%
0.01	17	1274437	76.47%	66.22%
0.04	13	597337	78.51%	84.17%
0.2	10	396290	56.21%	89.5%
0.8	8	257858	53.9%	93.17%
1	7	230417	50%	93.89%

As we can see, the value $\epsilon = 1e - 5$ produces the best results, including F1-Score (0.72).

In addition, we have selected Higgs from the UCI Machine Learning Repository as another example of learning task based on a large dataset (11,000,000 instances, 28 attributes). This dataset size is comparable in terms of instance number to the widely considered Imagenet benchmark. The Higgs dataset was used in [33], as a semi-realistic case, for showing that deep learning can improve the power of collider searches for exotic particles, concluding that deep-learning techniques discover powerful nonlinear feature combinations and provide better discrimination power than other classifiers.

In [33] a feed-forward network architecture is presented, formed by five layers with 300 hidden units in each layer, a learning rate of 0.05 and a weight decay of $1e - 5$, obtaining an area under the curve (AUC) value of 0.885. We have replicated this architecture and conditions to apply ODF2NNA, with a retraining phase of 50 epochs.

ODF2NNA is able to reduce by more than 23% the number of parameters and increases in a 5% the accuracy compared to the original model. As we can see in Figure 7, the AUC for the best ϵ is 0.93, compared to the 0.885 reported in [33]. Again, it is clear that ODF2NNA is an accurate optimization method for learning tasks based on large datasets.

Table 9: The Higgs dataset results

ϵ	#Layers	#Parameters	#Epochs train/retrain	Accuracy	%Reduction
Original	5	370502	1000	88%	0%
0.0001	5	283895	50	91.3%	23.38%
0.0002	17	282055	50	**93%**	**23.87%**
0.0003	17	280309	100	89%	24.34%
0.0004	17	247669	100	83.6%	33.15%
0.001	13	192351	100	76.13%	48.08%
0.01	10	95363	100	69.87%	74.26%

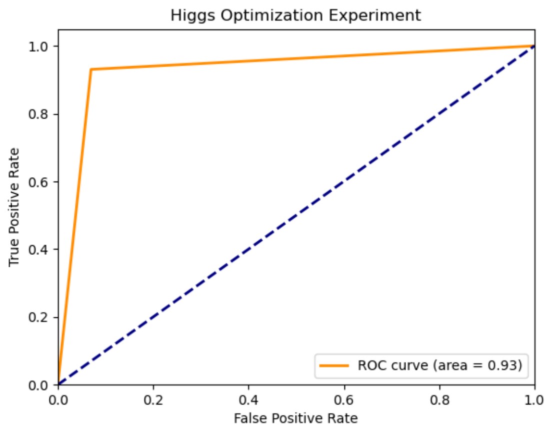


Figure 7: ROC Curve for the Higgs Dataset optimization $\epsilon = 0.0002$

4.2 Regression

We evaluated ODF2NNA not only on classification problems but also on regression ones. For this reason, we chose three regression datasets from UCI Machine Learning Repository: Ailerons, Compactive and Pole. The result tables of this section present the values of Mean Squared Error (MSE) and the number of parameters (N.PARAM) for each dataset.

Table 10: Ailerons dataset results (OR = original, PWR = Pruning without retraining, PR = Pruned and retrained, P = pruned)

ϵ	**MSE**			**N. PARAM**		
	OR	PWR	PR	OR	P	% Reduc
0.002	3.07e-6	4.06e-5	3.28e-6	2809	1148	59.13%
0.008	3.07e-6	0.0009	1.42e-5	2809	771	72.55%
0.014	3.07e-6	6.68e-5	2.64e-6	2809	757	73.05%
0.020	3.07e-6	2.66e-5	2.74e-6	2809	757	73.05%
0.026	3.07e-6	2.65e-5	3.04e-6	2809	757	73.05%

It should be remarked that a high reduction in the number of parameters is achieved without a relevant increase in the error measure.

Table 11: Compactiv dataset results (OR = original, PWR = Pruning without retraining, PR = Pruned and retrained, P = pruned)

ϵ	**MSE**			**N. PARAM**		
	OR	PWR	PR	OR	P	% Reduc
0.002	2.63e-7	2.19e-6	1.24e-6	1958	1279	34.68%
0.008	2.63e-7	1.14e-5	4.81e-6	1958	513	73.8%
0.014	2.63e-7	3.86e-5	5.55e-7	1958	438	77.63%
0.020	2.63e-7	3.86e-5	6.38e-7	1958	438	77.63%
0.026	2.63e-7	3.86e-5	8.61e-7	1958	438	77.63%

Table 12: Pole dataset results (OR = original, PWR = Pruning without retraining, PR = Pruned and retrained, P = pruned)

ϵ	**MSE**			**N. PARAM**		
	OR	PWR	PR	OR	P	% Reduc
0.002	6.35e-5	7.43e-5	1.76e-6	4159	973	76.6%
0.008	6.35e-5	5.64e-5	5.66e-6	4159	815	80.4%
0.014	6.35e-5	5.94e-5	7.05e-6	4159	650	84.37%
0.020	6.35e-5	5.72e-5	7.35e-6	4159	617	85.16%
0.026	6.35e-5	7.32e-5	6.87e-6	4159	540	87.02%

5 Empirical result analysis and Discussion

We start this section analyzing the classification results. The classification experiments are divided into five sections, where the size of the datasets is varied to verify the robustness of ODF2NNA when facing different learning tasks. In total, the results of ODF2NNA are compared with 13 techniques from the state of the art. The following presents the different proposed experiments.

Experiment 1 was designed to evaluate ODF2NNA on small-scale datasets. We compared its performance to other pruning methods which were built to face this kind of learning tasks. We found that ODF2NNA outperforms all the other approaches, in terms of neural network complexity (NN) and accuracy for the Cancer dataset. For Hepatitis and Diabetes, we obtained the highest accuracy after optimizing the model.

Experiment 2 measured the classification error after retraining and the optimized model sparseness, in other words, the rates of remaining weights after pruning. Using the same number of hidden layers (from 2 to 6) as the original work [23], ODF2NNA obtained very similar classification error rates while generating a model whose complexity in terms of the number of parameters was reduced by more than 60% compared to [23]. In fact, for two hidden layers, the rate of remaining weights after pruning dropped to one tenth (43% vs 5%); for three and four hidden layers, around half as much (62%-32%, 60%-27%); for five hidden layers, it dropped to three tenth (44%-13%); and for six hidden layers, the number of remaining weights dropped to one third (54%-19.8%). In consequence, in this test ODF2NNA showed an outstanding capacity of reducing deep learning models and robustness even when the neural network architecture is heavily modified. Experiment 3 endorsed this statement too, since we managed to reduce the LeNet 300-100's number of parameters from 266000 to 5000, in other words, the amount of parameters dropped by 98% without worsening the performance in prediction. Experiment 3 also showed that ODF2NNA is effective not only finding a new

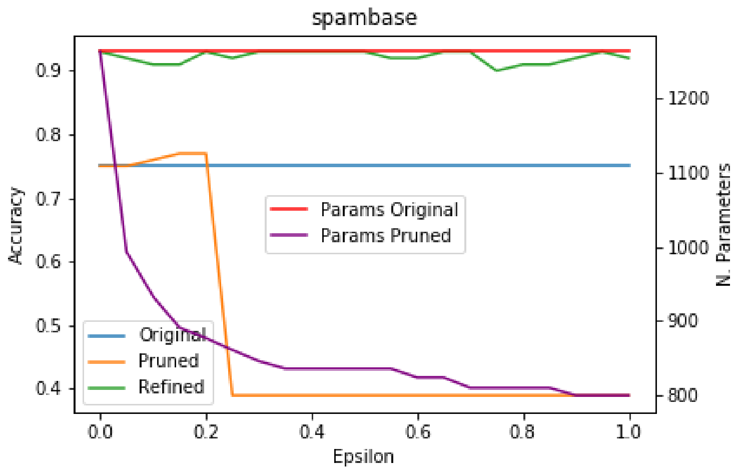


Figure 8: Experiment on the Spambase dataset. On the left Y-axis, Accuracy is expressed, and on the right Y-axis, the number of model parameters is shown. It can be observed that for all represented values of ϵ, the accuracy of the refined algorithm (in green) is higher than the original model (in blue). It is crucial to refine the model since pruning leads to a sharp decline in accuracy values for the pruned model (in orange). Furthermore, it is evident that a significant simplification in terms of the number of parameters can be achieved (in red for the original values vs. pruned parameter count in purple).

optimized model from an original one given a learning task, but also highly efficient due to the low amount of iterations to complete the retraining phase (#Re-Iters in Table 5) compared to other approaches.

Experiment 4 showed that Spambase (Figure 8) and Poker (Figure 9) are two examples which illustrate how effective pruning and retraining a model can be, increasing by more than 15%, in the case of Spambase, and by more than 40% for certain values of ϵ, in the case of Poker, the accuracy with respect to the original model. Besides, we reduced the number of parameters by more than 35% for Spambase. As a result, we obtained, for all $\epsilon \in (0, 1]$, a much more effective and efficient refined model to solve the Spambase learning task compared to the original model. For Poker, ODF2NNA reduced the number of parameters by more than 90%. Nonetheless, for $\epsilon = 0.7$, the refined model underperforms the original one, in other words, there are some ϵ values which generate a refined model less accurate than the original model. Therefore, it is clear that ϵ ought to be carefully chosen.

We finished the classification phase with Experiment 5, in which we faced two large-scale learning tasks: Hepmass and Higgs classification problems. In the case of Hepmass, the original model, which had more than 3.7 million of parameters, obtained an 81% of accuracy in prediction. We found that, for $\epsilon = 1e - 5$, the number of parameters was reduced by almost 40% (2.28 million of parameters) and the accuracy was increased by more than 4.5% (85.6% in test). For the Higgs dataset, we used the architecture proposed in [33] in order to compare the ODF2NNA results and the original ones. A 93% of accuracy in prediction, reducing in more than 23% the number of parameters, and a 0.93 in AUC is obtained, compared to the 0.885 that can be found in the original paper. In other words, ODF2NNA is effective in large-scale learning tasks too.

On the other hand, regression results are similar to classification ones. We can notice the importance of retraining the model again, obtaining in almost every case smaller mean square error values if a light retraining is applied.

The complexity of the original models for regression problems is reasonable (they are not oversized), taking into account their acceptable original performance in contrast to the poor MSE results obtained after the

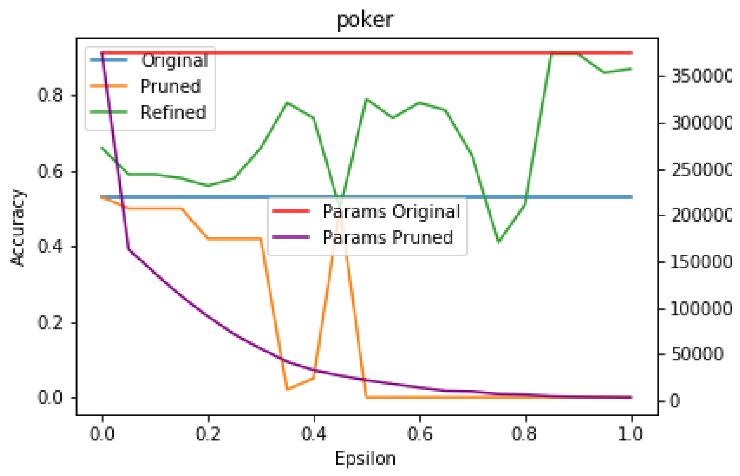


Figure 9: Experiment on the Poker dataset. On the left Y-axis, Accuracy is expressed, and on the right Y-axis, the number of model parameters is shown. Fluctuations in the accuracy of the refined model can be observed, although for most values of ϵ, the accuracy is higher than that of the original model. Once again, this underscores the importance of refining models after pruning.

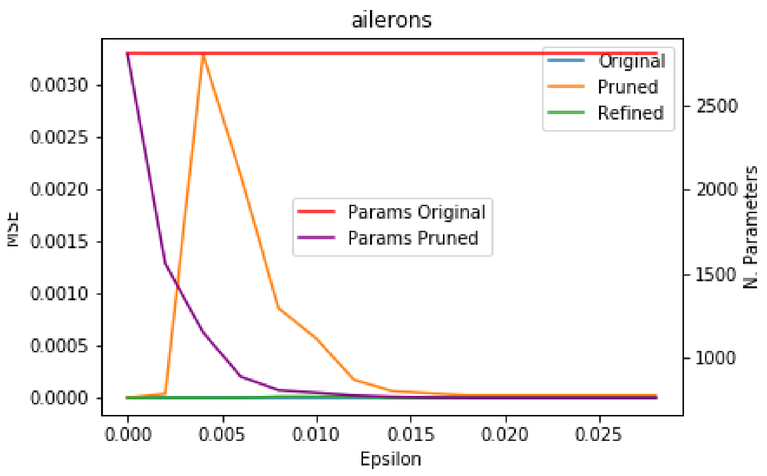


Figure 10: Experiment on the Ailerons dataset. On the left Y-axis, Mean Squared Error (MSE) is expressed, and on the right Y-axis, the number of model parameters is shown. As you can see, the refined model achieves the same MSE as the original one, while inducing a significant reduction in the number of parameters.

simplification procedure without retraining. This can be considered in the classification examples given above, too. We found many examples where our construction method generates models with less than 70% of the original number of parameters without affecting their performance in prediction. That is the case of Ailerons (Figure 10), House (Figure 11) and Compactiv (Figure 12).

Again, we emphasize the necessity of carefully choosing the value of ϵ. In our sensitivity study we can see again that, for some values of the pruning parameter, it is possible not only to maintain the original MSE in the refined model but also to improve it. That is the case of $\epsilon = 0.01$ in Pole (MSE value of $7{\cdot}10^{-6}$ versus $6.35{\cdot}10^{-5}$ and 80% of reduction of the number of parameters) (Figure 13).

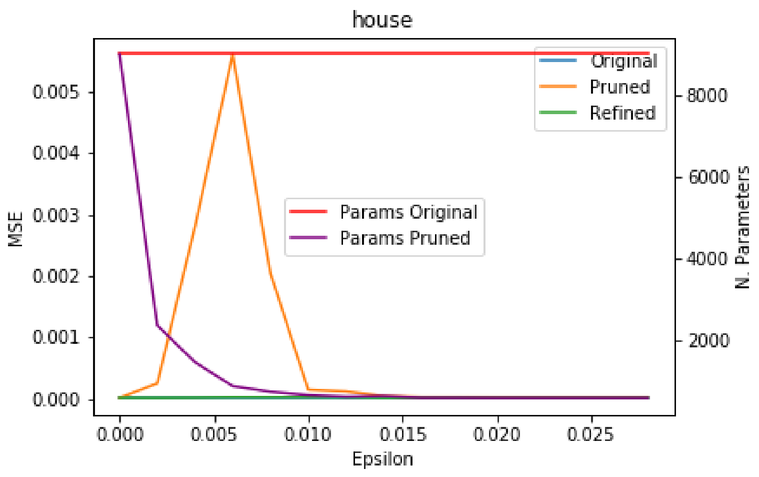


Figure 11: Experiment on the House dataset. On the left Y-axis, Mean Squared Error (MSE) is expressed, and on the right Y-axis, the number of model parameters is shown. As in the Ailerons dataset case, the refined model achieves the same MSE as the original one, while inducing a significant reduction in the number of parameters.

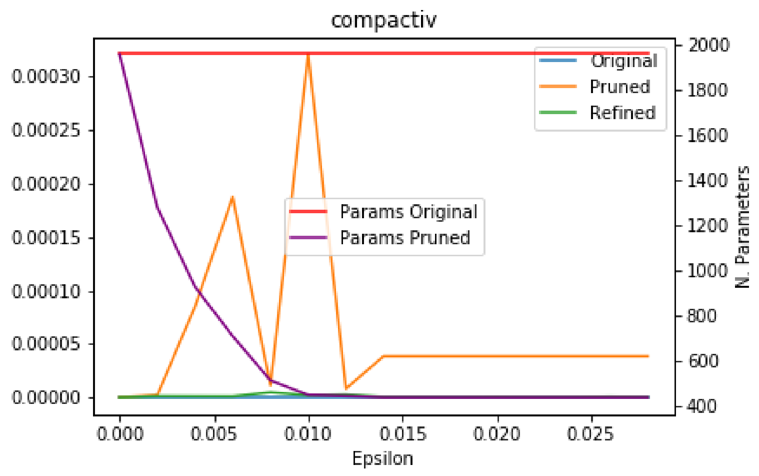


Figure 12: Experiment on the Compactiv dataset. On the left Y-axis, Mean Squared Error (MSE) is expressed, and on the right Y-axis, the number of model parameters is shown. Unlike the previous cases, for the Compactiv dataset, there is clear evidence of the need to refine the simplified models. The reduction in parameters from the original model to the refined one is quite remarkable.

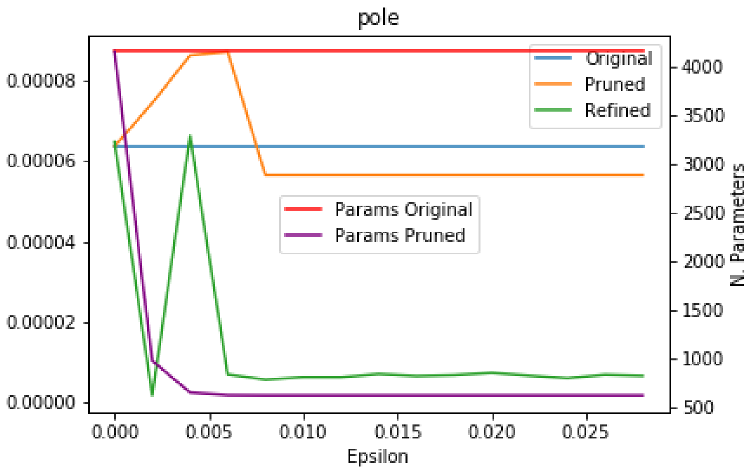


Figure 13: Experiment on the Pole dataset. On the left Y-axis, Mean Squared Error (MSE) is expressed, and on the right Y-axis, the number of model parameters is shown. In this case, a lower MSE is achieved compared to the original model for most values of ϵ, significantly reducing the number of parameters.

Finally, it could seem reasonable to think that the refining method could be used only to find the optimal topology, namely, the number of layers and unit distribution over them (most challenging parameters for the data scientist), and that a model with this topology built from scratch would reach the same performance as the refined model. However, our experiments show that this is not true at least with the currently available training algorithms.

We have selected Spambase (classification) and Pole (regression), two datasets where, for certain values of ϵ, the refined method outperforms the original one. In the case of Spambase, the optimal number of parameters after pruning is around 800 (35.82% reduction compared to the original model). We have built a neural network with 800 parameters, Adam optimizer and 240 epochs and early stopping with patience of 20 (the same hyper parameters as the original model). We obtained 75% of accuracy in prediction, which is smaller than the refined one (92%). On the other hand, for Pole, we built a model with 815 parameters (the optimal number of parameters as we can see in the above section), Adam optimizer and 1000 epochs and early stopping with patience of 50 (same parameters as the original model) obtaining a mean squared error of $6.35{\cdot}10^{-5}$, more than ten times greater than the MSE attained with the refined model ($5.66{\cdot}10^{-6}$). Similar results have been obtained on other classification and regression problems.

As a result, our refined method is not only guiding us in the construction of more efficient models but also it is generating more accurate neural networks compared to others conceived from the beginning. The high performance of this –longer path– approach may be supported by the fact that effective transfer knowledge is being achieved by composing a final model from the most meaningful components of a larger and well-performing initial model.

In summary, based on the empirical results, we believe that optimizers derived from back propagation are not powerful enough to find an optimal function given a neural network topology (which represents a functional subspace). In consequence, methods optimizing dense feed-forward neural networks, like the one we have proposed, might be the best way to lead the data scientist to design efficient and effective deep neural networks for a learning task.

6 Conclusion

Deep learning models have been widely used during the last decade due to their learning and generalization capacities. However, these neural networks entail significant energetic costs and are hard to design efficiently. In this paper, we propose ODF2NNA, a new building method for dense feed-forward neural networks based on a three-step process: constructing a general model to address the learning task, train it properly and refine the model by finding the useful units per layer, removing the irrelevant ones (according to a parameter ϵ) and finally, lightly retraining the new model. The proposal has been evaluated through a thorough empirical study on both classification and regression problems, including small, medium and large-scale datasets and learning tasks and comparing with 15 different techniques for pruning dense feed-forward neural networks. The experimental results confirm that simpler and more accurate models can be obtained. So the proposed method does not only allow to define effective topologies but drives to build well-performing neural networks based on knowledge transfer from relevant subnetworks.

Acknowledgment

This research has been partially supported by the Spanish Ministry of Economy, Industry and Competitiveness (TIN2016-81113-R and PID2020-118224RB-I00) and the Industry Andalusian Council (Consejería de Transformación Económica, Industria, Conocimiento y Universidades de la Junta de Andalucía, P18-TP-5168), with the cofinance of the European Union (FEDER).

References

[1] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, Commun. ACM 60 (6) (2017) 84–90. doi:10.1145/3065386.
URL https://doi.org/10.1145/3065386

[2] A. Graves, J. Schmidhuber, Framewise phoneme classification with bidirectional lstm and other neural network architectures, Neural Networks 18 (5) (2005) 602 – 610, iJCNN 2005. doi:https://doi.org/10.1016/j.neunet.2005.06.042.
URL http://www.sciencedirect.com/science/article/pii/S0893608005001206

[3] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, P. Kuksa, Natural language processing (almost) from scratch (2011). arXiv:1103.0398.

[4] S. Han, J. Pool, J. Tran, W. J. Dally, Learning both weights and connections for efficient neural networks (2015). arXiv:1506.02626.

[5] E. Strubel, A. Ganesh, A. McCallum, Energy and policy considerations for deep learning in nlp, Association for Computational Linguistics (2019) 3645–3650.

[6] Passalis, N., Tzelepi, M. & Tefas, A. Chapter 8 - Knowledge distillation. *Deep Learning For Robot Perception And Cognition*. pp. 165-186 (2022), https://www.sciencedirect.com/science/article/pii/B9780323857871000130

[7] J. Siestma, R. J. Dow, Creating artificial neural networks that generalize 4 (1991) 67–79.

[8] S. J. W. Y. e. a. Zeng, X., A sensitivity-based approach for pruning architecture of madalines, in: Neural Comput and Applic, Vol. 18, 2009.

[9] Y. LeCun, J. S. Denker, S. A. Solla, Optimal brain damage, in: D. S. Touretzky (Ed.), Advances in Neural Information Processing Systems 2, Morgan-Kaufmann, 1990, pp. 598–605.
URL http://papers.nips.cc/paper/250-optimal-brain-damage.pdf

[10] B. Hassibi, D. G. Stork, G. J. Wolff, Optimal brain surgeon and general network pruning, in: IEEE International Conference on Neural Networks, 1993, pp. 293–299 vol.1.

[11] X. Dong, S. Chen, S. J. Pan, Learning to prune deep neural networks via layer-wise optimal brain surgeon (2017). arXiv:1705.07565.

[12] Y. Guo, A. Yao, Y. Chen, Dynamic network surgery for efficient dnns (2016). arXiv:1608.04493.

[13] A. P. Engelbrecht, A new pruning heuristic based on variance analysis of sensitivity information, Trans. Neur. Netw. 12 (6) (2001) 1386–1399. doi:10.1109/72.963775.
URL https://doi.org/10.1109/72.963775

[14] M. Hagiwara, A simple and effective method for removal of hidden units and weights, Neurocomputing 6 (2) (1994) 207–218, backpropagation, Part IV. doi:https://doi.org/10.1016/0925-2312(94)90055-8.
URL https://www.sciencedirect.com/science/article/pii/0925231294900558

[15] K. Stanley, J. Clune, J. Lehman, R. Miikkulainen, Designing neural networks through neuroevolution, Nature Machine Intelligence 1 (1) (2019) 24-35 doi:10.1038/s42256-018-0006-z.
URL https://www.nature.com/articles/s42256-018-0006-z/

[16] M. Augasta, T. Kathirvalavakumar, A novel pruning algorithm for optimizing feedforward neural network of classification problems, Neural Processing Letters 34 (2011) 241–258.

[17] H.-J. Xing, B.-G. Hu, Two-phase construction of multilayer perceptrons using information theory, IEEE Transactions on Neural Networks 20 (4) (2009) 715–721. doi:10.1109/TNN.2008.2005604.

[18] S. Han, H. Mao, W. J. Dally, Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding (2016). arXiv:1510.00149.

[19] F. Manessi, A. Rozza, S. Bianco, P. Napoletano, R. Schettini, Automated pruning for deep neural network compression, in: 2018 24th International Conference on Pattern Recognition (ICPR), 2018, pp. 657–664. doi:10.1109/ICPR.2018.8546129.

[20] H. Guo, S. Li, B. Li, Y. Ma, X. Ren, A new learning automata-based pruning method to train deep neural networks, IEEE Internet of Things Journal 5 (5) (2018) 3263–3269. doi:10.1109/JIOT.2017.2711426.

[21] D. Blalock, J. J. G. Ortiz, J. Frankle, J. Guttag, What is the state of neural network pruning? (2020). arXiv:2003.03033.

[22] S. Han, J. Pool, J. Tran, W. J. Dally, Learning both weights and connections for efficient neural networks (2015). arXiv:1506.02626.

[23] Y. Guo, A. Yao, Y. Chen, Dynamic network surgery for efficient dnns (2016). arXiv:1608.04493.

[24] Stanley, K. & Miikkulainen, R. Evolving Neural Networks Through Augmenting Topologies. *Evolutionary Computation*. **10**, 99-127 (2002), http://nn.cs.utexas.edu/?stanley:ec02

[25] Real, E., Moore, S., Selle, A., Saxena, S., Suematsu, Y., Tan, J., Le, Q. & Kurakin, A. Large-Scale Evolution of Image Classifiers. *Proceedings Of The 34th International Conference On Machine Learning*. **70** pp. 2902-2911 (2017,8,6), https://proceedings.mlr.press/v70/real17a.html

[26] Real, E., Aggarwal, A., Huang, Y. & Le, Q. Regularized Evolution for Image Classifier Architecture Search. *Proceedings Of The AAAI Conference On Artificial Intelligence*. **33**, 4780-4789 (2019), https://ojs.aaai.org/index.php/AAAI/article/view/4405

[27] Liu, Y., Sun, Y., Xue, B., Zhang, M., Yen, G. & Tan, K. A Survey on Evolutionary Neural Architecture Search. *IEEE Transactions On Neural Networks And Learning Systems*. pp. 1-21 (2021)

[28] Lorenzo, P., Nalepa, J., Kawulok, M., Ramos, L. & Pastor, J. Particle Swarm Optimization for Hyper-Parameter Selection in Deep Neural Networks. *Proceedings Of The Genetic And Evolutionary Computation Conference*. pp. 481-488 (2017), https://doi.org/10.1145/3071178.3071208

[29] Capra, M., Bussolino, B., Marchisio, A., Masera, G., Martina, M. & Shafique, M. Hardware and Software Optimizations for Accelerating Deep Neural Networks: Survey of Current Trends, Challenges, and the Road Ahead. *IEEE Access*. **8** pp. 225134-225180 (2020)

[30] Ding, Y., Wu, Y., Huang, C., Tang, S., Wu, F., Yang, Y., Zhu, W. & Zhuang, Y. NAP: Neural architecture search with pruning. *Neurocomputing*. **477** pp. 85-95 (2022), https://www.sciencedirect.com/science/article/pii/S0925231221018361

[31] Bian, Y., Song, Q., Du, M., Yao, J., Chen, H. & Hu, X. Subarchitecture Ensemble Pruning in Neural Architecture Search. *IEEE Transactions On Neural Networks And Learning Systems*. pp. 1-9 (2021)

[32] Liang, T., Glossner, J., Wang, L., Shi, S. & Zhang, X. Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing*. **461** pp. 370-403 (2021), https://www.sciencedirect.com/science/article/pii/S0925231221010894

[33] Baldi, P., Sadowski, P. & Whiteson, D. Searching for exotic particles in high-energy physics with deep learning. *Nature Communications*. **5**, 4308 (2014,7), https://doi.org/10.1038/ncomms5308

2 Optimizing Convolutional Neural Networks

- Balderas, L., Lastra, M., & Benítez, J. M. (2024). Optimizing Convolutional Neural Network Architectures. *Mathematics*, 12(19), 3032. https://doi.org/10.3390/math12193032
  - Status: **Published**.
  - Impact Factor (JCR 2023): **2.3**
  - Subject Category: **Mathematics**
  - Rank: **21/490**
  - Quartile: **Q1 (D1)**

Optimizing Convolutional Neural Network Architectures

Luis Balderas
Department of Computer Science
and Artificial Intelligence
DiCITS, iMUDS, DaSCI
University of Granada, Granada, Spain 18071
`luisbalru@ugr.es`

Miguel Lastra
Department of Software Engineering
DiCITS, iMUDS, DaSCI
University of Granada, Granada, Spain 18071
`mlastral@ugr.es`

José M. Benítez
Department of Computer Science
and Artificial Intelligence
DiCITS, iMUDS, DaSCI
University of Granada, Granada, Spain 18071
`J.M.Benitez@decsai.ugr.es`

Abstract

Convolutional neural networks (CNNs) are commonly employed for demanding applications, such as speech recognition, natural language processing, and computer vision. As CNN architectures become more complex, their computational demands grow, leading to substantial energy consumption and complicating their use on devices with limited resources (e.g., edge devices). Furthermore, a new line of research seeking more sustainable approaches to Artificial Intelligence development and research is increasingly drawing attention: Green AI. Motivated by an interest in optimizing Machine Learning models, in this paper, we propose Optimizing Convolutional Neural Network Architectures (OCNNA). It is a novel CNN optimization and construction method based on pruning designed to establish the importance of convolutional layers. The proposal was evaluated through a thorough empirical study including the best known datasets (CIFAR-10, CIFAR-100, and Imagenet) and CNN architectures (VGG-16, ResNet-50, DenseNet-40, and MobileNet), setting accuracy drop and the remaining parameters ratio as objective metrics to compare the performance of OCNNA with the other state-of-the-art approaches. Our method was compared with more than 20 convolutional neural network simplification algorithms, obtaining outstanding results. As a result, OCNNA is a competitive CNN construction method which could ease the deployment of neural networks on the IoT or resource-limited devices.

The code implementing the method will be available at https://github.com/luisbalru/OCNNA.

Keywords convolutional neural network simplification; neural network pruning; efficient machine learning; Green AI

1 Introduction

Over the last years, deep neural networks (DNNs) have become the state-of-the-art technique in several challenging tasks, such as speech recognition [1], natural language processing [2], and computer vision [3]. In particular, convolutional neural networks (CNNs) have achieved remarkable success across a broad range of computer vision challenges, such as image classification [3], object detection in images [4], object detection in video [5], semantic segmentation [6], video restoration [7], and medical diagnosis [8]. The astonishing results of CNNs are associated with the huge number of annotated data and important advances in hardware. Unfortunately, CNNs usually have an immense number of parameters, incurring in high storage requirements, and significant computational and energetic costs [9]. This imposes severe restrictions on the deployment of these models on devices with limited resources, such as edge devices or mobile devices. Thus, some methods to develop smaller-size models or simplify the complexity of the available models are necessary [10].

On the other hand, these large models requiring large amounts of memory and computation time have an evident environmental impact. In 2019, researchers from the University of Massachusetts discovered that training various deep learning models, such as those involving neural architecture search, could release over 284019 kg of carbon dioxide. This amount is roughly equivalent to five times the total lifetime emissions of the typical American vehicle, including the car's production [11, 12]. Concretely, to classify a single image, the VGG-16 model [13] needs over 30 billion floating-point operations (FLOPs) and has 138 million parameters, which demand more than 500 MB of storage space [14].

In consequence, reducing the model's storage requirements and computational cost becomes critical for resource-limited devices, specially in IoT applications, embedded systems, autonomous agents, mobile devices, and edge devices [15]. Actually, the concerns relative to such a high level of energy consumption—among other resources—have lead to the surge and development of a new line of research seeking a more sustainable future for Artificial Intelligence development and deployment, named Green Artificial Intelligence [16].

Nonetheless, [17] remarks that a usual DNN property is their considerable redundancy in parameterization, which leads to the idea of reducing this redundancy by compressing the networks. However, a severe problem along with compressing the models is the loss of accuracy. In order to avoid it, there are some ways of designing efficient DNNs and generating effective solutions. For instance, one approach involves using memetic algorithms to discover an effective architecture for a given task, while another method entails employing alternative optimization algorithms to fine-tune the connection weights of a pre-defined architecture. Kernel Quantization is also used for efficient network compression [18]. Alternatively, pruning is a widely utilized technique for simplifying neural networks. Since the introduction of Optimal Brain Damage (OBD) and Optimal Brain Surgery (OBS) in 1990, pruning methods have been thoroughly researched for model compression. Over the years, numerous other approaches have been developed to create more efficient and effective neural networks of various types, including dense, convolutional, and recurrent networks [12]. The primary objective of pruning algorithms is to derive a subnetwork with significantly fewer parameters while maintaining the same level of accuracy.

In this paper, we propose a novel CNN optimization and construction technique called Optimizing Convolutional Neural Network Architectures (OCNNA) based on pruning which requires minimal tuning. OCNNA has been designed to assess the importance of convolutional layers. Since this measure is computed for every convolutional layer and unit from the model input to the output, it essentially reflects the importance of every single convolutional filter and its contribution to the information flow through the network [19]. Moreover, our method is able to sort convolutional filters within a layer by importance; as a consequence, it can be seen as a feature importance computation tool which can generate more explainable neural networks. OCNNA is easy to apply, having only one parameter (k), called the percentile of significance, which represents the proportion of filters which will be transferred to the new model based on their importance. Only the k-th percentile of filters with higher values after applying the OCNNA process will remain. The proposed OCNNA can directly be applied to trained CNNs, avoiding the training process from scratch.

We thoroughly evaluated the optimization efficacy of the OCNNA method compared with the state-of-the-art pruning techniques. Our experiments on the CIFAR-10, CIFAR-100 [20], and Imagenet [21] datasets and popular CNN architectures, such as ResNet-50, VGG-16, DenseNet40, and MobileNet, show that our algorithm leads to better performance in terms of the accuracy in prediction and the reduction in the number of parameters, following a cornerstone metric for Green AI [22] and efficient Machine Learning.

Our main contributions can be summarized as follows:

- A Green AI simplification method for CNNs called OCNNA is proposed. OCNNA measures the importance of each convolutional layer and unit from a trained model by combining well-known data science and statistical techniques, such as PCA, Frobenius norm, and Coefficient of Variation. After that, it builds a simplified model which can be even more precise than the original one and more efficient in terms of computational costs and memory footprint.
- Experiments on three benchmark datasets (CIFAR-10, CIFAR-100, and Imagenet) demonstrate that our simplification technique yields highly competitive results in terms of efficiency metrics (reduction in the number of parameters) and prediction accuracy when applied to the most widely used CNN models, such as VGG16, ResNet50, DenseNet40, and MobileNetV1.

The paper is organized as follows: Section 2 presents an overview of the latest methods for developing efficient deep neural networks. Our proposal is described in Section 3. In Section 4, our methodology is experimentally analyzed, and the results are analyzed and discussed in Section 5. Finally, Section 6 highlights the conclusions.

2 Previous work

The purpose of this section is to provide a brief overview of the main approaches in model compression: neuroevolution, neural architecture search, quantization, and pruning. Our research mainly focuses on convolutional neural network pruning.

2.1 Neuroevolution

Neuroevolution can be applied to several tasks related to efficient neural network design, such as learning neural networks (NN) building blocks, hyperparameters, or architectures. In 2002, NeuroEvolution of Augmenting Topologies (NEAT) was presented in [23], showing the effectiveness of a Genetic Algorithm (GA) in evolving NN topologies and strengthening the analogy with biological evolution. More recently, ref. [24] drew inspiration from NEAT, evolving deep neural networks by beginning with a simple neural network and gradually increasing its complexity through mutations. In [25], a more accurate approach, which consists in stacking the same layer module to make a deep neural network, like Inception, DenseNet, and ResNet, can be found [26].

2.2 Neural Architecture Search

Recently, neural architecture search (NAS), whose main goal is to automatically design the best DNN architecture, has achieved great importance in model design. On the one hand, NAS algorithms can be divided into two categories: (1) microstructure search focuses on identifying the best operation for each layer;

(2) macrostructure search aims to find the optimal number of channels or filters for each layer, or the ideal model depth [27]. Additionally, NAS algorithms can be categorized into three groups based on the optimizer used: reinforcement learning-based NAS algorithms, gradient-based NAS algorithms, and evolutionary NAS (ENAS) algorithms. In this sense, NSGA-II has been recently used for NAS creating NSGA-Net [28]. In [29], NATS-BEnch is proposed, consisting in a unified benchmark for topology and size aggregating more than 10 state-of-the-art NAS algorithms.

2.3 Quantization

Quantization refers to the process of approximating a continuous signal by using a set of discrete symbols or integer values [30]. In other words, it reduces computations by reducing the precision of the data type. Advanced quantization techniques, such as asymmetric quantization [31] or calibration-based quantization, have been presented to improve accuracy. In [30], we find a complete quantization guide and recommendations.

2.4 Knowledge Distillation

Knowledge distillation, which was first defined by [32] and generalized in [33], is the process of transferring knowledge from one neural network to a different one. In [34], a student–teacher framework is presented, introducing different scenarios, such as distillation based on the number of teachers (one teacher vs. multiple teachers), distillation based on data format (data-free, with a few samples, or cross-modal distillation), or online and teacher-free distillation.

2.5 Pruning

Network pruning is one of the most effective and prevalent approaches to model compression. Pruning techniques can be classified by various aspects: structured and unstructured pruning, depending on whether the pruned network is symmetric or not [35, 30]; neuron, weight, or filter pruning, depending on the network's element which is pruned; or static and dynamic pruning [30]. While static pruning removes elements offline from the network after training and before inference, dynamic pruning determines at runtime which elements will not participate in further activity [30]. Most researchers focus on how to find unimportant filters. Magnitude-based methods [36] use the magnitude of the weights in feature maps from certain layers as a measure of importance, pruning those with lower magnitudes. Others measure the importance of a filter through their reconstruction loss (Thinet) [37] or Taylor expansion [38, 39]. In [40], Average Percentage of Zeros (APoZ) is used to assess the proportion of zero activations in a neuron following ReLU mapping, thus allowing for the pruning of redundant neurons. HRank [41] understands filter pruning as an optimization problem, using the feature maps as the function which measures the importance of a filter part of the CNN. Inspired by HRank, FPWT [42] introduces a new method which transforms the feature map in the spatial domain into the frequency domain by using Wavelet Transform.

In [43], a new CNN compression technique is presented based on the filter-level pruning of redundant weights according to entropy importance criteria (FPEI) with different versions depending on the learning task and the NN. SFP [44] and FPGM, based on filter pruning via geometric median [45], use soft filter pruning; PScratch [46] proposes to prune from scratch, before training the model. In [47], a criterion for CNN pruning inspired by NN interpretability is proposed: the most relevant units are identified based on their relevance scores, which are derived from explainable AI (XAI) concepts. Ref. [48] introduces a data-driven CNN architecture determination algorithm called AutoCNN which consists of three training stages (spatial filter, classification layers, and hyperparameters). AutoCNN uses statistical parameters to decide whether to add new layers, prune redundant filters, or add new fully connected layers pruning low-information units. An iterative pruning method based on deep reinforcement learning (DRL) called Neon, formed by a DRL agent which controls the trade-off between performance and the efficiency of the pruned network, is introduced in [49]. For each hidden layer, Neon extracts the architecture-based and the layer-based feature maps which represent an observation. Then, the aforementioned hidden layer is compressed and fine-tuned. After that, a reward is calculated and used to update the deep reinforcement learning agent's weights. This process is repeated several times for the whole neural network.

In [50], a multi-objective evolution strategy algorithm called DeepPruningES is proposed. Its final output consists of three neural networks with different trade-offs called knee (with the best trade-off between training error and the number of FLOPs), boundary-heavy (with the smallest training error), and boundary-light solutions (with the smallest number of FLOPs). In [27], a customized correlation-based filter-level pruning method for deep CNN compression called COP, which removes redundant filters through their correlations, is presented. In [51], SCWC is introduced, a shared channel weight convolution method to decrease the number of multiplications in CNNs by leveraging the distributive property, made possible through structured channel parameter sharing. In [52], a new method called CHWP for identifying the most redundant filters is proposed, taking into account the size of filters, the difference between them, and the role of Batch Normalization layers. In [53], a training method for CNN compression is proposed. It integrates matrix factorization and regularization techniques, based on Singular Value Decomposition. Nonetheless, the method has been evaluated only on ResNet-18, ResNet-20, and ResNet-32.

In [54], a CNN pruning method called MOP-FMS is introduced, in which the pruning task is modeled as a bi-objective optimization problem based on feature map selection. The two objectives of the method are accuracy and FLOPs, which are achieved by using an ad hoc evolutionary optimization algorithm designed to perform the pruning. Ref. [55] presents a CNN pruning method, which obtains a simplified network by clustering its filters. After that, it searches the optimal compact network structure by applying a social group optimization algorithm. In [56], an evolutionary method called Bi-CNN-Pruning is proposed to prune filters and channels with the objective of preserving the performance of the original model according to different pruning criteria, such as weight magnitude or activation statistics, among others. Concretely, it maintains the important channels in an ordered way, i.e., useful filters are selected first and then the channels. ResPrune is proposed in [57]. It is a selection filter method which uses two criteria: l_2-norm and redundancy. Unlike other methods, ResPrune does not completely omit the filters identified as irrelevant; instead, it restores them to their original values by using stochastic techniques. In [58], a pruning framework called MGPF is introduced. MGPF generates sparse models of different granularity without fine-tuning the remaining connections after pruning. CIE [59] is a cross-layer importance evaluation technique used for neural network pruning, which assesses the significance of convolutional layers based on the model's prediction accuracy. This evaluation is highly efficient in terms of time, as the process is performed only once for a given model.

2.6 Summary

As demonstrated in this section, there are numerous approaches and techniques for simplifying CNNs. Over the years, increasingly sophisticated solutions have been proposed; whether through neuroevolution, neural architecture search, quantization, knowledge distillation, or pruning, the goal remains the same: to achieve smaller models without sacrificing prediction accuracy. However, designing an efficient, versatile, and effective method is not easy. In the case of neuroevolution and neural architecture search, competitive techniques can be achieved in terms of accuracy, but the computational cost of generating them can be prohibitive. For quantization and knowledge distillation, conceptual challenges may arise, while pruning algorithms are typically designed for specific models, unable to tackle the simplification of networks of different natures or assess their performance on various benchmarks. To address all these challenges, we developed OCNNA, an efficient technique that will be evaluated on the de facto benchmarks for image classification and will simplify the most paradigmatic and general convolutional networks. It can be applied in virtually any industry or academic use case involving CNNs and image classification.

3 Proposal

In this paper, we address the challenge of finding an optimized topology for a convolutional neural network. Thus, we introduce the Optimizing Convolutional Neural Network Architectures (**OCNNA**) method, a new convolutional neural network construction method based on pruning. In this section, we provide a detailed description of the method.

3.1 Notation

First of all, we introduce the notation used in our proposal. Let Ω be a neural network with L convolutional layers. Let w^l_m and o^l_m be the convolutional filter and the output of the l-th layer. The subscript $m \in 1, \dots, M^l$ represents the filter index, where M^l indicates the total number of output filters in the corresponding layer. In consequence, pruning the l-th filter in layer m implies removing the corresponding w^l_m.

Principal Component Analysis (PCA) [60] is a data analysis tool applied to identify the most meaningful basis to re-express, revealing a hidden structure, a given dataset. We define $P^l_m = PCA(o^l_m)$ as the matrix result of computing PCA on the m-th filter's output of the l-th layer.

The Frobenius norm [61] is a norm of an $m \times n$ matrix A defined as

$$||A||_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{N} |a_{ij}|^2} \tag{1}$$

It is also equal to the square root of the matrix trace of A, A^H:

$$||A||_F = \sqrt{Tr(AA^H)} \tag{2}$$

where A^H is the conjugate transpose [62]. Let us define $F^l_m = ||P^l_m||_F$ as the Frobenius norm of the above PCA calculation (on the m-th filter's output of the l-th layer).

The Coefficient of Variation (CV) is the relationship between mean and standard deviation [63]. If D is a data distribution, with σ_D its standard deviation and μ_D its mean, CV is calculated as

$$CV_D = \frac{\sigma_D}{\mu_D} \tag{3}$$

It is attractive as a statistical tool because it permits the comparison of variables free from scale effects (dimensionless). We call $C = CV(x)$ if $x \in \mathbb{R}^n$, for a given $n \in \mathbb{N}$.

Finally, we define the percentile of significance k. Only the k-th percentile of filters with higher values in terms of importance will remain. All notations can be found in Table 1.

3.2 OCNNA: the algorithm

Since the main components of a CNN are the convolutional filters, OCNNA is designed to identify the most important ones, thus creating a new model in which the less significant convolutional units are not included. This way, OCNNA generates a more efficient model in terms of the number of parameters with

Table 1: Notations and definitions

Notation	Definition
L	Number of convolutional layers
w^l_m	m-th filter from the l-th convolutional layer
o^l_m	Output of the m-th filter from the l-th layer
M^l	Number of output filters in the l-th layer
P^l_m	PCA applied to the m-th filter from the l-th layer
F^l_m	Frobenius norm of P^l_m
C	Coefficient of Variation of a vector
k-th percentile	Percentile of significance

minimum precision loss—as we will see in the next section. Additionally, OCNNA allows for the ordering of the convolutional filters by importance, providing a feature importance assessment method and, as a result, helping to better understand these models. Our method employs three important techniques to identify the significant filters: Principal Components Analysis (PCA), for selecting the most important features based on their hidden structure; the Frobenius norm, to summarize the PCA output information; and the Coefficient of Variation (CV), to measure the variability of Frobenius norm outputs. Figure 1 shows, as an example, the simplification process of a VGG-16 convolutional filter. As can be seen, the algorithm takes the filters from each layer (Figure 1, 1); applies PCA, Frobenius norm, and Coefficient of Variation (Figure 1, 2); and ranks the filters by importance, selecting only their top k-th percentile (Figure 1, 3). More details can be found in Algorithm 1.

Algorithm 1 OCNNA

```
1:  function OCNNA(model, $k$, $D_{var}$)
2:      compressed_model = new Model()
3:      for layer in model.Layers do
4:          if layer is Convolutional then
5:              variability = List()
6:              for filter in layer do
7:                  $o$ = model.predict($D_{var}$)
8:                  $p$ = PCA($o$, $95\%$ var)
9:                  $pn$ = FrobeniusNorm($p$)
10:                 $c$ = CV($pn$)
11:                 Append $c$ to variability
12:             end for
13:             new_layer_index = get $k$-th percentile in variability
14:             Add new layer with new_layers_index filters from model
15:         else
16:             Add layer to compressed_model
17:         end if
18:     end for
19:     return compressed_model
20: end function
```

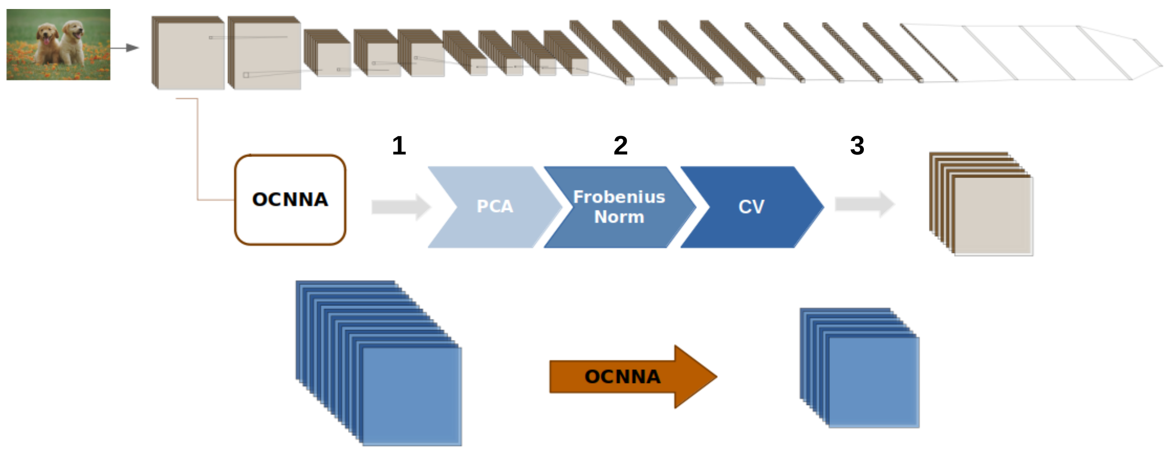


Figure 1: OCNNA applied to VGG-16. Given the output from the i-th convolutional layer, PCA, Frobenius norm, and Coefficient of Variation are applied to identify the most significant filters. The k-th percentile of filters, in terms of importance, are selected, generating a new model whose i-th convolutional layer is a optimized version of the original one. This approach is applied to every convolutional filter.

Given convolutional layer w^l, a D_{var} dataset, used exclusively for measuring the importance of filters, is evaluated by generating output o^l. o^l is formed by M^l filters. In consequence, o^l_m is the m-th filter's output from the l-th layer. By using o_l, OCNNA helps us to measure the importance of the M^l filters in the m-th layer. Concretely, we adopt a three-stage process. First, for each filter and image contained in D_{var}, we apply PCA retaining the 95% of variance.

$$P^l_m = PCA(o^l_m) \tag{4}$$

with P^l_m a matrix which contains the most meaningful features generated by the m-th filter of the l-th convolutional layer. Nonetheless, the information embedded in P^l_m is too large. In consequence, we apply the Frobenius norm to summarize this information:

$$F^l_m = ||P^l_m||_F \tag{5}$$

obtaining a vector F^l_m, in which each component is the result of the process described above applied to each image from D_{var}. Finally, we calculate the CV:

$$C_m = CV(F^l_m) \tag{6}$$

C_m is a number which summarizes the m-th filter significance within the l-th convolutional layer by measuring the variability of the process PCA and Frobenius norm for each image in D_{var}. In other words, OCNNA gives a low score of importance to a filter if for a subset of images, it generates an output whose hidden structures (PCA, 95% variance), after being summarized (Frobenius norm), have little variability (CV).

To sum up, OCNNA is able to extract insights and measure the importance of each filter of a convolutional layer, starting from hundreds of arrays which constitute the output from a dataset, called D_{var}, and generating a holistic vision summarizing the filter significance into a single number (Figure 2). As a result, OCNNA transforms the output of a convolutional layer into an array in which the i-th component represents the i-th filter's importance. Finally, using parameter k, or percentile of significance, we extract the k-th percentile of filters in terms of significance, completing the simplification process.

The larger k, the more strict the filter selection. In consequence, fewer filters will be selected and the new model will be simpler (a smaller number of parameters compared with the original one).

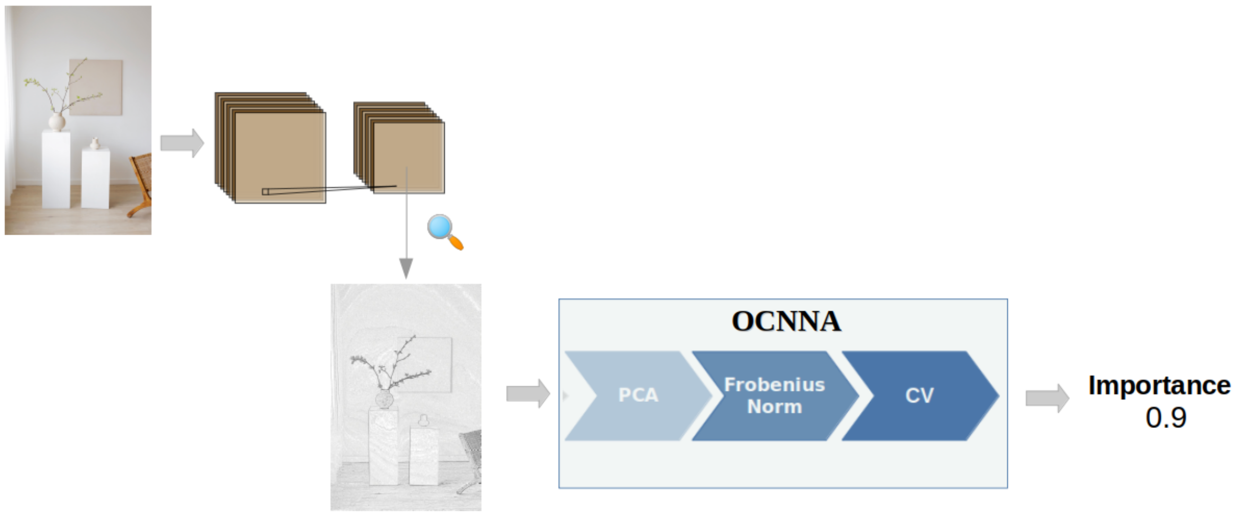


Figure 2: OCNNA provides a single number for this filter which reflects its importance. This process is iterated over all filters from a layer and their k-th percentile in terms of significance will form part of the new model.

3.3 Implementation

As mentioned above, OCNNA can measure the importance of a convolutional filter by extracting hidden insights from a multidimensional array and express it through a number.

This process implies heavy computational costs. In this sense, OCNNA is designed to maximize its performance in terms of the time required to complete the simplification process by proposing a parallel computing paradigm. In other words, this entails counting the number of CPUs available and distributing the tasks associated to each filter (prediction, PCA, Frobenius norm, and CV) of the convolutional layer among them, carrying out the calculations simultaneously and, as a result, speeding-up the results. This parallelism is absolutely transparent to the user. Once all the operations are finished, a synchronization process between the different subtasks is accomplished, mapping every result (the significance of the filter) in the correct component of the vector, which represents the importance of each filter in the convolutional layer. It was implemented in Python 3.9, and Tensorflow 2.9 was used as the machine learning framework.

4 Empirical Evaluation

To assess the performance of OCNNA, we designed a thorough empirical procedure that included different well-known datasets (CIFAR-10, CIFAR-100, and Imagenet) and architectures (ResNet-50, VGG-16, DenseNet-40, and MobileNet) which represent core benchmarks extensively referenced in the literature. Moreover, OCNNA was compared with 20 state-of-the-art CNN simplification techniques, obtaining successful results. This section is structured as follows: The architectures and datasets, metrics, compared state-of-the-art approaches, and training process settings are explained in order to assure the experiments' reproducibility. Finally, the results and analysis for the CIFAR and Imagenet datasets are presented, comparing them with the other state-of-the-art techniques.

4.1 Common Architectures and Datasets

We thoroughly evaluated our CNN building and optimizing scheme. To obtain comparable results with other state-of-the-art approaches, we selected four popular CNN architectures: VGG-16 [13], ResNet-50 [64], DenseNet-40 [65], and MobileNet [66]. VGG-16 is a convolutional neural network formed by 138.4M parameters and 16 layers. The input is an RGB image with a size of 224×224, which is passed through a stack of 3×3 receptive field convolutional layers, with padding fixed to 1 pixel. Five max-pooling layers (pixel window of size 2×2 and stride set to 2), which follow some of the convolutional layers, are included as spatial pooling. The last stack of convolutional layers is followed by three dense layers of 4096, 4096, and 1000 channels.

ResNet-50, drawing inspiration from the VGG networks, incorporates the idea of residual learning to simplify training by reconfiguring the layers to learn residual functions relative to their inputs, rather than learning functions without references. In practice, ResNet includes shortcut connections and has lower complexity than VGG-16, given the fact that it is formed by 25.6M parameters. DenseNet (1M parameters and 40 layers) connects each convolutional unit as if it were a feed-forward neural network, reducing the number of parameters and diminishing problems such as the vanishing gradient. Finally, MobileNet is a light-weight neural network designed for mobile and embedded vision apps. It has 4.3M parameters and 55 layers.

To demonstrate the versatility of our approach, we evaluated it by using a core set of widely used benchmark datasets for image classification: CIFAR-10 [20], CIFAR-100 [20], and ImageNet [21]. The CIFAR-10 dataset is an image classification problem with 10 classes formed by 60,000 images with size 32×32 (each class consists of 6000 images, with a total of 50000 images for training and 10,000 images for testing). CIFAR-100 contains the same number of images as CIFAR-10 but 100 classes (600 images each). On the other hand, Imagenet is a dataset formed by 1431,167 annotated images (224×224) and 1000 object classes. As we have mentioned, OCNNA requires a D_{var} dataset to measure convolutional filter importance. In the case of CIFAR-10 and CIFAR-100, we selected 10% of the training images, in other words, 5000 images identically distributed by class. After completing the optimization process, we evaluated the new model's performance by using the test images. For the Imagenet dataset, we used as D_{var} the "imagenet_v2/topimages" subset and as test set "imagenet_v2/matched-frequency". Both of them have 10,000 images sampled after a decade of progress on the original ImageNet dataset, making the new test data independent of existing models and guaranteeing that the accuracy scores are not affected by adaptive overfitting [67]. These datasets can be found in [68].

4.2 Metrics

We measured the prediction performance of the optimized models with accuracy (ACC). In addition, we registered the number of parameters to assess complexity and efficiency in terms of memory requirements and runtime as it is defined for the Green AI paradigm (a lower number of parameters may lead to increased efficiency). We also recorded the remaining parameters ratio (RPR) compared with the original model for compression, as previously performed for other state-of-the-art approaches. A higher parameter reduction ration means a smaller model size and, as a result, a less complex model. The definition of RPR is

$$RPR = 1 - \frac{NP_O - NP_S}{NP_O} \tag{7}$$

where NP_O and NP_S represent the number of parameters of the original model and the optimized one, respectively. In any case, we adjusted the metrics (and their presentation) to match those used in the literature, ensuring a fair and precise comparison between OCNNA and other approaches.

4.3 Training Process Settings

For the VGG-16, ResNet-50, MobileNet, and DenseNet models training on CIFAR-10 or CIFAR-100, we set the weight decay to 1×10^{-6}, the momentum to 0.9, the learning rate to 1×10^{-3}, and the batch size to 64. All images were augmented by horizontal and vertical flipping, zoom with range between 0.85 and 1.5, rotation range of 180, and fill mode as reflect; in other words, pixels outside the boundaries of the input image were filled according to the following mode [69]:

$$abcddcba|abcd|dcbaabcd$$

We did not train any model on Imagenet due to the existence of publicly available pre-trained models.

4.4 Results and Analysis on CIFAR Datasets

The results on the CIFAR datasets for different architectures are shown in Tables 2–5. The Dataset column shows the learning task; the Architecture column shows the neural network used in the learning task; the Base (%) column refers to the accuracy originally obtained with the aforementioned architecture for the dataset given; Acc (%) is the accuracy obtained after applying the pruning algorithm; RPR means the remaining parameters ratio, where the lower, the better; Acc. Drop is the accuracy loss after pruning, where the smaller, the better. As we have said, we used three benchmark architectures, ResNet-50, VGG-16, and DenseNet-40.

In the case of ResNet-50, OCNNA generates a compressed model where just over 45% of the parameters remain (57.12% for CIFAR-10 and 46.95% for CIFAR-100), while the accuracy loss is very small. As a result, OCNNA is an effective method for compressing CNNs; actually, it achieves the best values for both metrics among all the considered methods.

Given the fact that VGG-16 is a very complex network, it might present greater redundancy than the other architectures. In fact, we were able to reduce the model, pruning 86.68% of the parameters for CIFAR-10 and 74.03% for CIFAR-100 without any accuracy loss for CIFAR-10 (actually, an improvement of 0.42%) and keeping it nearly unchanged for CIFAR-100 (-0.44%). OCCNA achieved the best scores for both metrics in CIFAR-10 and CIFAR-100, with the exception of RPR, where it ranked third.

For DenseNet-40, OCNNA again achieved the best results in both metrics and both datasets. In addition, it improved test accuracy for CIFAR-10 (0.39%) and for CIFAR-100 (0.23%).

Finally, we can observe the same behavior for MobileNet: OCNNA ranked first in both RPR and ACC for both datasets but only in size reduction for CIFAR-10. While not compressing the most for MobileNet built for CIFAR-10, its simplified model improves test accuracy (0.65%).

Overall, OCNNA is the winner according to the results for the CIFAR-10 and CIFAR-100 datasets.

Table 2: Results of pruning ResNet-50 on CIFAR datasets. RPR means the remaining parameters ratio, where the lower, the better. Acc. Drop is the accuracy loss after pruning, where the smaller, the better. The best results are highlighted in bold.

Dataset	**Base (%)**	**Algorithm**	**Acc. (%)**	**Acc. Drop (%)**	**RPR (%)**
CIFAR-10	93.55	FPEI [43]	91.85	1.70	45.69
		LRP [70]	93.37	0.18	75.24
		DeepPruningES [50]	91.89	1.66	78.69
		OCNNA	**93.42**	**0.13**	**42.88**
CIFAR-100	73.24	FPEI	69.58	3.66	57.53
		DeepPruningES	57.81	15.43	80.91
		OCNNA	**70.32**	**2.92**	**53.05**

Table 3: Results of pruning VGG-16 on CIFAR datasets. RPR means the remaining parameters ratio, where the lower, the better. Acc. Drop is the accuracy loss after pruning, where the smaller, the better. The best results are highlighted in bold.

Dataset	**Base (%)**	**Algorithm**	**Acc. (%)**	**Acc. Drop (%)**	**RPR (%)**
CIFAR-10	93.70	FPGM [40]	93.00	0.7	48.97
		DeepPruningES	91.79	1.91	64.99
		ThiNet [40]	92.99	0.71	26.92
		PScratch [71]	93.02	0.68	26.96
		HRank [41]	93.43	0.27	17.10
		Slimming [36]	93.44	0.26	16.71
		COP v1 [27]	93.37	0.18	15.15
		COP v2	93.86	−0.17	13.56
		SNACS [72]	91.06	−0.17	**3.84**
		White-Box [73]	93.47	0.23	–
		SOKS-80% [74]	94.01	−0.31	–
		EACP(k=80%) [55]	93.29	0.41	24.6
		MOP-FMS (80%) [54]	91.51	2.19	19.50
		Bi-CNN-Pruning [56]	93.88	−0.18	63.48
		ResPrune [57]	93.76	−0.27	–
		MGPF [58]	93.88	−0.18	9.14
		FPWT [42]	93.94	−0.24	26.70
		OCNNA	**94.12**	−**0.42**	13.32
CIFAR-100	73.51	SFP [75]	71.74	1.77	60.66
		DeepPruningES	67.06	6.45	80.07
		FPGM	72.76	1.25	48.99
		HRank [41]	72.43	1.08	44.07
		COP v1	72.63	0.88	34.81
		Slimming	72.76	0.75	33.40
		COP v2	72.99	0.52	26.16
		Bi-CNN-Pruning	72.11	1.4	78.27
		OCNNA	**73.07**	**0.44**	**25.97**

Table 4: Results of pruning DenseNet-40 on CIFAR datasets. RPR means the remaining parameters ratio, where the lower, the better. Acc. Drop is the accuracy loss after pruning, where the smaller, the better. The best results are highlighted in bold.

Dataset	**Base (%)**	**Algorithm**	**Acc. (%)**	**Acc. Drop (%)**	**RPR (%)**
CIFAR-10	76.52	HRank	75.94	0.58	58.68
		Slimming	75.90	0.62	54.28
		COP v1	75.53	0.99	56.10
		COP v2	76.03	0.49	54.08
		OCNNA	**76.91**	**−0.39**	**52.96**
CIFAR-100	94.84	HRank	93.68	0.60	39.00
		Slimming	94.35	0.49	34.80
		COP v1	94.19	0.65	37.66
		COP v2	94.54	0.30	34.80
		OCNNA	**95.07**	**−0.23**	**34.12**

Table 5: Results of pruning MobileNet on CIFAR datasets. RPR means the remaining parameters ratio, where the lower, the better. Acc. Drop is the accuracy loss after pruning, where the smaller, the better. MobileNet-0.75 means that every layer is 75% of the original one. COP-0.50 implies that 50% of filters are maintained. The best results are highlighted in bold.

Dataset	**Base (%)**	**Algorithm**	**Acc. (%)**	**Acc. Drop (%)**	**RPR (%)**
CIFAR-10	94.07	MobileNet-0.75	93.36	0.71	53.96
		MobileNet-0.50	92.84	1.23	25.28
		COP v1-0.50	93.59	0.48	34.06
		COP v1-0.30	92.97	1.1	25.94
		COP v2-0.50	93.89	0.18	32.72
		COP v2-0.30	93.47	0.6	25.38
		Adapt-DCP	94.57	−0.6	21.74
		MGPF	92.5	1.5	**9.14**
		OCNNA	**94.72**	**−0.65**	24.60
CIFAR-100	74.94	MobileNet-0.75	73.99	0.95	53.96
		MobileNet-0.50	73.20	1.74	25.28
		COP v1-0.50	73.95	0.99	42.50
		COP v1-0.30	73.45	1.49	25.35
		COP v2-0.50	74.66	0.28	40.85
		COP v2-0.30	74.01	0.93	25.42
		CIE [59]	57.53	17.41	–
		OCNNA	**74.72**	**0.22**	**23.87**

4.5 Results and Analysis on Imagenet Dataset

The performance of OCNNA and state-of-the-art methods for VGG-16, ResNet-50, and MobileNet on the Imagenet dataset are presented in Tables 6–8. We begin by describing the results obtained for the VGG-16 architecture (Table 6). In this case, the best performing algorithm for compression was MGPF, at the cost of a suboptimal accuracy. OCNNA ranked second in RPR.

For ResNet-50 (Table 7), more extensive experimental results can be found in the literature. To the best of our knowledge, ResPrune and SCWC ($s = 0.5$, $s = 0.4$, and $s = 0.3$) are the only methods which improve the accuracy (negative accuracy drop) but with an RPR above 65% for SCWC (no data available for ResPrune).

FPEI-R7 with DR obtains a notable RPR (37.88%), still smaller than OCNNA's (37.44%), but the accuracy drop is quite higher (1.68% vs. 0.57% for OCNNA). In [42] (FPWT), we found the best approach to compressing ResNet-50 (31.8%). OCNNA was not as effective as FPWT in terms of RPR, but the accuracy drop for OCNNA was more than four times smaller compared with FPWT (0.57% for OCNNA vs. 2.48% for FPWT).

In real-world applications, it is essential to balance performance and compression rates based on varying computational demands, energy consumption constraints [39], and accuracy requisites. For MobileNet (Table 8), OCNNA outperformed the different approaches of the COP method and the direct simplification of the original model.

As a final, overall conclusion, we can assert that OCNNA can obtain simplified CNNs with remarkable complexity reduction while retaining the accuracy or even improving it in some cases.

Table 6: Comparison of OCNNA and other methods on Imagenet (VGG-16). RPR means the remaining parameters ratio, where the lower, the better. Acc. is the test accuracy after pruning. Acc. Drop is the test accuracy loss after pruning, where the smaller, the better. The best results are highlighted in bold.

Base (%)	**Method**	**Acc (%)**	**Acc. Drop (%)**	**RPR (%)**
74.4% [13]	$SCWC_{(s=0.6)}$ [51]	**73.80**	**0.6**	60.5
	$SCWC_{(s=0.5)}$	73.20	1.2	50.3
	$SCWC_{(s=0.4)}$	73.17	1.23	40.8
	$SCWC_{(s=0.3)}$	72.16	1.64	30.6
	$SCWC_{(s=0.2)}$	71.42	2.98	19.7
	APoZ [40]	73.09	1.31	49.0
	CP [75]	70.7	3.7	–
	ThiNet-GAP [40]	72.64	1.76	–
	SSR [71]	72.75	1.65	–
	MGPF	70.96	3.44	**6.0**
	OCNNA	71.54	2.86	18.9

5 Sensibility Study

As we have seen, OCNNA is a parametric algorithm designed to simplify CNN models. It can be applied to any convolutional model (e.g., ResNet or VGG networks) without any adjustment, in contrast with other state-of-the-art approaches, such as FPEI [43], which presents different versions (FPEI, FPEI-R4 with DDR, FPEI-R5, FPEI-R6, and FPEI-R7 with DR) in order to ensure quality in prediction in different situations. OCNNA counts only with one parameter, k, which represents the k-th percentile of filters with higher importance, after applying transformations such as PCA, Frobenius norm, and CV. What effect does changing

Table 7: Comparison of OCNNA and other methods on Imagenet (ResNet-50). RPR means the remaining parameters ratio, where the lower, the better. Acc. Drop is the accuracy loss after pruning, where the smaller, the better. The best results are highlighted in bold.

Base (%)	**Method**	**Acc. (%)**	**Acc. Drop (%)**	**RPR (%)**
75.3% [64]	HRank-C1 [41]	74.13	1.17	62.99
	FPEI-R5 [43]	74.36	0.94	61.79
	FPEI-R4 with DR	74.72	0.58	44.31
	HRank-C2	71.13	4.17	53.71
	FPEI-R6	72.23	3.07	51.53
	FPEI-R7 with DR	73.62	1.68	37.88
	SFP [75]	74.18	1.12	61.74
	FPGM [40]	73.54	1.76	61.79
	PScratch [71]	74.75	0.55	49.95
	COP v2 [27]	74.97	0.33	44.79
	$SCWC_{(s=0.5)}$ [51]	75.52	-0.22	77.20
	RED++ [76]	75.2	0.1	55.00
	$SCWC_{(s=0.4)}$	75.45	-0.15	72.60
	$SCWC_{(s=0.3)}$	75.35	-0.05	67.90
	$SCWC_{(s=0.2)}$	75.23	0.07	63.30
	$SCWC_{(s=0.1)}$	75.17	0.13	58.60
	Thinet-70 [37] [71]	74.03	1.27	67.10
	SSR [71]	72.47	2.83	47.80
	APoZ [40]	71.83	3.47	47.80
	LSTM-SEP [77]	74.4	0.9	57.00
	AOFP-C2 [77]	75.07	0.23	–
	Adapt-DCP [78]	74.44	0.86	45.10
	ResNet-50-pruned-70 [14]	75.06	0.24	70.00
	ResNet-50-pruned-50 [14]	73.99	1.31	50.00
	IoT-Qi [39]	72.92	2.38	33.70
	SNACS [72]	74.65	0.65	44.90
	White-Box [73]	74.21	1.09	–
	CHWP [52]	74.95	0.35	–
	EACP ($k = 70\%$) [55]	73.95	1.35	45.30
	ResPrune [57]	**76.15**	**-0.85**	–
	FPWT [42]	72.82	2.48	**31.80**
	CIE [59]	74.06	1.24	–
	OCNNA	74.73	0.57	37.44

Table 8: Results of pruning MobileNet on the Imagenet dataset. RPR means the remaining parameters ratio, where the lower, the better. Acc. Drop is the accuracy loss after pruning, where the smaller, the better. MobileNet-0.75 means that every layer is 75% of the original one. COP-0.50 implies that 50% of filters are maintained. The best results are highlighted in bold.

Dataset	**Base (%)**	**Algorithm**	**Acc. (%)**	**Acc. Drop (%)**	**RPR (%)**
Imagenet	69.96	MobileNet-0.75	68.01	1.95	60.94
		MobileNet-0.50	63.29	6.67	36.29
		COP v1-0.70	68.52	1.44	59.31
		COP v1-0.40	64.38	5.58	28.99
		COP v2-0.70	69.02	0.94	57.09
		COP v2-0.40	65.33	4.63	29.81
		Adapt-DCP	69.58	0.38	66.73
		OCNNA	**69.75**	**0.21**	**27.22**

the value of k have in terms of accuracy and network simplification? It is illustrated through the experiment depicted in Figure 3. In this experiment, different values of k, ranging between 10 and 75, were used, and the resulting accuracy and final number of parameters were evaluated. As the k value increases, OCNNA boosts the reduction in the number of filters which will form part of the new model. In consequence, the number of parameters substantially drops. Nonetheless, accuracy also suffers a dramatically drop as k increases. Notice the remarkable drop between $k = 40$ (74.43% in accuracy) and $k = 50$ (46.31%). In fact, 40-th percentile is the best value of k, obtaining the highest possible accuracy bearing in mind the significant reduction in parameters.

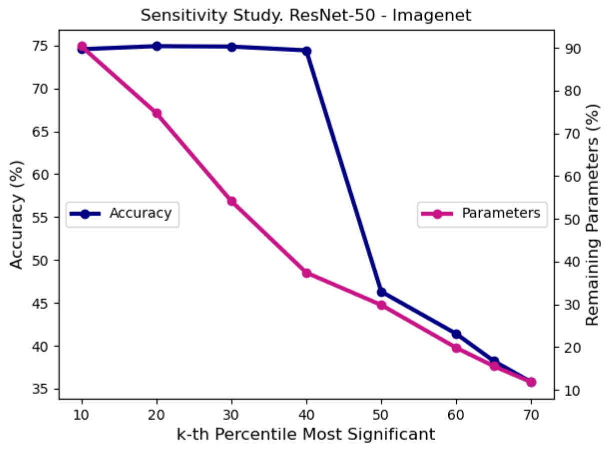


Figure 3: Sensitivity study of k percentile of significance for ResNet-50 and Imagenet dataset. The left Y-axis shows test accuracy, and the right Y-axis shows the remaining parameters ratio. The base accuracy is 75.4%. As we can see, when $k = 40$ (40-th percentile), we obtain a significant reduction in parameters (remaining 37.44%) with an accuracy drop of 0.57%.

6 Conclusions

Deep neural networks have emerged as the leading technique for tackling various challenging tasks in AI. In particular, CNNs have achieved an extraordinary success in a wide range of computer vision problems. However, these models come with high energy demands and are challenging to design efficiently.

In this paper, we introduce OCNNA, a novel CNN optimization and construction method based on pruning designed to assess the importance of convolutional layers, ordering the filters (features) by importance. Our approach enables the efficient end-to-end training and compression of CNNs. It is easy to apply and depends on a single parameter k, called percentile of significance, which represents the proportion of filters which will be transferred to the new model based on their importance. Only the k-th percentile of filters with higher values after applying the OCNNA process (PCA for feature selection, Frobenius norm for summary, and CV for measuring variability) will form part of the new optimized model.

OCNNA was evaluated through a comprehensive and detailed experimentation including the best known datasets (CIFAR-10, CIFAR-100, and Imagenet) and CNN architectures (VGG-16, ResNet-50, DenseNet-40, and MobileNet). The experimental results, based on the comparison with 20 state-of-the-art CNN simplification techniques and obtaining successful results, confirm that more efficient CNN models, following typical Green AI metrics, can be obtained with small accuracy losses. As a result, OCNNA is a competitive CNN construction method based on pruning which could ease the deployment of AI models onto edge devices (e.g., IoT devices) or other resource-limited devices.

Acknowledgment

This research was partially supported by Proyecto PID2020-118224RB-100 and Proyecto PID2023-151336OB-I00 financiado por MICIU/AEI /10.13039/501100011033 y por FEDER, UE.

References

[1] Graves, A.; Schmidhuber, J. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Netw.* **2005**, *18*, 602–610. https://doi.org/10.1016/j.neunet.2005.06.042.

[2] Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; Kuksa, P. Natural Language Processing (almost) from Scratch. *arXiv* **2011**, arXiv:1103.0398.

[3] Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. In Proceedings of the 26th Annual Conference on Neural Information Processing Systems 2012, Lake Tahoe, NV, USA, 3–6 December 2012; Volume 25.

[4] Zhao, H.; Yang, G.; Wang, D.; Lu, H. Deep mutualearning for visual object tracking. *Pattern Recognit.* **2021**, *112*, 107796. https://doi.org/10.1016/j.patcog.2020.107796.

[5] Olmos, R.; Tabik, S.; Herrera, F. Automatic handgun detection alarm in videos using deepearning. *Neurocomputing* **2018**, *275*, 66–72. https://doi.org/10.1016/j.neucom.2017.05.012.

[6] Yu, Q.; Gao, Y.; Zheng, Y.; Zhu, J.; Dai, Y.; Shi, Y. Crossover-Net: Leveraging vertical-horizontal crossover relation for robust medical image segmentation. *Pattern Recognit.* **2021**, *113*, 107756. https://doi.org/10.1016/j.patcog.2020.107756.

[7] Guo, J.; Chao, H. Building an end-to-end spatial-temporal convolutional network for video super-resolution. In Proceedings of the 31th AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017; pp. 4053–4060.

[8] Cai, L.; Gao, J.; Zhao, D. A review of the application of deepearning in medical image classification and segmentation. *Ann. Transl. Med.* **2020**, *8*, 713.

[9] Han, S.; Pool, J.; Tran, J.; Dally, W.J. Learning both Weights and Connections for Efficient Neural Networks. *arXiv* **2015**, arXiv:1506.0262.

[10] Khouas, A.R.; Bouadjenek, M.R.; Hacid, H.; Aryal, S. Training Machine Learning Models at the Edge: A Survey. *arXiv* **2024**, arXiv:2403.02619.

[11] Strubell, E.; Ganesh, A.; McCallum, A. Energy and Policy Considerations for Deep Learning in NLP. *arXiv* **2019**, arXiv:1906.02243.

[12] Balderas, L.; Lastra, M.; Benítez, J.M. Optimizing dense feed-forward neural networks. *Neural Netw.* **2024**, *171*, 229–241. https://doi.org/10.1016/j.neunet.2023.12.015.

[13] Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv* **2015**, arXiv:1409.1556.

[14] Alqahtani, A.; Xie, X.; Jones, M.W.; Essa, E. Pruning CNN filters via quantifying the importance of deep visual representations. *Comput. Vis. Image Underst.* **2021**, *208–209*, 103220. https://doi.org/10.1016/j.cviu.2021.103220.

[15] Tu, Y.; Lin, Y. Deep Neural Network Compression Technique Towards Efficient Digital Signal Modulation Recognition in Edge Device. *IEEE Access* **2019**, *7*, 58113–58119. https://doi.org/10.1109/ACCESS.2019.2913945.

[16] Bolón-Canedo, V.; Morán-Fernández, L.; Cancela, B.; Alonso-Betanzos, A. A review of green artificial intelligence: Towards a more sustainable future. *Neurocomputing* **2024**, *599*, 128096. https://doi.org/10.1016/j.neucom.2024.128096.

[17] Denton, E.; Zaremba, W.; Bruna, J.; LeCun, Y.; Fergus, R. Exploitinginear structure within convolutional networks for efficient evaluation. In Proceedings of the Annual Conference on Neural Information Processing Systems 2014, Montreal, QC, Canada, 8–13 December 2014. pp. 1269–1277.

[18] Yu, Z.; Shi, Y. Kernel Quantization for Efficient Network Compression. *IEEE Access* **2022**, *10*, 4063–4071. https://doi.org/10.1109/ACCESS.2022.3140773.

[19] Gong, M.; Gao, Y.; Wu, Y.; Zhang, Y.; Qin, A.K.; Ong, Y.S. Heterogeneous Multi-Party Learning With Data-Driven Network Sampling. *IEEE Trans. Pattern Anal. Mach. Intell.* **2023**, *45*, 13328–13343. https://doi.org/10.1109/TPAMI.2023.3290213.

[20] Krizhevsky, A. Learning Multiple Layers of Features from Tiny Images. 2009. Available online: https://www.cs.toronto.edu/~kriz/cifar.html(accessed on 15 May 2023).

[21] Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. ImageNet large scale visual recognition challenge. *Int. J. Comput. Vis.* **2015**, *115*, 211–252.

[22] Schwartz, R.; Dodge, J.; Smith, N.A.; Etzioni, O. Green AI. *Commun. ACM* **2020**, *63*, 54–63.

[23] Stanley, K.O.; Miikkulainen, R. Evolving Neural Networks through Augmenting Topologies. *Evol. Comput.* **2002**, *10*, 99–127. https://doi.org/10.1162/106365602320169811.

[24] Real, E.; Moore, S.; Selle, A.; Saxena, S.; Suematsu, Y.L.; Tan, J.; Le, Q.V.; Kurakin, A. Large-scale evolution of image classifiers. In Proceedings of the 34th International Conference on Machine Learning, ICML’17, Sydney, Australia, 6–11 August 2017; Volume 70, pp. 2902–2911.

[25] Real, E.; Aggarwal, A.; Huang, Y.; Le, Q.V. Regularized evolution for image classifier architecture search. *Proc. Conf. AAAI Artif. Intell.* **2019**, *33*, 4780–4789.

[26] Stanley, K.O.; Clune, J.; Lehman, J.; Miikkulainen, R. Designing neural networks through neuroevolution. *Nat. Mach. Intell.* **2019**, *1*, 24–35.

[27] Wang, W.; Yu, Z.; Fu, C.; Cai, D.; He, X. COP: Customized correlation-based Filter level pruning method for deep CNN compression. *Neurocomputing* **2021**, *464*, 533–545. https://doi.org/10.1016/j.neucom.2021.08.098.

[28] Lu, Z.; Whalen, I.; Boddeti, V.; Dhebar, Y.; Deb, K.; Goodman, E.; Banzhaf, W. NSGA-Net: Neural Architecture Search Using Multi-Objective Genetic Algorithm. In Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '19, New York, NY, USA, 13–17 July 2019; pp. 419–427. https://doi.org/10.1145/3321707.3321729.

[29] Dong, X.; Liu, L.; Musial, K.; Gabrys, B. NATS-Bench: Benchmarking NAS Algorithms for Architecture Topology and Size. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *44*, 3634–3646. https://doi.org/10.1109/TPAMI.2021.3054824.

[30] Liang, T.; Glossner, J.; Wang, L.; Shi, S.; Zhang, X. Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing* **2021**, *461*, 370–403. https://doi.org/10.1016/j.neucom.2021.07.045.

[31] Jacob, B.; Kligys, S.; Chen, B.; Zhu, M.; Tang, M.; Howard, A.; Adam, H.; Kalenichenko, D. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 8–23 June 2018; pp. 2704–2713. https://doi.org/10.1109/CVPR.2018.00286.

[32] Buciluă, C.; Caruana, R.; Niculescu-Mizil, A. Model compression. In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, 20–23 August 2006.

[33] Hinton, G.; Vinyals, O.; Dean, J. Distilling the Knowledge in a Neural Network. *arXiv* **2015**, arXiv:1503.02531.

[34] Wang, L.; Yoon, K. Knowledge Distillation and Student-Teacher Learning for Visual Intelligence: A Review and New Outlooks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *44*, 3048–3068. https://doi.org/10.1109/TPAMI.2021.3055564.

[35] Liu, Z.; Sun, M.; Zhou, T.; Huang, G.; Darrell, T. Rethinking the Value of Network Pruning. *arXiv* **2019**, arXiv:1810.05270.

[36] Liu, Z.; Li, J.; Shen, Z.; Huang, G.; Yan, S.; Zhang, C. Learning Efficient Convolutional Networks through Network Slimming. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2755–2763. https://doi.org/10.1109/ICCV.2017.298.

[37] Luo, J.H.; Zhang, H.; Zhou, H.Y.; Xie, C.W.; Wu, J.; Lin, W. ThiNet: Pruning CNN Filters for a Thinner Net. *IEEE Trans. Pattern Anal. Mach. Intell.* **2019**, *41*, 2525–2538. https://doi.org/10.1109/TPAMI.2018.2858232.

[38] Molchanov, P.; Tyree, S.; Karras, T.; Aila, T.; Kautz, J. Pruning Convolutional Neural Networks for Resource Efficient Inference. *arXiv* **2017**, arXiv:1611.06440.

[39] Qi, C.; Shen, S.; Li, R.; Zhao, Z.; Liu, Q.; Liang, J.; Zhang, H. An efficient pruning scheme of deep neural networks for Internet of Things applications. *EURASIP J. Adv. Signal Process.* **2021**, *2021*, 31.

[40] Hu, H.; Peng, R.; Tai, Y.W.; Tang, C.K. Network Trimming: A Data-Driven Neuron Pruning Approach towards Efficient Deep Architectures. *arXiv* **2016**, arXiv:1607.03250.

[41] Lin, M.; Ji, R.; Wang, Y.; Zhang, Y.; Zhang, B.; Tian, Y.; Shao, L. HRank: Filter Pruning using High-Rank Feature Map. *arXiv* **2020**, arXiv:2002.10179.

[42] Liu, Y.; Fan, K.; Zhou, W. FPWT: Filter pruning via wavelet transform for CNNs. *Neural Netw.* **2024**, *179*, 106577. https://doi.org/10.1016/j.neunet.2024.106577.

[43] Wang, J.; Jiang, T.; Cui, Z.; Cao, Z. Filter pruning with a feature map entropy importance criterion for convolution neural networks compressing. *Neurocomputing* **2021**, *461*, 41–54. https://doi.org/10.1016/j.neucom.2021.07.034.

[44] He, Y.; Kang, G.; Dong, X.; Fu, Y.; Yang, Y. Soft filter pruning for accelerating deep convolutional neural networks. *arXiv* **2018**, arXiv:1808.06866.

[45] He, Y.; Liu, P.; Wang, Z.; Hu, Z.; Yang, Y. Filter Pruning via Geometric Median for Deep Convolutional Neural Networks Acceleration. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 4335–4344. https://doi.org/10.1109/CVPR.2019.00447.

[46] Wang, Y.; Zhang, X.; Xie, L.; Zhou, J.; Su, H.; Zhang, B.; Hu, X. Pruning from Scratch. *arXiv* **2019**, arXiv:1909.12579.

[47] Yeom, S.K.; Seegerer, P.; Lapuschkin, S.; Binder, A.; Wiedemann, S.; Müller, K.R.; Samek, W. Pruning by explaining: A novel criterion for deep neural network pruning. *Pattern Recognit.* **2021**, *115*, 107899. https://doi.org/10.1016/j.patcog.2021.107899.

[48] Aradhya, A.M.; Ashfahani, A.; Angelina, F.; Pratama, M.; de Mello, R.F.; Sundaram, S. Autonomous CNN (AutoCNN): A data-driven approach to network architecture determination. *Inf. Sci.* **2022**, *607*, 638–653. https://doi.org/10.1016/j.ins.2022.05.100.

[49] Hirsch, L.; Katz, G. Multi-objective pruning of dense neural networks using deep reinforcement learning. *Inf. Sci.* **2022**, *610*, 381–400. https://doi.org/10.1016/j.ins.2022.07.134.

[50] Fernandes Jr., F.E.; Yen, G.G. Pruning Deep Convolutional Neural Networks Architectures with Evolution Strategy. *Inf. Sci.* **2021**, *552*, 29–47. https://doi.org/10.1016/j.ins.2020.11.009.

[51] Li, G.; Zhang, M.; Wang, J.; Weng, D.; Corporaal, H. SCWC: Structured channel weight sharing to compress convolutional neural networks. *Inf. Sci.* **2022**, *587*, 82–96. https://doi.org/10.1016/j.ins.2021.12.020.

[52] Geng, X.; Gao, J.; Zhang, Y.; Xu, D. Complex hybrid weighted pruning method for accelerating convolutional neural networks. *Sci. Rep.* **2024**, *14*, 5570.

[53] Sharma, M.; Heard, J.; Saber, E.; Markopoulos, P.P. Convolutional Neural Network Compression via Dynamic Parameter Rank Pruning *arXiv* **2024**, arXiv:2401.08014.

[54] Jiang, P.; Xue, Y.; Neri, F. Convolutional neural network pruning based on multi-objective feature map selection for image classification. *Appl. Soft Comput.* **2023**, *139*, 110229. https://doi.org/10.1016/j.asoc.2023.110229.

[55] Liu, Y.; Wu, D.; Zhou, W.; Fan, K.; Zhou, Z. EACP: An effective automatic channel pruning for neural networks. *Neurocomputing* **2023**, *526*, 131–142. https://doi.org/10.1016/j.neucom.2023.01.014.

[56] Louati, H.; Louati, A.; Bechikh, S.; Kariri, E. Joint filter and channel pruning of convolutional neural networks as a bi-level optimization problem. *Memetic Comput.* **2024**, *16*, 71–90. https://doi.org/10.1007/s12293-024-00406-6.

[57] Jayasimhan, A.; P., P. ResPrune: An energy-efficient restorative filter pruning method using stochastic optimization for accelerating CNN. *Pattern Recognit.* **2024**, *155*, 110671. https://doi.org/10.1016/j.patcog.2024.110671.

[58] Zhang, P.; Tian, C.; Zhao, L.; Duan, Z. A multi-granularity CNN pruning framework via deformable soft mask with joint training. *Neurocomputing* **2024**, *572*, 127189. https://doi.org/10.1016/j.neucom.2023.127189.

[59] Lian, Y.; Peng, P.; Jiang, K.; Xu, W. Cross-layer importance evaluation for neural network pruning. *Neural Netw.* **2024**, *179*, 106496. https://doi.org/10.1016/j.neunet.2024.106496.

[60] Kurita, T. Principal component analysis (PCA). In *Computer Vision: A Reference Guide*; Springer: Cham, Switzerland, 2019; pp. 1–4.

[61] Golub, G.; Van Loan, C. *Matrix Computations*, 3rd, ed.; JHU Press: Baltimore, MD, USA, 1996.

[62] Frobenius Norm—from Wolfram MathWorld. Available online: https://mathworld.wolfram.com/FrobeniusNorm.html (accessed on 25 April 2023).

[63] Everitt, B.; Skrondal, A. *The Cambridge Dictionary of Statistics B.S. Everitt Aut; A. Skrondal Aut*; Cambridge University Press: Cambridge, UK, 2011.

[64] He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016.

[65] Huang, G.; Liu, Z.; van der Maaten, L.; Weinberger, K.Q. Densely Connected Convolutional Networks. *arXiv* **2018**, arXiv:1608.06993.

[66] Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv* **2017**, arXiv:1704.04861.

[67] Recht, B.; Roelofs, R.; Schmidt, L.; Shankar, V. Do ImageNet Classifiers Generalize to ImageNet? In Proceedings of the 36th International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; Volume 97, pp. 5389–5400.

[68] imagenet_v2|TensorFlow Datasets—tensorflow.org. 2022. Available online: https://www.tensorflow.org/datasets/catalog/imagenet_v2 (accessed on 10 May 2023).

[69] Tensorflow. *Image Data Generator v2.14.1*; Tensorflow. 2022. Available online: https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator (accessed on 20 May 2023).

[70] Guillemot, M.; Heusele, C.; Korichi, R.; Schnebert, S.; Chen, L. Breaking Batch Normalization for better explainability of Deep Neural Networks through Layer-wise Relevance Propagation **2020**. [arXiv:cs.LG/2002.11018].

[71] Lin, S.; Ji, R.; Li, Y.; Deng, C.; Li, X. Toward Compact ConvNets via Structure-Sparsity Regularized Filter Pruning. *IEEE Trans. Neural Networks Learn. Syst.* **2020**, *31*, 574–588. https://doi.org/10.1109/TNNLS.2019.2906563.

[72] Ravi Ganesh, M.; Blanchard, D.; Corso, J.J.; Sekeh, S.Y. Slimming Neural Networks Using Adaptive Connectivity Scores. *IEEE Trans. Neural Networks Learn. Syst.* **2024**, *35*, 3794–3808. https://doi.org/10.1109/TNNLS.2022.3198580.

[73] Zhang, Y.; Lin, M.; Lin, C.W.; Chen, J.; Wu, Y.; Tian, Y.; Ji, R. Carrying Out CNN Channel Pruning in a White Box. *IEEE Trans. Neural Networks Learn. Syst.* **2023**, *34*, 7946–7955. https://doi.org/10.1109/TNNLS.2022.3147269.

[74] Liu, G.; Zhang, K.; Lv, M. SOKS: Automatic Searching of the Optimal Kernel Shapes for Stripe-Wise Network Pruning. *IEEE Trans. Neural Networks Learn. Syst.* **2023**, *34*, 9912–9924. https://doi.org/10.1109/TNNLS.2022.3162067.

[75] He, Y.; Zhang, X.; Sun, J. Channel Pruning for Accelerating Very Deep Neural Networks. *arXiv* **2017**, arXiv:1707.06168.

[76] Yvinec, E.; Dapogny, A.; Cord, M.; Bailly, K. RED++ : Data-Free Pruning of Deep Neural Networks via Input Splitting and Output Merging. *IEEE Trans. Pattern Anal. Mach. Intell.* **2023**, *45*, 3664–3676. https://doi.org/10.1109/TPAMI.2022.3179616.

[77] Ding, G.; Zhang, S.; Jia, Z.; Zhong, J.; Han, J. Where to Prune: Using LSTM to Guide Data-Dependent Soft Pruning. *IEEE Trans. Image Process.* **2021**, *30*, 293–304. https://doi.org/10.1109/TIP.2020.3035028.

[78] Liu, J.; Zhuang, B.; Zhuang, Z.; Guo, Y.; Huang, J.; Zhu, J.; Tan, M. Discrimination-aware Network Pruning for Deep Model Compression. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *44*, 4035–4051. https://doi.org/10.1109/TPAMI.2021.3066410.

3 A Green AI Methodology Based on Persistent Homology for Compressing BERT

- Balderas, L., Lastra, M., & Benítez, J. M. (2025). A Green AI Methodology Based on Persistent Homology for Compressing BERT. *Applied Sciences*, 15(1), 390. https://doi.org/10.3390/app15010390
  - Status: **Published**.
  - Impact Factor (JCR 2023): **2.5**
  - Subject Category: **Engineering, Multidisciplinary**
  - Rank: **44/181**
  - Quartile: **Q1**

A GREEN AI METHODOLOGY BASED ON PERSISTENT HOMOLOGY FOR COMPRESSING BERT

Luis Balderas
Department of Computer Science
and Artificial Intelligence
DiCITS, iMUDS, DaSCI
University of Granada, Granada, Spain 18071
luisbalru@ugr.es

Miguel Lastra
Department of Software Engineering
DiCITS, iMUDS, DaSCI
University of Granada, Granada, Spain 18071
mlastral@ugr.es

José M. Benítez
Department of Computer Science
and Artificial Intelligence
DiCITS, iMUDS, DaSCI
University of Granada, Granada, Spain 18071
J.M.Benitez@decsai.ugr.es

ABSTRACT

Large Language Models (LLMs) like BERT have gained significant prominence due to their remarkable performance in various natural language processing tasks. However, they come with substantial computational and memory costs. Additionally, they are essentially black-box models, challenging to explain and interpret. In this article, Persistent BERT Compression and Explainability (PBCE) is proposed, a Green AI methodology to prune BERT models using persistent homology, aiming to measure the importance of each neuron by studying the topological characteristics of their outputs. As a result, PBCE can compress BERT significantly by reducing the number of parameters (47% of the original parameters for BERT Base, 42% for BERT Large). The proposed methodology has been evaluated on the standard GLUE Benchmark, comparing the results with state-of-the-art techniques achieving outstanding results. Consequently, PBCE can simplify the BERT model by providing explainability to its neurons and reducing the model's size, making it more suitable for deployment on resource-constrained devices.

The code implementing the method will be available at https://github.com/luisbalru/PBCE.

***K*eywords** BERT compression; Green AI; persistent homology; neural networks explainability

1 Introduction

Over the last years, Large Language Models (LLM) like BERT (Bidirectional Encoder Representations from Transformers) [14] have achieved phenomenal results in a wide variety of natural language processing (NLP) tasks. These models, pre-trained on large amounts of data, are available for use by the scientific community, either as a semantic analytical tool or to build fine-tuned solutions for specific problems.

However, these models suffer from two weaknesses. On one hand, they are computationally and memory-intensive, making it challenging to deploy them on devices with limited resources. As a result, these advancements run counter to the Green AI [7, 40] paradigm, which is dedicated to developing AI technologies that minimize computational costs and place equal emphasis on efficiency and predictive accuracy. On the other hand, they are black-box models: due to the large number of neurons, layers, parameters, and data transformations (e.g., in Attention layers [43]), it is practically impossible to explain the internal state of the network at any given moment and provide interpretability to the final output.

In this article, Persistent BERT Compression and Explainability (PBCE) is proposed, a Green AI methodology based on homology theory to compress the BERT model and give insights about the importance of its neurons. BERT has been chosen as the reference model given the fact that is one of the foundational models that laid the groundwork for LLM encoders. Several subsequent BERT-based encoder models, such as RoBERTa [30], DistilBERT [39], ALBERT [24] or TinyBERT [23], have been developed to address specific limitations of the original. PBCE is focused on the internal representation of the neural network, considering the individual role of each neuron when the model makes an inference. In particular, PBCE uses zero-dimensional persistent homology, extracting the topological characteristics of the neurons' outputs and providing an assessment of their importance within the network. This way, it is feasible to identify neurons that contribute with more information to the overall computation of the network and pruning those that contribute less. As a consequence, the proposed technique becomes an effective method for compressing BERT.

To the best of our knowledge, this is the first approach that uses persistent homology to compress and give insights of the importance of the units of the BERT model. To measure the effectiveness of PBCE, an extensive experimentation has been designed based on natural language processing tasks proposed in GLUE (General Language Understanding Evaluation) benchmark [45], achieving results that outperform other state-of-the-art techniques for BERT compression. The main purpose of the experimentation is to provide an answer to the following research questions (RQ):

(RQ1:) How effective and robust is persistent homology at measuring the importance of neurons in a transformer encoder model such as BERT?

(RQ2:) Is it feasible to propose a practical methodology using persistent homology for simplifying BERT-based models?

(RQ3:) Can persistent homology be employed to enhance the explainability of language models like BERT?

The main contributions of the article can be summarized as follows:

- A methodology has been developed for compressing the BERT model by analyzing the topological features of the outputs of each neuron. This can be understood as explainability in terms of the individual role of each neuron.

- This methodology is applied to two versions of the BERT model, interpreting the topological characteristics as a tool to assess the importance of neurons, simplifying those that contribute less information, and generating a pruned version of the BERT model.

- The performance of the simplified models has been evaluated using the GLUE Benchmark and compared the results with other state-of-the-art compression techniques, demonstrating the effectiveness of PBCE for model explicability and compression.

The rest of this paper is structured as follows: In Section 2, the state-of-art of different approaches using persistent homology and deep learning models to solve problems and BERT compression techniques is introduced. In Section 3, PBCE is described. In Section 4 the methodology is experimentally analyzed and discussed. Section 5 highlights the conclusions.

2 Previous works

In this section, the most relevant articles from the state of the art related to PBCE are presented. To the best of our knowledge, there is no technique in the literature that uses persistent homology to prune neural networks. Therefore, techniques that utilize homology theory and deep learning to solve scientific problems are presented first. Finally, pruning techniques for LLMs are introduced, which will serve as a reference to validate the proposed methodology.

2.1 Persistent homology applied to machine learning problems

In recent years, numerous machine learning methods based on persistent homology, which is a mathematical method used in topological data analysis to study features of data, have been proposed. Some of them involve feature extraction through the analysis of persistence diagrams (Birth-Death diagrams) and persistence barcodes. Additionally, there are methods which establish similarity metrics between barcodes [32]. From a mathematical perspective, solutions based on persistent homology have been presented to regularize the internal representation of deep neural networks applied to image classification [10], [12].

Persistent homology, combined with machine learning, also finds applications in the fields of biology and chemistry, particularly in protein analysis. Deep neural networks are not directly applicable to molecular data, as they are characterized by complex three-dimensional structures. Therefore, the use of topological representations and features is essential for solving protein classification problems [36]. They are also highly useful for identifying structures and functions in a protein sequence [38], for automatic protein annotation [33], or in chemistry, for tasks like simultaneous prediction of partition coefficients and aqueous solubility [49].

Getting closer to the topic of the article, homology theory has been used to analyze natural language. Specifically, in [37], TopoBERT is presented as a visual tool to explore the fine-tuning process of different language models from a topological perspective. Besides, it facilitates the visualization of the shape of embedding spaces and the linguistic and semantic connection between the input dataset and the topology of its embedding space.

2.2 Brief description of the BERT model

BERT, an acronym for “Bidirectional Encoder Representations from Transformers", [14] is a language model developed by Google that has revolutionized natural language processing. Its primary innovation lies in its ability to comprehend a word’s context within a text by analyzing all surrounding words, both to the left and right. Unlike previous language models, which were unidirectional and predicted words based on previous ones, BERT takes the entire context into account.

BERT is built upon the Transformer architecture, a deep neural network autoencoder. The Transformer architecture employs multi-head attention to process sequences. Before the network processes the information, the texts are tokenized, meaning texts are divided into a set of tokens that represent the fundamental semantic content of the word and are translated into their numerical position in the model’s vocabulary. Additionally, special tokens are added, such as [CLS], marking the start of a sentence; [SEP], marking the end, and [PAD], signifying the end of padding. Some research papers from the literature, such as [13], show that the [CLS]

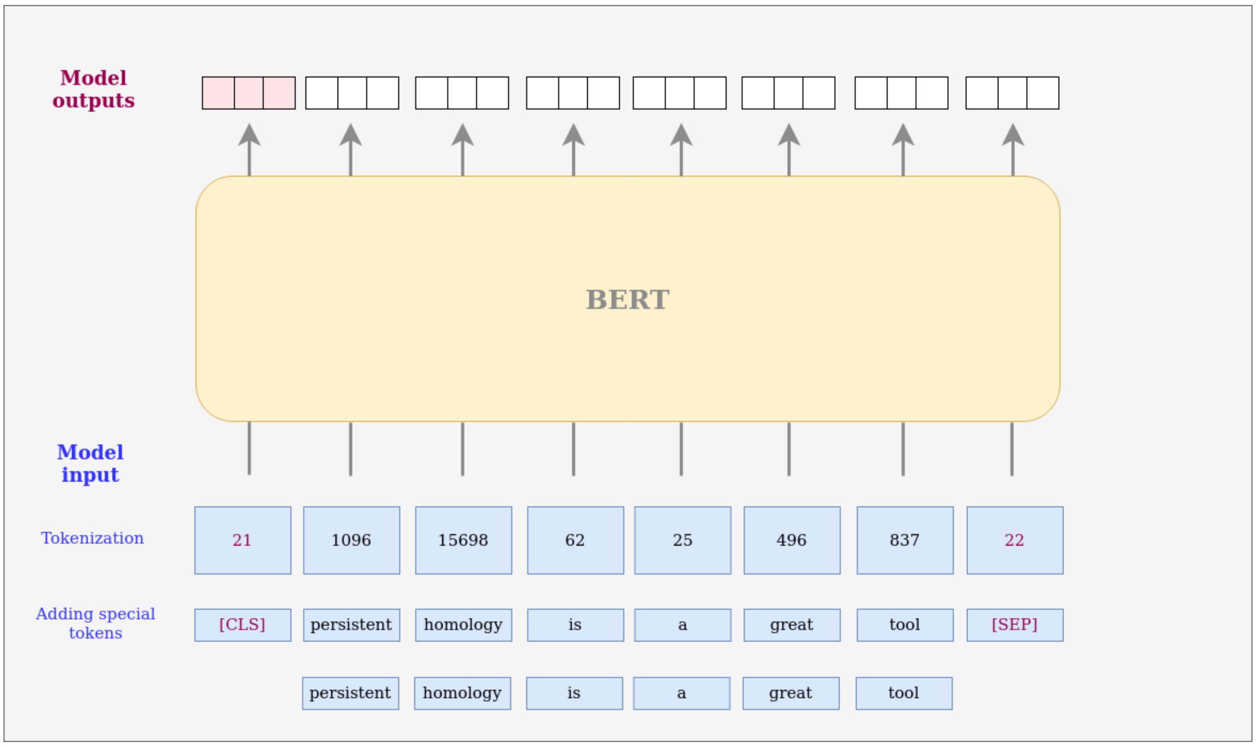


Figure 1: Usage of BERT involves taking a sentence, adding the special tokens [CLS] and [SEP], tokenizing the words, and using these tokens as input for the neural network.

token, besides marking the beginning of a sequence, provides an aggregated representation of the input. In fact, it is used in text classification tasks and sentiment analysis. Figure 1 shows an example of how sequences are generated to be a valid input for BERT model.

To pre-train and fine-fune BERT, an extensive amount of text was used. During the training process, BERT learns to predict hidden words in complete sentences, enabling it to better understand how words relate to each other in a given context.

This pre-trained model can be fine-tuned for specific tasks such as text classification, entity tagging, machine translation, and more. Due to its context-awareness, BERT has outperformed many records in various natural language processing tasks and has become an essential tool in this field.

Each BERT Layer, as part of the Encoder of the Transformers architecture, consists of three components: the Multi-Head Attention layer, comprised of the Q (query), K (key), V (value) matrices, and Attention Output; the intermediate layer; and the output layer. Figure 2 shows the complete representation of the BERT architecture. Besides, more detailed information can be found in [14] and [43].

Let L be the number of layers, H the hidden size, and A the number of self-attention layers of the model, there are two implementations of the BERT model: the BERT Base Cased model [2] ($L = 12$, $H = 768$, $A = 12$, total number of parameters $= 110M$) and the BERT Large Cased model [3] ($L = 24$, $H = 1024$, $A = 16$, total number of parameters $= 340M$).

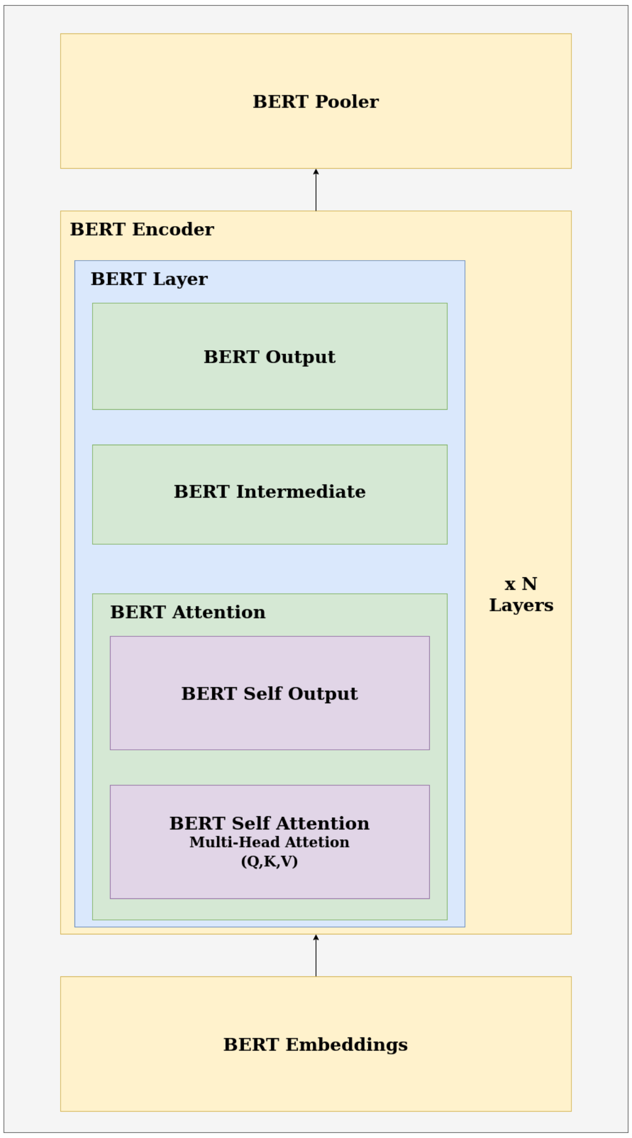


Figure 2: Representation of the BERT architecture. It is composed of an embedding module, followed by the Encoder part, which consists of N BERT Layers (12 or 24, depending on whether it's BERT Base or Large). Within it, three main components stand out: the Attention layer, the Intermediate layer, and the Output layer. After the Encoder, BERT has a Pooler layer.

2.3 BERT model pruning methods

As mentioned earlier, to the best of our knowledge, there are no pruning methods for LLMs based on persistent homology. However, there are compression algorithms for the BERT architecture that provide competitive results [19]. These methods can be divided into structured and unstructured approaches [16].

Among structured methods, some of them focus on pruning attention heads exclusively. In [25] a regularization method for BERT is proposed using reinforcement learning to prune attention heads. For text classification, a compression method based on adaptive truncation is presented in [51]. In [31], a forward and backward pass is used to calculate gradients, which are used as an importance score for each attention layer. [44] suggests constructing a loss function that minimizes both classification error and the number of heads used, pruning unproductive ones while maintaining performance. [21] presents a technique with three stages: in the first, n pruning strategies are generated with the same pruning ratio. In stage II, n candidates from the training set are evaluated, and after all iterations, the one that performs the best on a subnetwork is chosen as the best candidate. This best candidate undergoes fine-tuning to obtain a good subnetwork. In [53] a new sentence-level feature alignment loss distillation mechanism for BERT compression is introduced. It is guided by mixture-of-experts to capture contextual semantic knowledge from the teacher model to the student model while reducing its parameters. In [52] a dynamic structure pruning method based on differentiable search and recursive knowledge distillation technique called DDK is presented. It is focused on pruning all feed-forward and self-attention layers. Additionally, a recursive knowledge distillation method is preseted designed to extract the most important features from the intermediate layers. Another uncertainty-driven knowledge distillation technique is presented in [22]. The uncertainty modeling guides the training process effectively, especially when there is a large gap in network performance between pretrained language models and compressed ones. In [28], a Layer-wise Adaptive Distillation (LAD) method is proposed for model compression. Specifically, an iterative aggregation mechanism is designed to distill layer-wise internal knowledge from the teacher model to the student model. Finally, in [50], a novel approach to compress BERT called You Only Compress Once-BERT (YOCO-BERT) is introduced. YOCO-BERT constructs a search space with all the possible configurations for BERT model. Then, by using an stochastic nature gradient optimization method an optimal candidate architecture is generated.

Among unstructured methods, [11] uses unstructured magnitude pruning to find subnetworks with sparsity levels between 40% and 90%. It concludes that those with 70% sparsity, using the masked language modeling task, are universal and can solve other tasks without losing accuracy. [18] proposes a weight pruning algorithm that integrates reweighted L1 minimization with a proximal algorithm. More information on pruning algorithms for LLMs is available in [19].

Quantization algorithms have also been proposed, such as [41], which uses second-order Hessian matrix information for quantizing BERT models to low precision. Hardware-based techniques have been proposed too [26]. In [35], a sensitivity-aware mixed precision quantization method called SensiMix is proposed, which applies a 1-bit quantization to insensitive parts of the model. Finally, there are knowledge distillation-based algorithms, like the one presented in [22], which is based on parameter retention and feed forward network parameter distillation.

3 Our proposal

In this article, Persistent BERT Compression and Explainability (PBCE) is proposed, a novel Green AI technique to compress BERT-based models through the application of homology theory, allowing to derive a simplified yet effective version of the model. Specifically, using the BERT architecture as a reference, persistent homology is employed as a fundamental tool for analyzing the topological characteristics of the neurons, drawing conclusions to discard those that do not make a relevant contribution to the model. In this section, firstly, an intuitive geometric description of how PBCE applies zero-dimensional persistent homology to the vectors generated by hidden layers of a neural network is presented. Finally, the proposed methodology for compressing the model is presented.

3.1 An intuitive geometric description of persistent homology applied to LLM explanations

Homology theory is a branch of algebraic topology increasingly used in data science. In particular, persistent homology is a powerful tool for studying patterns in data. Its mathematical foundation, summarized in Appendix A, is deep and complex with a significant algebraic burden. However, it can be given a much more intuitive geometric interpretation, which will be explained below.

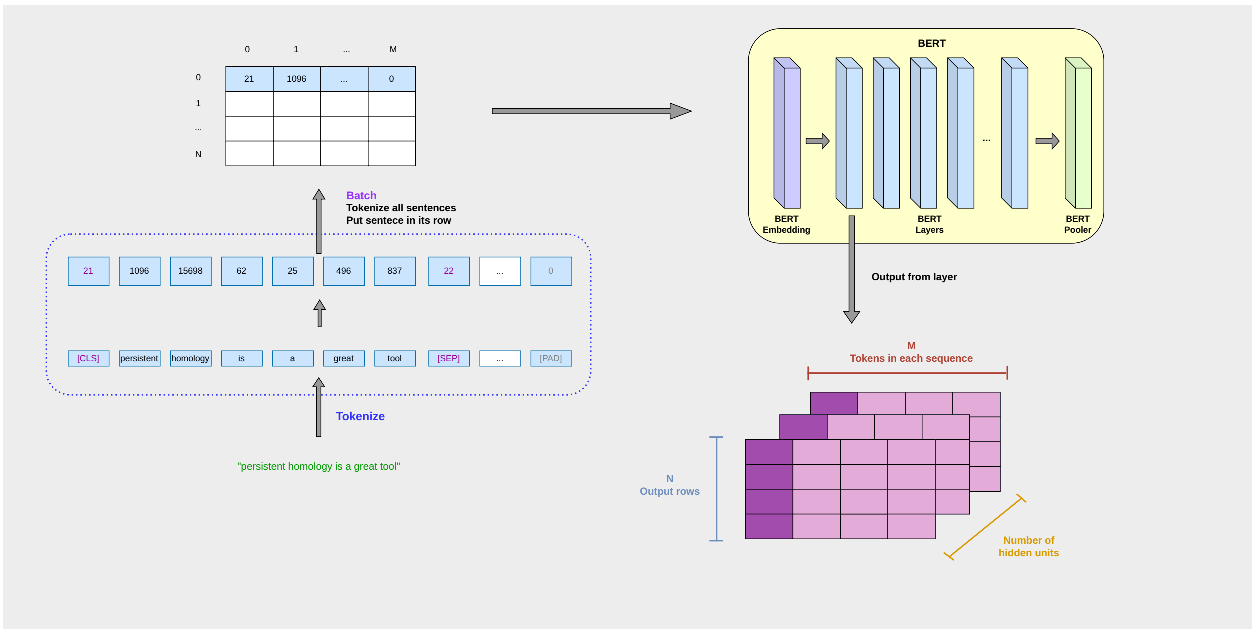


Figure 3: Representation of information through the BERT model and subsequent extraction of values from intermediate dense layers. The process begins with the processing of a set of elements from a corpus. Once the input of N sentences with length M is constructed, it is fed into the neural network. In the lower right part of the image, the output of any dense layer is represented. The output is a three-dimensional matrix: the number of sentences in the input (N), the length of those tokenized sentences (M), and the number of hidden neurons in that layer. This information, in the form of an n-dimensional matrix, is taken to represent the information associated with each neuron. To analyze it, persistent homology is used.

In Figure 3, a complete example of data processing in BERT can be found. Starting with a set of elements from a corpus, after being tokenized, they are fed into the network. The output of any neuron in a hidden layer, which serves as input for the next layers, consists of vectors that, in an abstract sense, could be represented as points in a hyperspace. The geometric distribution of these points can provide meaningful information about the behavior and role that the corresponding neuron plays within the data flow of the network. Zero-dimensional persistent homology helps explain this role.

As it can be seen in Figure 3, each BERT layer generates multidimensional vectors of size $N \times M \times NHU$, being N the number of output rows, M the number of tokens in each sequence and NHU the number of hidden units. These vectors represent the internal state of texts used as inputs. For the sake of clarity, a simple two dimensional example is introduced in Figure 5. Two plots at three different time moments can be seen in that figure. The plot on the left depics some vectors generated by any layer (as explained earlier), which are going to be converted into the centers of disks. The disks will evolve, growing uniformly with a common radius. The value of the radius is marked by the red line on the right (Birth-Death diagram). Notice that this red line will move upwards on the Y-axis of the graph on the right.

Since PBCE is based on zero-dimensional persistent homology, each output vector becomes a singleton connected component and all originate at time zero (value 0 on the Bith-Death diagram). In consequence, there are as many connected components as output vectors. As the radius of the disks grows, the connected components eventually collapse. When two connected components touch, they merge becoming a single area (a single component) and a blue point is marked on the Birth-Death plot at the radius value that led to the merger. This point on the Birth-Death plot implies the death of a connected component. Remarkably, the radius grows until the last two connected components touch, creating a large single connected component that includes all the vectors. Let's call r_f the radius value at which all connected components collapse. Please, note that this is the smallest value for the radius at which all the connected componets are glued together. Figure 4 shows an example of persistence diagram in which $r_f = 0.20229639$.

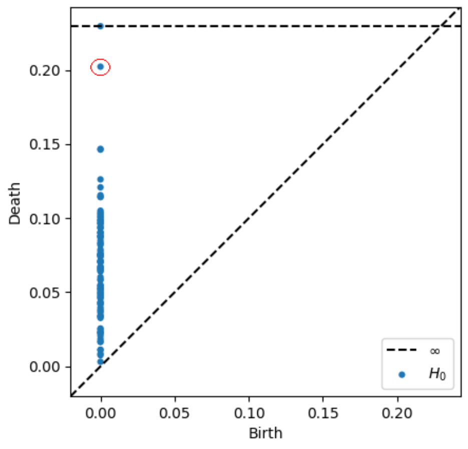


Figure 4: [Diagram: X-axis (Birth Time), Y-axis (Death Time. Persistence)] Here is an example of a Birth-Death persistence diagram. The points where connected components are born are presented on the X-axis. Since zero-dimensional persistent homology is used, all connected components are born at time zero. As the value of r increases (Y-axis), the connected components collapse. Each time two components collapse, a point is represented. The last value below the dashed line corresponds to r_f (circled in red).

The r_f value will be fundamental in the analysis because it provides information about the distribution of output vectors from each neuron and their variability. This helps assess whether the neuron has very uniform outputs (providing little information) or exhibit variability in its outputs (providing more information). Concretely, if r_f is small, it means that the different outputs of a neuron are very similar to each other. However, if r_f is large, then the outputs are less redundant. All in all, this procedure allows to "measure" the variability in the output distribution of a neuron. It can be described through a simple geometric procedure. Nevertheless, it has a founded mathematical background.

3.2 PBCE: using persistent homology to compress BERT

In this article, PBCE is proposed, a methodology aimed at compressing BERT models. PBCE uses zero-dimensional persistent homology to analyze the topological characteristics of neuron outputs for each layer, aiming to identify which of them play a more significant role in the information flow and, consequently, in the model's decision-making process. This allows for the removal of neurons that provide less information, compressing the network and making it more efficient. Algorithm 1 outlines the proposed methodology. In summary, the methodology starts by selecting a corpus. Next, zero-dimensional persistent homology is used as a tool to measure the importance of neurons. Then, using the persistence diagram, it can be found which neurons can be removed. Finally, the simplified model is built and evaluated on the GLUE Benchmark.

Next, a detailed description of each of the steps outlined in Algorithm 1 can be found.

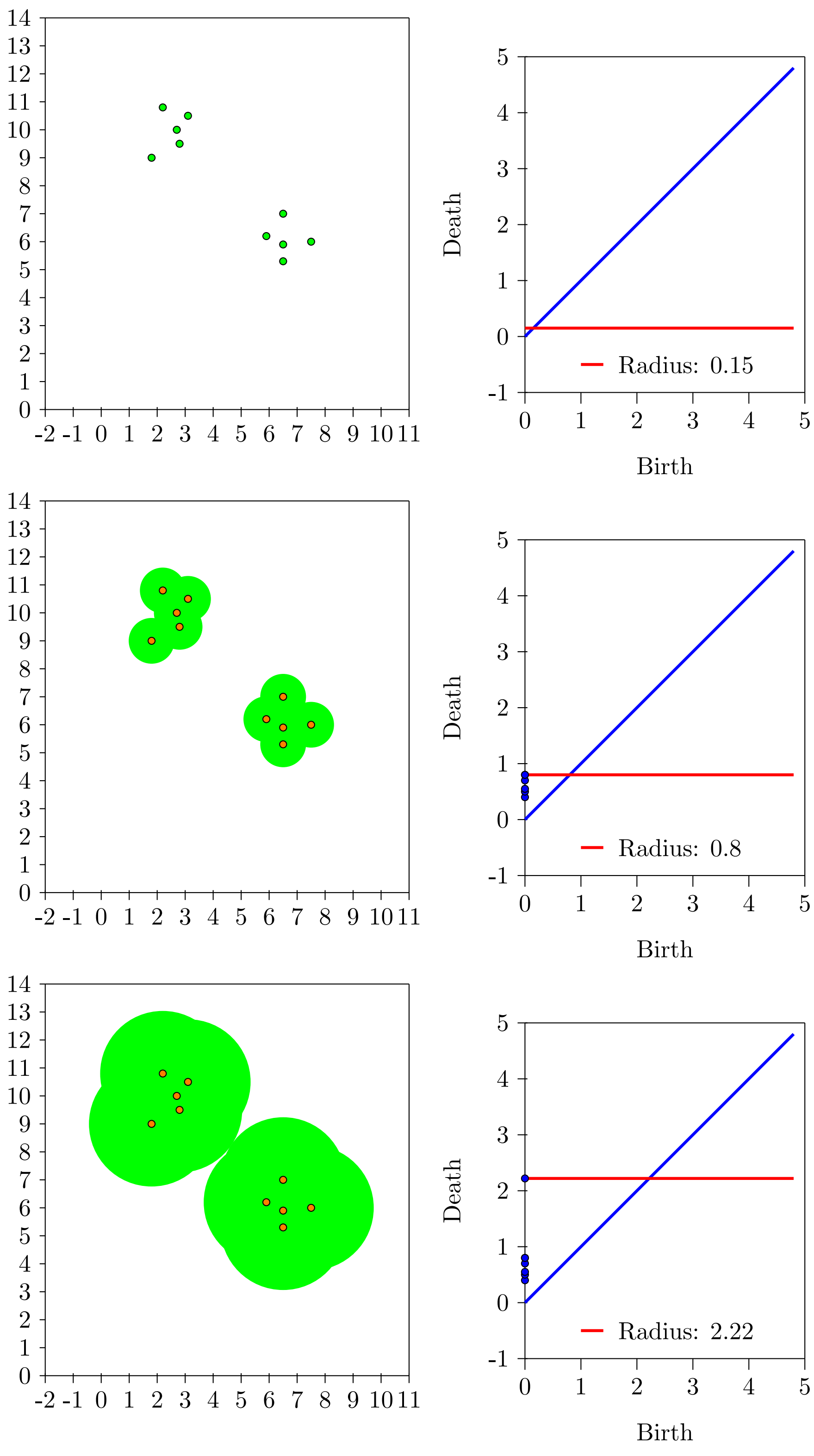


Figure 5: Application of persistent homology on the output of a neuron at three specific moments. On the left, it can be seen that each of the points comprising the output becomes the center of a disk whose radius grows uniformly for all points. On the right, the Birth-Death diagram is represented for persistent homology of dimension zero. Each blue point corresponds to the disappearance of a connected component after collapsing with another. The last moment depicted in the figure represents the point at which the value of r is reached for which all connected components first merge. This value is called r_f , and it is crucial in the proposed methodology because it provides information about the importance of neurons based on their output within the neural network's data flow.

Algorithm 1 PBCE: BERT compression through zero-dimensional persistent homology

```
1) Select the corpus to carry out the analysis.
2) Use the zero-dimensional persistent homology and the persistence diagram for the outputs of each unit
and layer evaluated on the corpus.
3) Analyze the distribution of $r_f$ from the persistence diagram. Select the units to be removed.
4) Construct the simplified model.
5) Evaluate the simplified model with the GLUE Benchmark.
```

3.2.1 Corpus selection

To measure the importance of each unit in the BERT model, an extensive text corpus is selected. Specifically, English Wikipedia [4] is chosen, consisting of more than 20GB of sanitized text, including all entries from the Wikipedia in the mentioned language. To ensure that the texts, after tokenization, adhere to the input size constraint imposed by the model, the entries are splitted by periods, generating sentences, which will be the definite input for PBCE.

This corpus is suitable because it contains a large number of texts on diverse topics, all written with high quality. Moreover, it is not directly related to any of the tasks in the GLUE benchmark, which will be used to evaluate the performance of the simplified models.

3.2.2 Using persistent homology to analyze BERT Layer outputs

Once the corpus has been selected for analysis, persistent homology is used to collect and analyze the topological characteristics expressed by each neuron in the network. Considering the output vectors generated by each neuron, the connected components are constructed. These connected components represent the clusters of data points that are topologically connected in some way, and their persistence features helps reveal meaningful hidden structures in the data. This information is crucial for understanding the topological features that each neuron contributes to the model. As it can be seen in Figure 5, the connected components evolve as the radius r grows, until all collapse into one. The least value of r which provokes the union of all the connected components, denoted as r_f, is crucial in the proposed methodology. The larger r_f, the farther are neuron's outputs from one other, meaning they exhibit a higher variability. Conversely, smaller values of r_f suggest that the neuron is less relevant, as its outputs are very close in the metric space generated by the unit and can thus be removed. In the case the neuron is removed, to avoid losing all the information generated by the neuron, the mean plus the standard deviation of the outputs is taken and added to the bias of the layer. This way, the contribution of the eliminated neuron is implicitly considered in the network's inference process.

As mentioned in the previous section, the persistence diagram is used to analyze the evolution of the connected components. Since PBCE focuses on zero-dimensional persistent homology, the points in the diagram align along a line parallel to the Y-axis, since all the points share the abscissa (given the fact that all connected components are born simultaneously). Therefore, the values on the Y-axis are only considered. r_f can be readily identified in the persistence diagram as the value of r for which all connected components first merge.

3.2.3 Evaluation of r_f distribution and selection of the important units

Once the value of r_f for each neuron has been obtained, the analysis of the distribution of these values begins. This helps understand which unit can potentially be suppressed. To facilitate the simplification task, three levels of pruning are established:

1. The first level, which is the lightest, involves calculating the first quartile (Q1) of the r_f values and retaining neurons with r_f values higher than Q1.
2. The second level is slightly more severe, applying the same operation but with the second quartile (Q2).
3. The most intense pruning is the third level, where only the neurons with r_f values higher than the third quartile (Q3) are kept.

The more information a unit provides, the higher its chance to remain in the model. This way, the most influential neurons are retained and the less relevant ones are removed.

3.2.4 Evaluation of the compressed model through the GLUE Benchmark

The evaluation of the simplified model through the GLUE (General Language Understanding Evaluation) benchmark involves assessing the model's performance on a set of diverse natural language understanding tasks. GLUE is a benchmark that consists of multiple downstream NLP tasks, such as text classification, sentence similarity, and question answering. It serves as a standard evaluation suite for assess the generalization and performance of language models.

Here is how the evaluation process generally works:

- **Fine-tuning process:** The simplified model is fine-tuned on the GLUE benchmark tasks. The tasks are MNLI [48], QQP [46], QNLI [5], SST-2 [42], CoLA [47], STS-B [8], MRPC [1] and RTE [6].
- **Model Evaluation:** Use the fine-tuned simplified model to make predictions on the GLUE benchmark tasks. For each task, the model will generate predictions.
- **Evaluation Metrics:** Calculate task-specific evaluation metrics for each GLUE task. These metrics can vary depending on the task but often include accuracy, F1 score, or other relevant measures. GLUE provides a standard evaluation script for each task (Table 1).
- **Comparison:** Compare the performance of the simplified model to the performance of the original, more complex BERT model and other simplified approaches from the literature. This will give an indication of how much simplification impacted the model's ability to perform various NLP tasks.

The GLUE benchmark provides a standardized way to assess the trade-off between model complexity and task performance. It helps determine if the simplified model retains sufficient performance on a range of NLP tasks while being computationally more efficient than other BERT models.

4 Empirical evaluation

To assess the performance of PBCE, a thorough empirical procedure has been designed that involves a wide range of natural language tasks included in the GLUE benchmark through their datasets and specific metrics (MNLI, QQP, QNLI, SST-2, CoLA, STS-B, MRPC, and RTE, as listed in Table 1); and both architectures of the BERT model: BERT Base and BERT Large. This represents a benchmark commonly used in the literature, as shown in [34], [27], [9]. Additionally, PBCE has been compared with the state-of-art BERT compression techniques.

This section presents and analyzes the experimental results. First, the distribution of r_f values and their role in identifying significant neurons for prediction are examined. Subsequently, specific results for simplifying both BERT Base and BERT Large models are presented, addressing research questions RQ1 and RQ2.

Table 1: GLUE tasks, train and evaluation sizes and metrics

Task	Train	Evaluation	Metric
CoLA	10K	1K	Matthew's Correlation
SST-2	67K	872	Accuracy
MRPC	5.8K	1K	F1/Accuracy
STS-B	7K	1.5K	Pearson-Spearman Correlation
QQP	400K	10K	F1/Accuracy
MNLI	393K	20K	Accuracy
QNLI	108K	11K	Accuracy
RTE	2.7K	0.5K	Accuracy

4.1 Distribution of r_f and selection of the more informative neurons

The analysis of the distribution of r_f is crucial for correctly identifying which neurons contribute the most information to a neural network's predictions and which units are dispensable. The premise here is that a neuron with a high associated r_f value implies that its outputs exhibit sufficient variability, making it important when the network computes outputs.

Bearing in mind that the mean can be heavily influenced by excessively high r_f values in some situations, the median was chosen as the key metric to consider. Figure 6 shows the median of the r_f values for each layer and component of BERT Base (upper image, 6a) and BERT Large (lower image, 6b). The Figure helps determine the pruning intensity to apply to each component and layer of the model. By analyzing the median of the r_f values of the neurons within that component, PBCE quantifies the importance of each component and, consequently, the extent to which it should be pruned. The experimentation revealed that components with r_f values above 0.2 will receive light pruning, values between 0.1 and 0.2 should receive medium pruning, and values below 0.1 should undergo intensive pruning. Nevertheless, it is crucial to asses the optimal cut-off value for each specific case to ensure effective pruning.

Before delving into the details of simplification for each architecture, some considerations that apply to both BERT Base and BERT Large should be taken into account. Firstly, PBCE starts the simplification process from the third layer on. This decision minimally reduces compression capacity but does favor subsequent fine-tuning to meet the requirements of the GLUE benchmark tasks. In addition, Attention Ouput and Output components' implementation involves a LayerNorm operation in which the state of the data and the original input are added together. In consequence, these components cannot be simplified and will be excluded of the analysis. Besides, it is observed that the behavior of the Q and K components for each *BertLayer* is very similar, so the same pruning process will be applied to them. Additionally, it has been verified that the component that contributes the most information, according to our hypothesis, is the Intermediate component. Table 2 and Table 3 present the distribution of levels of pruning to each component and layer. For example, for the Q component, layers 3, 4 and 12 are lightly pruned (retaining neurones with r_f values higher than the first quartile), layers from 8 to 11 are pruned with medium intensity (retaining the second quartile of neurons) and layers 5, 6 and 7 are severely pruned (retaining the third quartile of neurons). Next,the main ideas derived from Figure 6 are explained.

In the case of BERT Base, there is an increasing trend in the contribution of neurons, with outputs becoming more variable as they approach the final layer. This trend experiences an abrupt disruption in layers 3 and 4, where all components exhibit a significant high value for the median of their r_f values. In consequence, as shown in Table 2, the third and fourth layers are lightly pruned for all components, except for the V component, in which a medium pruned is applied to the third layer. In contrast, layers from 5 to 7 are the less relevant for each component. Besides, the Intermediate component is the most meaningful one. As a result, none of its layers is heavily pruned.

In contrast, in the case of BERT Large, the median of the r_f values across the layers does not exhibit a consistent growth pattern. As observed, between layers 3 and 11, all components show increasing values, contributing more information over time. However, there are significant drops in the median of r_f for the Q, K, and V components between layers 12 and 13, as well as between layers 17 and 18. During these same layers, the Intermediate component experiences local maxima. In general, the values are not very high, and the Intermediate component continues to provide a significant amount of information compared to the others, except for layers 15, 16, and 17, where the Q and K components are more informative.

Table 2: Levels of pruning applied to each component and layer for the BERT Base architecture. The columns Q1, Q2, and Q3 indicate in which layers light, intermediate, or intense pruning is performed based on their r_f values. Note that the V component tends to contribute less information (most of its layers are heavily pruned), while the Intermediate component provides a significant amount of information (none of its layers are pruned heavily, and most are pruned lightly). For more details, please refer to Figure 6

BertLayer Component	**Q1**	**Q2**	**Q3**
Q	3, 4, 12	8-11	5-7
K	3, 4, 12	8-11	5-7
V	4	3, 11, 12	5-10
Intermediate	3, 4	5-12	-

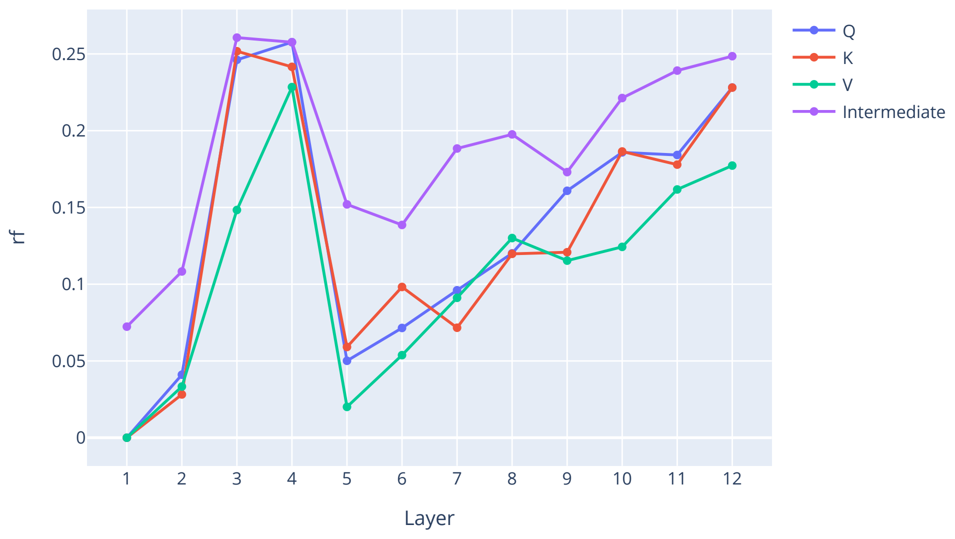
Table 3: Levels of pruning applied to each component and layer for the BERT Large architecture. The columns Q1, Q2, and Q3 indicate in which layers light, intermediate, or intense pruning is performed based on their r_f values. As in the case of BERT Base, the Intermediate component proves to be relevant for the network's prediction in the BERT Large architecture as well. For more details, please refer to Figure 6

BertLayer Component	**Q1**	**Q2**	**Q3**
Q	11,12, 14-17	4-10, 18	3, 13, 19-24
K	11, 14-17	4-10, 12, 18, 19	3, 13, 20-24
V	-	4, 8-12, 14-17	3, 5-7, 10, 13, 18-24
Intermediate	3, 4, 8-16, 18	5-7, 17, 19-24	-

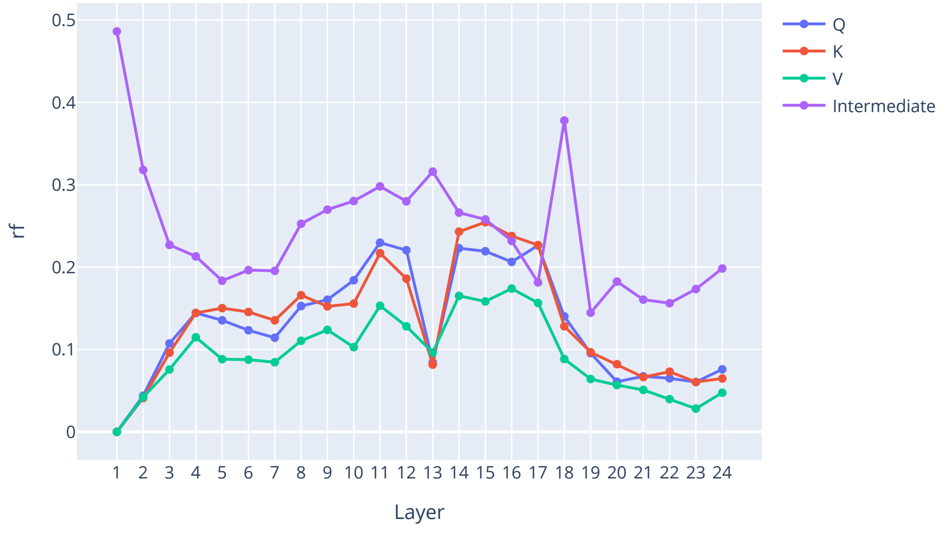
4.2 Results and analysis on the BERT Base model

The results for BERT Base are presented in Table 4. In this table, each column corresponds to a task, and each row represents a compression technique. Thus, for each row, the result obtained by each technique is expressed in the metric corresponding to the task (Table 1). Results are reported according to the common practices in the literature.

Notice that most state-of-the-art techniques do not perform a complete evaluation on the tasks in the GLUE Benchmark. In this work, a thorough study of these tasks has been conducted, reducing the model to 47% of the original parameters (from 110M to 52M). PBCE achieves better results in all the learning tasks compared to state-of-the-art techniques except for SST-2 and RTE, although it obtains competitive results. Even with a significant reduction in the number of parameters, it manages to improve the performance of the original model in most tasks, such as QQP (+20), QNLI (+1.2), CoLA (+10.86), STS-B (+3.72), MRPC (+2.19), and RTE (+5.58). This was accomplished with only 40 epochs in the fine-tuning process for each of the tasks. Regarding the number of parameters retained after pruning, PBCE is far from SENSIMIX (13.75M) and



(a) BERT Base



(b) BERT Large

Figure 6: Median per layer of the distribution of r_f for BERT Base (upper image) and BERT Large (lower image). The components of the BertLayer for each layer are shown: Self-Attention Layer (Q, blue; K, red; V, green) and Intermediate (orange). A higher value of r_f (y-coordinate) indicates a greater contribution of information generated by that component to the network. As can be observed, the flow of information in both networks is different, showing distinct behaviors in each component and layer for BERT Base and Large.

MicroBERT (14.5M), although the accuracy results indicate that PBCE achieves the best balance between accuracy and size.

4.3 Results and analysis on the BERT Large model

In contrast to BERT Base, there are far fewer experimental results reported in the state of the art for BERT Large. Once again, PBCE outperforms RPP [18] in all tasks except for MRPC and RTE, while improving upon the original model in MNLI (0.4/0.33), QNLI (0.45), and CoLA (1.28) (Table 5), again, with only 40 epochs in the fine-tuning process for each of the tasks. In this case, the reduction in the number of parameters is significant, resulting in a compressed model with only 43% of the parameters (from 340M to 146.2M). This can be explained by the fact that BERT Large is much larger than BERT Base, hence it exhibits greater redundancy.

Table 4: Results of compressing BERT Base on GLUE Benchmark tasks. For each metric and task, the higher the value the better (Table 1). For the remaining parameters value (RP), expressed in millions of parameters, the smaller the value the better. Best results are highlighted in bold (excluding original BERT)

Method	**MNLI-(m/mm)**	**QQP**	**QNLI**	**SST-2**	**CoLA**	**STS-B**	**MRPC**	**RTE**	**RP (M)**
Original [14]	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	-
AUBER [25]	-	-	-	-	60.59 ± 0.73	-	85.62 ± 0.51	65.31 ± 1.30	-
AE-BERT [21]	-	-	88.7	-	-	86.1	89.5	69.7	-
ETbLSL [26]	82.9	90.7	88.2	89.3	52.6	84.6	88.3	63.9	-
LotteryTicketBert [11]	-	-	88.9	-	53.8	88.2	84.9	66	-
Michel et al. [31]	-	-	-	-	58.86 ± 0.64	-	84.22 ± 0.33	63.9	-
Voita et al [44]	-	-	-	-	55.34 ± 0.81	-	83.92 ± 0.71	64.12 ± 1.65	-
QBERT [41]	77.02/76.56	-	-	84.63	-	-	-	-	-
YOCO-BERT [50]	82.6	90.5	87.2	91.6	59.8	-	89.3	72.9	67
SENSIMIX [35]	-	89.6	86.5	90.3	-	-	87.2	-	**13.75**
LAD [28]	81.01/81.47	87.56	89.24	91.74	-	-	88.71	67.15	52.5
DDK [52]	83.6	88.2	91.6	**92.7**	61.9	89.1	90.7	**73.7**	67
UEM [22]	-	86.2	86.4	87.5	46.8	-	86.7	-	66.8
MicroBERT [53]	80.3	86.6	-	89.6	-	-	88.7	62.8	14.5
PBCE	**83.7/81.6**	**91.23**	**91.73**	91.87	**62.96**	**89.52**	**91.09**	71.98	52

Table 5: Results of compressing BERT Large on GLUE Benchmark tasks. For each metric and task, the higher the value the better (Table 1). For the remaining parameters value (RP), expressed in millions of parameters, the smaller the value the better. Best results are highlighted in bold (excluding original BERT)

Method	**MNLI-(m/mm)**	**QQP**	**QNLI**	**SST-2**	**CoLA**	**STS-B**	**MRPC**	**RTE**	**RP (M)**
Original [14]	86.7/85.9	72.1	92.7	94.9	60.5	86.5	89.3	70.1	-
RPP [18]	86.1/85.7	-	-	-	61.3	-	88.1	70.1	201
PBCE	**87.1/86.2**	**71.9**	**93.2**	**94.7**	**62.1**	**85.2**	**88.6**	**71.7**	**146.2**

5 Discussion

The current section elaborates on the key results detailed in the previous section. As evidenced in Table 4, PBCE demonstrates superior performance on 6 of the 8 GLUE Benchmark tasks when applied to BERT Base. Notably, for the remaining tasks, while our approach may not achieve the highest performance, offers a substantial reduction in model size, with a parameter count that is more than 20% lower (15M fewer parameters) compared to the DDK method. Moreover, when scaled up to BERT Large, Table 5 shows that

PBCE consistently outperforms the state-of-the-art, achieving a 27% reduction in model size (55M fewer parameters).

In summary, given the results and methodology, it can be concluded that research questions RQ1 and RQ2 have been answered. It has been demonstrated that persistent homology is a useful tool for selecting the most important neurons in the predictive process. Furthermore, a methodology such as PBCE is proposed to simplify the BERT models.

The explainability of a machine learning is associated with its internal logic and the transformations that occur within it. The more explainable a model is, the greater the level of understanding a human can achieve in terms of the internal processes that take place when the model makes decisions [29]. As mentioned in [17], understanding the role of individual units in the inference and learning process is one possible way to endow explainability to a deep neural model. Given the fact that PBCE identifies the most significant units for each layer by analyzing the topological features of its outputs, it can be considered that PBCE provides explainability to highly complex neural networks, such as Transformer encoders like BERT, answering research question RQ3.

6 Conclusions

Large Language Models (LLMs) are revolutionizing a number of applications of artificial intelligence, especially in the field of natural language processing. However, due to their complex Transformer-based architecture, LLMs are black-box models with a large number of parameters. In this work, PBCE is presented, a zero-dimensional persistent homology Green AI methodology designed to compress the BERT models and give insights of the role of its units in the inference and learning process. In particular, PBCE analyzes the topological characteristics of neuron outputs, thereby identifying which neurons contribute more information to the inference process and which ones are dispensable. Even though it can be simply described through a geometric procedure, it has well founded mathematical background based on homology theory. Therefore, persistent homology is an effective tool for selecting key neurons, answering RQ1.

As a result of the proposed methodology, simplified versions of the BERT model are built that outperform state-of-the-art techniques, even surpassing the original model's performance for most tasks included in the GLUE Benchmark. This allows to understand the topological behavior of LLMs like BERT, making them more explainable while maintaining performance and increasing efficiency. Therefore, RQ2 and RQ3 are answered by PBCE, which enhances model explainabilit.

As a line of research, it would be worthwhile to investigate the generalization of PBCE to other encoder architectures, particularly those based on the BERT family (e.g., RoBERTa, DeBERTa, DistillBERT). Furthermore, it would be interesting to explore the potential of applying PBCE to downstream tasks by fine-tuning these models on specific predictive objectives, such as sentiment classification, text summarization, and machine translation.

Acknowledgment

This research was partially supported by the project wth reference PID2020-118224RB-100 (funded by MICIU/AEI/10.13039/501100011033) and PID2023-151336OB-I00 granted by the Spanish Ministerio de Ciencia, Innovación y Universidades.

A Homology theory: notation and mathematical background

Firstly, based on the book [15] and the survey [20], in which a thorough explanation of homology theory can be found, some concepts are introduced, such as affine combination, affine independence, $k-$simplex, face of a $k-$simplex, simplicial complex, chain complexes and the boundary of a $p-$simplex, which are essential for building the theory of homology. Let $v_0, v_1, \ldots, v_k$ be vectors in $\mathbb{R}^d$. A point

$$x = \sum_{i=0}^{k} \lambda_i v_k$$

is an affine combination of the v_i if $\sum_{i=0}^{k} \lambda_i = 1$. The set of affine combinations constitutes the affine hull.

Affinely Independent. Let $x = \sum \lambda_i v_i, y = \sum \gamma_i v_i$, $x, y \in \mathbb{R}^d$ affine combinations. x and y are affinely independent when $x = y$ if and only if $\lambda_i = \gamma_i \forall i$. In other words, in a plane of dimension k ($k-$plane), $k+1$ points are affinely independent if the k vectors $v_i - v_0$, for $1 \leq i \leq k$, are linearly independent.

An affine combination $x = \sum \lambda_i v_i$ is a convex combination if $\lambda_i \geq 0$, for $1 \leq i \leq k$. The set of convex combinations is called the convex hull. Now, the $k-$simplex concept is introduced.

$k-$simplex. A $k-$simplex, σ, is the convex hull of $k+1$ affinely independent points. Its dimension is $\dim \sigma = k$.

Note that a $0-$simplex is a vertex, a $1-$simplex is an edge, a $2-$simplex is a triangle and a $3-$simplex is a tetrahedron. A face of a $k-$simplex σ is the convex hull of a non-empty subset of $\{v_0, v_1, \ldots, v_k\}$. If τ is a face of σ, then $\tau \leq \sigma$.

At this point, it is interested to focus on sets of simplices that are closed under taking faces and that have no improper intersections.

Simplicial complex. A simplicial complex is a finite collection of simplices K such that

1. $\tau \in K \; \forall t \leq \sigma, \sigma \in K$.
2. If $\sigma, \sigma_0 \in K \Rightarrow \sigma \cap \sigma_0$ is empty of a face of both.

Simplicial complexes ultimately emerge as the intersection of collections of sets. While this theory is developed considering entirely general simplicial complexes, in practice, the mentioned sets will be finally considered as geometric disks. Thus, the special case of the Vietoris-Rips (VR) complex arises, where the convex sets are disks with radius r. The VR complex will play a fundamental role in the methodology proposed, as it will be crucial for interpreting persistent homology and its application in PBCE towards the explainability and simplification of the BERT model.

Vietoris-Rips VR complex. Let (P, d) a finite metric space. The VR complex of P and r consisting of all subsets of diameter at most $2r$. That is, $\sigma \in \mathbb{R}(r) \Leftrightarrow d(p, q) \leq 2r, \forall p, q \in \sigma$.

After introducing the most basic ideas, let's construct more elaborated concepts that will be part of the definition of the homology group or persistent homology, which are the key tools of this work.

Chain complexes. Let K be a simplicial complex and p a dimension. A $p-$chain is a formal sum of $p-$simplices, $c = \sum a_i \sigma_i$, where σ_i are $p-$simplices and the a_i are modulo 2 coefficients.

Two $p-$chains can be added componentwise. In fact, $p-$chains with the addition operation form the abelian group of $p-$chains $(C_p, +)$.

A $p-$simplex is defined as the sum of its $(p-1)-$dimensional faces. If $\sigma = [v_0, v_1, \dots, v_p]$ for the simplex defined by the listed vertices, its boundary is

$$\partial_p \sigma = \sum_{j=0}^{p} [v_0, v_1, \dots, \tilde{v_j}, \dots, v_p],$$

where $\tilde{v_j}$ is omitted. For a $p-$chain, $c = \sum a_i \sigma_i$, the boundary is $\partial_p c = \sum a_i \partial_p \sigma_i$. Hence, the boundary maps a $p-$chain to a $(p-1)-$chain

$$\partial_p : C_p(K) \mapsto C_{p-1}(K).$$

Note that $\partial_p(c + c') = \partial_p c + \partial_p c'$, in other words, the boundary is an homomorphism. The chain complex is the sequence of chain groups connected by boundary homomorphisms,

$$\dots \xrightarrow{\partial_{p+2}} C_{p+1}(K) \xrightarrow{\partial_{p+1}} C_p(K) \xrightarrow{\partial_p} C_{p-1}(K) \dots$$

The elements of $Z_p(K) = \ker(\partial_p)$ are called $p-$cycles and those of $B_p(K) = \text{im}(\partial_{p+1})$ are called $p-$boundaries.

Homology group. The $p-$th homology group is defined as

$$H_p = Z_p / B_p.$$

The $p-$th Betti number is the rank of this group, $\beta_p = rank\ H_p$.

Finally, persistent homology is presented, which measures the scale of a topological feature combining geometry and algebra. Consider a simplicial complex, K, and a $f : K \mapsto \mathbb{R}$ monotonic, which implies that the sublevel set, $K(a) = f^{-1}(-\infty, a]$ is a subcomplex of K $\forall a \in \mathbb{R}$. Letting m be the number of simplices in K, $n + 1 \leq m + 1$ different subcomplexes appear

$$\emptyset = K_0 \subseteq K_1 \subseteq \cdots \subseteq K_n = K.$$

In other words, if $-\infty = a_0 < a_1 < a_2 < \cdots < a_n$ are the function values of the simplices in K, then $K_i = K(a_i)$. This sequence of complexes is known as the filtration of f. For every $i \leq j$, there is an inclusion map from K_i to K_j and therefore an induced homomorphism

$$f_p^{i,j} : H_p(K_i) \mapsto H_p(K_j).$$

The filtration corresponds to a sequence of homology groups connected by homomorphisms

$$0 = H_p(K_0) \mapsto H_p(K_1) \mapsto \cdots \mapsto H_p(K_n) = H_p(K).$$

for each dimension p. In the transition from $Ki - 1$ to K_i, new homology classes are gained and other are loosen when they merge with each other. Classes that are born are collected at a given threshold and die after another threshold in groups.

Persistent homology. The $p-$th persistent homology groups are the images of the homomorphisms induced by inclusion, $H_p^{i,j} = \text{im } f_p^{i,j}$, for $0 \leq i \leq j \leq n$. The corresponding $p-$the persistent Betti numbers are the rank of these groups.

The collection of persistent Betti numbers can be visualized by drawing points in the extended real plane $\overline{\mathbb{R}^2}$. Letting $\mu_p^{i,j}$ be the number of $p-$dimensional classes born at K_i and dying entering K_j,

$$\mu_p^{i,j} = (\beta_p^{i,j-1} - \beta_p^{i,j}) - (\beta_p^{i-1,j-1} - \beta_p^{i-1,j}),$$

$\forall i < j, \forall p$. Drawing each point (a_i, a_j) with multiplicity $\mu_p^{i,j}$, the $p-$th persistence diagram of the filtration is defined.

References

[1] Download Microsoft Research Paraphrase Corpus from Official Microsoft Download Center — microsoft.com. https://www.microsoft.com/en-us/download/details.aspx?id=52398. [Accessed 04-09-2023].

[2] google-bert/bert-base-cased · Hugging Face — huggingface.co. https://huggingface.co/bert-base-cased. [Accessed 03-09-2023].

[3] google-bert/bert-large-cased · Hugging Face — huggingface.co. https://huggingface.co/bert-large-cased. [Accessed 03-09-2023].

[4] legacy-datasets/wikipedia · Datasets at Hugging Face — huggingface.co. https://huggingface.co/datasets/wikipedia. [Accessed 08-09-2024].

[5] The Stanford Question Answering Dataset — rajpurkar.github.io. https://rajpurkar.github.io/SQuAD-explorer/. [Accessed 04-09-2024].

[6] Luisa Bentivogli, Ido Dagan, and Bernardo Magnini. *The Recognizing Textual Entailment Challenges: Datasets and Methodologies*, pages 1119–1147. Springer Netherlands, Dordrecht, 2017.

[7] Verónica Bolón-Canedo, Laura Morán-Fernández, Brais Cancela, and Amparo Alonso-Betanzos. A review of green artificial intelligence: Towards a more sustainable future. *Neurocomputing*, 599:128096, 2024.

[8] Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. pages 1–14, August 2017.

[9] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, Wei Ye, Yue Zhang, Yi Chang, Philip S. Yu, Qiang Yang, and Xing Xie. A survey on evaluation of large language models. *ACM Trans. Intell. Syst. Technol.*, 15(3), March 2024.

[10] Muyi Chen, Daling Wang, Shi Feng, and Yifei Zhang. Topological regularization for representation learning via persistent homology. *Mathematics*, 11(4), 2023.

[11] Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. The lottery ticket hypothesis for pre-trained bert networks. 33:15834–15846, 2020.

[12] Seungho Choe and Sheela Ramanna. Cubical homology-based machine learning: An application in image classification. *Axioms*, 11(3), 2022.

[13] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. What does bert look at? an analysis of bert's attention. *Association for Computational Linguistics*, pages 276–286, 2019.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. pages 4171–4186, June 2019.

[15] Herbert Edelsbrunner and John Harer. *Computational Topology: An Introduction*. 01 2010.

[16] Prakhar Ganesh, Yao Chen, Xin Lou, Mohammad Ali Khan, Yin Yang, Hassan Sajjad, Preslav Nakov, Deming Chen, and Marianne Winslett. Compressing large-scale transformer-based models: A case study on BERT. *Transactions of the Association for Computational Linguistics*, 9:1061–1080, 2021.

[17] Leilani H. Gilpin, David Bau, Ben Z. Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining explanations: An overview of interpretability of machine learning. pages 80–89, 2018.

[18] Fu-Ming Guo, Sijia Liu, Finlay S. Mungall, Xue Lin, and Yanzhi Wang. Reweighted proximal pruning for large-scale language representation. 2019.

[19] Manish Gupta and Puneet Agrawal. Compression of deep learning models for text: A survey. *ACM Trans. Knowl. Discov. Data*, 16(4), jan 2022.

[20] Felix Hensel, Michael Moor, and Bastian Rieck. A survey of topological machine learning methods. *Frontiers in Artificial Intelligence*, 4, 2021.

[21] Shaoyi Huang, Ning Liu, Yueying Liang, Hongwu Peng, Hongjia Li, Dongkuan Xu, Mimi Xie, and Caiwen Ding. An automatic and efficient bert pruning for edge ai systems, 2022.

[22] Tianyu Huang, Weisheng Dong, Fangfang Wu, Xin Li, and Guangming Shi. Uncertainty-driven knowledge distillation for language model compression. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 31:2850–2858, 2023.

[23] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. TinyBERT: Distilling BERT for natural language understanding. pages 4163–4174, November 2020.

[24] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. 2020.

[25] Hyun Dong Lee, Seongmin Lee, and U. Kang. Auber: Automated bert regularization. *PLOS ONE*, 16(6):1–16, 06 2021.

[26] Bingbing Li, Zhenglun Kong, Tianyun Zhang, Ji Li, Zhengang Li, Hang Liu, and Caiwen Ding. Efficient transformer-based large scale language representations using hardware-friendly block structured pruning. pages 3187–3199, November 2020.

[27] Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Cosgrove, Christopher D. Manning, Christopher Ré, Diana Acosta-Navas, Drew A. Hudson, Eric Zelikman, Esin Durmus, Faisal Ladhak, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue Wang, Keshav Santhanam, Laurel Orr, Lucia Zheng, Mert Yuksekgonul, Mirac Suzgun, Nathan Kim, Neel Guha, Niladri Chatterji, Omar Khattab, Peter Henderson, Qian Huang, Ryan Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas Icard, Tianyi Zhang, Vishrav Chaudhary, William Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, and Yuta Koreeda. Holistic evaluation of language models, 2023.

[28] Ying-Jia Lin, Kuan-Yu Chen, and Hung-Yu Kao. Lad: Layer-wise adaptive distillation for bert model compression. *Sensors*, 23(3), 2023.

[29] Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis. Explainable ai: A review of machine learning interpretability methods. *Entropy*, 23(1), 2021.

[30] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. 2019.

[31] Paul Michel, Omer Levy, and Graham Neubig. *Are sixteen heads really better than one?* Curran Associates Inc., Red Hook, NY, USA, 2019.

[32] Yuriy Mileyko, Sayan Mukherjee, and John Harer. Probability measures on the space of persistence diagrams. *Inverse Problems*, 27(12):124007, nov 2011.

[33] Mohammad Nauman, Hafeez Ur Rehman, Gianfranco Politano, and Alfredo Benso. Beyond homology transfer: Deep learning for automated annotation of proteins. *Journal of Grid Computing*, 17(2):225–237, Jun 2019.

[34] Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. A comprehensive overview of large language models, 2024.

[35] Tairen Piao, Ikhyun Cho, and U. Kang. Sensimix: Sensitivity-aware 8-bit index 1-bit value mixed precision quantization for bert compression. *PLOS ONE*, 17(4):1–22, 04 2022.

[36] Chi Seng Pun, Si Xian Lee, and Kelin Xia. Persistent-homology-based machine learning: a survey and a comparative study. *Artificial Intelligence Review*, 55(7):5169–5213, Oct 2022.

[37] Archit Rathore, Yichu Zhou, Vivek Srikumar, and Bei Wang. Topobert: Exploring the topology of fine-tuned word representations. *Information Visualization*, 22(3):186–208, 2023.

[38] Mukti Routray, Swati Vipsita, Amrita Sundaray, and Srinidhi Kulkarni. Deeprhd: An efficient hybrid feature extraction technique for protein remote homology detection using deep learning strategies. *Computational Biology and Chemistry*, 100:107749, 2022.

[39] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. 2020.

[40] Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. Green ai, 2019.

[41] Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. Q-bert: Hessian based ultra low precision quantization of bert. 2019.

[42] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. pages 1631–1642, October 2013.

[43] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. page 6000–6010, 2017.

[44] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. pages 5797–5808, July 2019.

[45] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. pages 353–355, November 2018.

[46] Zhiguo Wang, Wael Hamza, and Radu Florian. Bilateral multi-perspective matching for natural language sentences. page 4144–4150, 2017.

[47] Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641, 2019.

[48] Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. pages 1112–1122, June 2018.

[49] Kedi Wu, Zhixiong Zhao, Renxiao Wang, and Guo-Wei Wei. Topp–s: Persistent homology-based multi-task deep neural networks for simultaneous predictions of partition coefficient and aqueous solubility. *Journal of Computational Chemistry*, 39(20):1444–1454, 2018.

[50] Shaokun Zhang, Xiawu Zheng, Guilin Li, Chenyi Yang, Yuchao Li, Yan Wang, Fei Chao, Mengdi Wang, Shen Li, and Rongrong Ji. You only compress once: Towards effective and elastic bert compression via exploit–explore stochastic nature gradient. *Neurocomputing*, 599:128140, 2024.

[51] Xin Zhang, Jing Fan, and Mengzhe Hei. Compressing bert for binary text classification via adaptive truncation before fine-tuning. *Applied Sciences*, 12(23), 2022.

[52] Zhou Zhang, Yang Lu, Tengfei Wang, Xing Wei, and Zhen Wei. Ddk: Dynamic structure pruning based on differentiable search and recursive knowledge distillation for bert. *Neural Networks*, 173:106164, 2024.

[53] Dashun Zheng, Jiaxuan Li, Yunchu Yang, Yapeng Wang, and Patrick Cheong-Iao Pang. Microbert: Distilling moe-based knowledge from bert into a lighter model. *Applied Sciences*, 14(14), 2024.

4 MoEITS: A Green AI approach for simplifying MoE-LLMs

- Balderas, L., Lastra, M., & Benítez, J. M. (2025). MoEITS: A Green AI approach for simplifying MoE-LLMs
  - Status: **Submitted**.
  - Impact Factor (JCR 2023): **7.5**
  - Subject Category: **Computer Science, Artificial Intelligence**
  - Rank: **0/204**
  - Quartile: **Q1**

MoEITS: A Green AI approach for simplifying MoE-LLMs

Luis Balderas
Department of Computer Science
and Artificial Intelligence
DiCITS, iMUDS, DaSCI
University of Granada, Granada, Spain 18071
luisbalru@ugr.es

Miguel Lastra
Department of Software Engineering
DiCITS, iMUDS, DaSCI
University of Granada, Granada, Spain 18071
mlastral@ugr.es

José M. Benítez
Department of Computer Science
and Artificial Intelligence
DiCITS, iMUDS, DaSCI
University of Granada, Granada, Spain 18071
J.M.Benitez@decsai.ugr.es

Abstract

Large Language Models have taken the world by storm. They have attracted the attention of researchers, practitioners and even the general public. In the trek for more powerful architectures Mixture-of-Experts, inspired by ensemble models, have emerged as one of the most effective ways to follow. This implies, however, high computational burden both for training and inference. To reduce the impact on computing and memory footprint as well as the energy consumption simplification methods has arisen as very effective procedures.

In this paper, an original algorithm, MoEITS, for MoE-LLMs simplification is presented. The beauty of the algorithm lies in its simplicity and on being based on well-established Information Theory. MoEITS is analyzed in depth from theoretical and practical points of view. Its computational complexity is studied. Its performance on accuracy of the simplified LLMs and reduction rate achieved is assessed through a thoroughly designed experimentation. This empirical evaluation includes a comparison with state-of-the-art MoE-LLM pruning methods applied on Mixtral 8×7B. The results attest the superiority of the compressed models produced by MoEITS that are a 57% of their original size.

The code implementing the method will be available at https://github.com/luisbalru/MoEITS.

***K*eywords** MoE-LLMs, Mixtral 8×7B, pruning, simplification, normalized mutual information

1 Introduction

Large language models have disrupted the information society, revolutionizing all sectors, from a purely research perspective, with use cases in natural language processing [14, 29] or computer vision [21, 46]; to applied research, including medicine [17], smart cities [53, 51], or finance [3], among others. In particular, Mixture-of-Experts LLMs emerge, inspired by traditional ensemble models, to significantly augment the capabilities of LLMs with minimal computational overhead [4].

However, the quantity of hardware resources required to train and deploy models such as MoE-LLMs is immense. Furthermore, their use constitutes a significant impact on the environment [42, 43, 12]. To mitigate these harmful effects, the Green AI paradigm [35] has emerged, which aims to build more sustainable artificial intelligence systems through the use of metrics that consider not only accuracy but also the number of parameters, electricity consumption, or the number of model operations. Among these, simplification algorithms stand out as a fundamental tool for reducing the size of pre-trained models. In this article Mixture of Expert Information Theory Simplifier, MoEITS, is proposed, a novel simplification method for MoE-LLMs based on information theory. Specifically, normalized mutual information (NMI) is used as a metric to quantify the level of redundancy present in the experts of each network block. MoEITS is an algorithm that receives a parameter τ, which indicates the intensity with which the simplification of the models is carried out.

To measure the effectiveness of MoEITS, we have designed and carried out an extensive experimentation, including the simplification of MoE models such as Mixtral 8×7B [23] and benchmarks to evaluate the reasoning capacity of LLMs, such as BoolQ [6], HelloSwag [50], WinoGrande [34], ARC-e and ARC-c [7]. The main goal of this article is to address the following research questions (RQ):

(RQ1) To what extent are information theory metrics effective in measuring expert redundancy within a MoE?

(RQ2) Can an algorithmic methodology be defined that utilizes information theory to construct simplified MoE-LLMs?

(RQ3) At what point in the simplification process of a MoE, through the elimination of its experts, does the model cease to exhibit reasoning capabilities?

The main contributions of the article can be summarized as follows:

- A methodology based on normalized mutual information is proposed to detect redundancy among experts within a block and generate a simplified MoE.
- Both a theoretical analysis of computational complexity and an empirical analysis based on the simplification of Mixtral 8×7B evaluated on the most established LLM reasoning benchmarks are performed.
- An ablation study is conducted to measure the evolution of the generative capacity of Mixtral 8×7B applied to the BoolQ benchmark when different levels of simplification are applied.

The remainder of the article is organized as follows: Section 2 presents various state-of-the-art approaches for the simplification of MoE-LLMs, including those based on quantization and pruning. Section 3 provides a theoretical introduction to MoE-LLMs and the main current models. Section 4 presents MoEITS. Section 5 contains the empirical evaluation. Section 6 includes a discussion of the results, emphasizing the difference between simplification approaches and an ablation study. Finally, Section 7 presents the conclusions of the article.

2 Related Work

As with other approaches related to neural network simplification, especially for language models, quantization and pruning are the two most prominent paradigms. In the case of MoE models, this necessity becomes even more pressing, given the substantial increase in the number of parameters resulting from the introduction of expert submodels. The following sections present the state-of-the-art techniques in each of these two disciplines.

2.1 Quantization techniques

Quantization has emerged as a promising technique for compressing LLMs, particularly for deployment in resource-constrained environments, by reducing the precision of model weights and activations [20, 11, 27]. This section explores and compares several recent quantization techniques designed for MoE-LLMs, grouping them according to their underlying principles.

Post-Training Quantization (PTQ) methods offer a streamlined approach by quantizing model weights after the training phase, eliminating the need for further training. This simplicity makes them computationally inexpensive and highly attractive. Xiao *et al.* introduce in [45] a method called Smooth Quant, which tackles the challenge of quantizing activations, which are often more difficult due to outliers, by redistributing the quantization difficulty between weights and activations. It achieves this by scaling both, effectively smoothing the activations to make them more quantization-friendly. Another PTQ method, BILLM [19], focuses on aggressively compressing LLM weights down to approximately 1 bit while preserving accuracy. BILLM achieves this extreme compression through a combination of strategies. First, it structurally selects salient weights based on Hessian metrics and employs a residual approximation to maintain the dynamic range of these crucial weights. Second, it utilizes an optimal splitting strategy for the remaining non-salient weights to minimize binarization errors, leveraging the observation that these weights often follow a bell-shaped distribution. Both Smooth Quant and BILLM share the goal of preserving the most important information within the model during quantization, but they differ in their specific approaches. Smooth Quant emphasizes the importance of activation scaling, while BILLM prioritizes the identification and preservation of salient weights through Hessian metrics and residual approximation.

Quantization-Aware Training (QAT) methods integrate the quantization process directly into the training phase, allowing the model to adapt to the effects of quantization and potentially achieve superior accuracy compared to PTQ methods. PB-LLM [49] exemplifies this approach by exploring how to optimally identify salient weights and reintegrate them into the model through both PTQ and QAT. This hybrid approach allows PB-LLM to achieve substantial compression while maintaining the language reasoning capabilities of the original, larger model. PB-LLM leverages the concept of mixed-precision quantization, assigning varying bit-widths to different parts of the model based on their relative importance. However, the specific strategies for determining these bit-widths and managing different model components (e.g., experts vs. tokens) distinguish these methods.

Mixed-precision quantization methods strategically assign different bit-widths to various parts of the model according to their sensitivity to quantization or their overall importance. MC-MoE [18] embodies this strategy by allocating varying bit-widths to experts within the MoE architecture based on their significance. This targeted approach allows for high compression rates. Furthermore, MC-MoE incorporates online dynamic pruning, identifying and pruning less important tokens and dynamically selecting experts during inference. This combination of mixed-precision quantization and dynamic pruning further optimizes computational efficiency, setting MC-MoE apart from other methods. Finally, Activation-Aware Weight Quantization (AWQ) [28] introduces a novel approach to weight quantization by focusing on the distribution of activations rather than solely on weight magnitudes. This method identifies and protects the most salient weights by scaling them based on activation distributions. By focusing on activations, AWQ avoids the need for costly retraining or reconstruction steps, thus preserving the generalization capabilities of LLMs across different tasks and modalities. This activation-centric perspective distinguishes AWQ from methods relying on weight magnitudes or Hessian metrics.

2.2 Pruning Mixture-of-Experts LLMs

The compression of MoE LLMs via pruning techniques has emerged as a critical area of research [2, 31]. Several novel methodologies have been proposed, each with distinct approaches to identifying and removing redundant or less impactful experts and parameters. A common thread across these studies is the goal of balancing model compression with the preservation of task performance.

One cluster of approaches focuses on expert-level pruning, with variations in how experts are deemed redundant. Lu *et al.* [30] explore both task-agnostic and task-specific expert pruning, demonstrating significant efficiency gains on the Mixtral 8×7B model. Zhang *et al.* [52] propose a two-stage task-agnostic method that groups and merges similar experts based on feature representations, showcasing its effectiveness across various MoE architectures. These methods underscore the importance of identifying and eliminating redundant experts to reduce computational overhead.

A more refined strategy is presented by Lee *et al.* [26], who introduce STUN (Structured-Then-Unstructured Pruning), a method that first applies structured pruning at the expert level, followed by unstructured pruning. This counter-intuitive approach leverages behavioral similarity among experts to achieve superior compression rates and performance. Complementing these approaches, Xie *et al.* [47] introduce MoE-Pruner, which identifies and removes unimportant weights based on magnitude, input activations, and router weights, further enhanced by expert-wise knowledge distillation.

Furthermore, Yang *et al.* [48] present MoE-I^2, a two-stage framework that combines inter-expert pruning using genetic search with intra-expert decomposition via low-rank approximation, offering a task-agnostic structured compression. Besides, Chowdhury *et al.* propose in [5] a method which selectively removes experts that not relevant for specific tasks. Therefore, it reduces memory requirements and runtime costs. Finally, Gao *et al.* [16] introduce ToMoE, a dynamic structural pruning technique that converts dense LLMs into MoE models by selectively activating model parameters, achieving sparse and efficient computation.

Collectively, these studies advance the field by proposing diverse pruning techniques tailored to MoE architectures, addressing the challenges of model compression while maintaining or even enhancing performance. The choice of pruning method depends on the specific requirements of the application, including the balance between compression rate, computational efficiency, and task-specific performance.

3 Preliminaries: Mixture-of-Experts Large Language Models

Mixture-of-Experts (MoE) models, first introduced in [22], are models containing multiple feed-forward neural networks (FFN), acting as experts, that independently process a complete dataset. Following this processing, the best experts for each inference are dynamically selected by means of a router that assigns weights to the experts' contributions. Concretely, a MoE architecture comprises a set of independent FFNs called experts, $\{E_1, E_2, \dots, E_n\}$, and a router network or gate G. The output is calculated as follows:

$$F(x) = \sum_{i=1}^{n} g_i E_i(x), \tag{1}$$

with

$$g_i = \begin{cases} s_i, & s_i \in \text{TopK}(\{s_i, 1 \leq i \leq n\}, K) \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

$$G(x) = [s_1, s_2, \ldots, s_n] \tag{3}$$

where g_i represents the score for the i-th expert, $G(x)$ is the output of the gate, corresponding to the affinity between the input token and each expert, and $\text{TopK}(\cdot, K)$ yields the K highest values among the experts' scores for each input x ([33]).

Although the MOE structure was employed for the design of Recurrent Neural Networks [38], it has gained significant popularity in recent years due to its application in Transformer models, including both encoder-decoder and decoder-only architectures, for natural language applications. Some of the most relevant MoE architectures for LLMs are Mixtral 8×7B [23], Google Switch-Transformers [15], Qwen1.5 MoE [40], and DeepSeek MoE [9].

Google's Switch Transformers models address the computational challenges associated with scaling transformer-based language networks by introducing a simplified routing mechanism within the MoE framework. Rather than employing complex gating networks to determine expert selection, Switch Transformers utilize a single "switch" layer, which directs each token to a single expert. This simplification significantly reduces routing overhead, thereby enabling the training of extremely large MoE models with greater efficiency. The core innovation of Switch Transformers lies in their ability to maintain high model capacity while minimizing computational cost. By directing each token to a single, chosen expert, the computational burden is distributed across the available experts, allowing for the scaling of model parameters without a proportional increase in processing demands. This approach facilitates the training of models with orders of magnitude more parameters than traditional dense transformers, leading to improved performance on a variety of language tasks. Furthermore, the simplified routing mechanism contributes to increased training stability and reduced communication overhead, making Switch Transformers a practical and effective approach for scaling language models to unprecedented sizes. In essence, Google's Switch Transformers provide a pathway to efficient and scalable language modeling through the judicious application of a simplified MoE architecture.

Mixtral 8×7B represents a highly acclaimed and widely adopted MoE architecture, establishing itself as a de-facto benchmark within both industry and academia, particularly in studies concerning the simplification of MoE architectures. This decoder-only model comprises eight expert per layer, yet activates only two at any given inference step. Consequently, the number of active parameters per token is significantly lower compared to other Large Language Models (LLMs), such as Llama 2 70B [41] (13B versus 70B, respectively), while achieving superior performance. This efficient parameter utilization underscores Mixtral 8×7B's ability to deliver high-quality outputs with reduced computational demands, facilitating its application in resource-constrained environments and its adoption as a standard for evaluating and advancing MoE methodologies.

DeepSeek AI's DeepSeekMoE 16B model marks a notable progression in the field of large language models, particularly through its refined implementation of the Mixture-of-Experts (MoE) architecture. The model's core innovation resides in its capacity to achieve a sophisticated balance between high-level performance and computational efficiency, a feat primarily realized through the strategic deployment of sparse activation within its network. The model's architectural design centers on a sparse MoE framework, which, unlike dense models, selectively activates specific subsets of its parameters during each processing cycle. This selective activation allows for the maintenance of a substantial overall parameter count—16 billion—while simultaneously mitigating the computational overhead typically associated with such large-scale models. DeepSeek AI has further augmented the MoE paradigm through the introduction of fine-grained expert segmentation and shared expert isolation. These methodological advancements are engineered to foster a higher degree of specialization among the model's constituent experts, thereby enhancing the granularity and fidelity of its knowledge representation.

Finally, Qwen1.5-MoE, developed by Alibaba Cloud, distinguishes itself by its focus on balancing performance with computational efficiency, a critical consideration for deploying large models in practical applications. The architecture of Qwen1.5-MoE leverages the MoE paradigm to selectively activate a subset of its parameters during each inference, thereby mitigating the computational cost associated with dense transformer models. This approach allows for the model to possess a substantial parameter count, enhancing its capacity to capture complex linguistic patterns, while maintaining a manageable computational footprint. Furthermore, the model incorporates advancements in routing mechanisms and expert management, contributing to improved stability and performance across diverse language tasks.

4 Our proposal

In this paper, we introduce MoEITS, a method for simplifying Mixture-of-Experts Large Language Models. To achieve this simplification, we employ tools from information theory [36], specifically Normalized Mutual Information (NMI) [13, 25], to quantify the redundancy present within each expert block of the network. This analysis enables the generation of a refined model where experts exhibit complementary behavior during inference tasks, effectively reducing redundancy. Despite its inherent power, MoEITS is remarkably easy to use, requiring only one parameter, the threshold τ, which indicates the level or intensity of simplification applied to the original model. Furthermore, MoEITS obviates the need for subsequent retraining, resulting in minimal computational demands for its application. More details can be found in Algorithm 1 and Figure 1.

Algorithm 1 MoEITS method

```
1: procedure MoEITS(model, $\tau$)
2:     Get NMI metrics from experts.
3:     Build the simplified model using NMI metrics and $\tau$.        ▷ [Algorithm 2]
4:     Add model's weights to the simplified model.
5:     return simplified model
6: end procedure
```

4.1 Using Normalized Mutual Information to reduce redundancy among experts

Information theory, introduced by Claude Shannon in [36], is a branch of mathematics dedicated to the study of the quantification and communication of information. One of the most important metrics in information theory is entropy, which measures the amount of uncertainty or information associated with the potential states of a random variable [37]. If $p(x)$ is the probability of the event $x \in X$, the entropy of a discrete random variable X is calculated as

$$H(X) = -\sum_{x} p(x) \log p(x) \tag{4}$$

In other words, $H(X)$ indicates how much information it can be found on average from one vector X. Similarly, joint entropy, denoted as $H(X, Y)$, can be defined as the amount of uncertainty inherent in two random variables, X and Y, considered together.

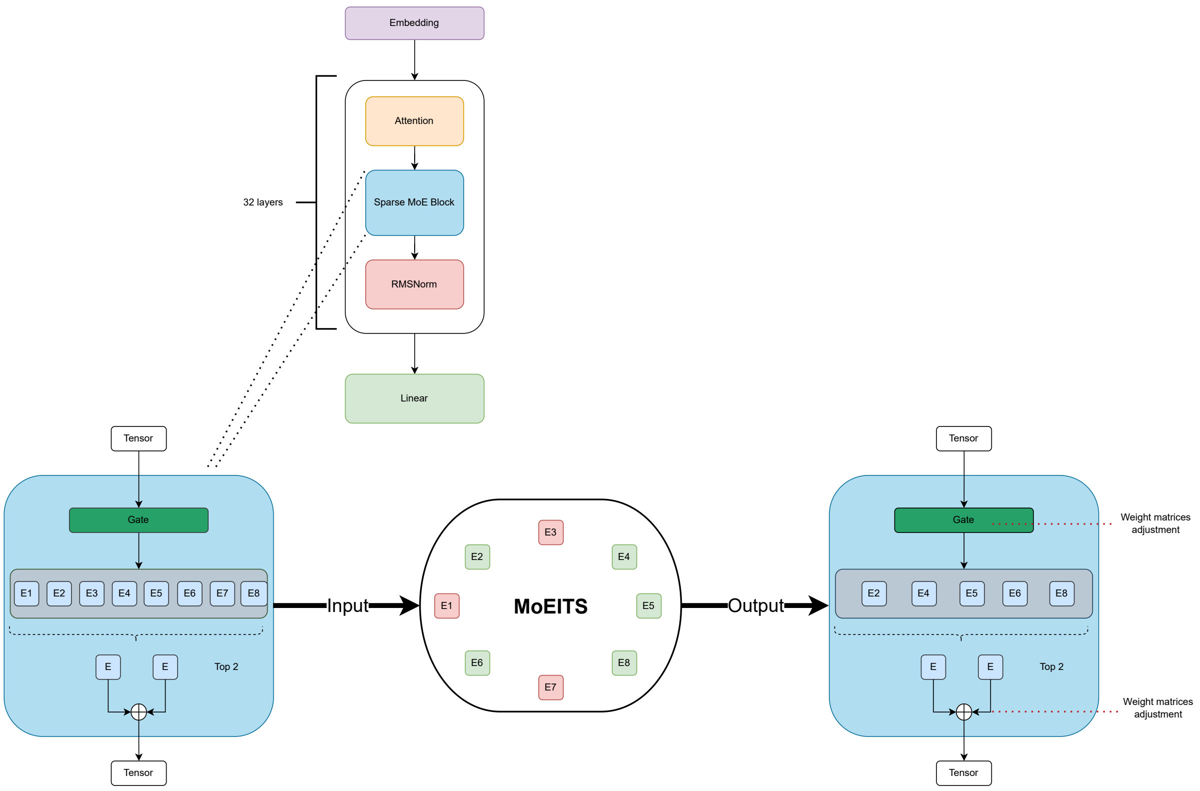


Figure 1: Diagram of MoEITS. Starting from a block of model experts, the redundancy analysis metric between experts based on normalized mutual information is applied. Next, the most relevant experts are simplified and a new version of the model is generated by simplifying, inheriting the knowledge of the original model by means of the weight matrices. Note that the layers immediately preceding and following the simplified expert block must be adjusted.

Another fundamental tool within information theory is mutual information, which is a symmetric metric that quantifies the statistical information shared between two distributions [8]. Mutual information is calculated as:

$$I(X,Y) = H(X) + H(Y) - H(X,Y) \tag{5}$$

Given that $I(X,Y)$ is a not upper-bounded metric, in order to enhance interpretability and comparability, it would be desirable to have a normalized version of $I(X,Y)$, that is, one that takes values between 0 and 1. Following the notation proposed in [39], we define Normalized Mutual Information (NMI) as:

$$\text{NMI}(X,Y) = \frac{I(X,Y)}{\sqrt{H(X)H(Y)}} \tag{6}$$

In light of the foregoing, NMI is a highly useful metric for quantifying the amount of shared information between two distributions or vectors. Furthermore, given that it takes values between 0 and 1, it is straightforward to compare results across different vectors. Specifically, given two vectors X and Y, a high value of $\text{NMI}(X,Y)$ implies that X and Y share a significant amount of information or, in other words, possess redundant information. Conversely, a low value of NMI(X, Y) implies that X and Y share little information in common.

Considering that this work aims to simplify MoE-LLMs by identifying and eliminating redundant experts (RQ1), and that each expert is defined by a specific number of layers, whose information and learning are expressed in a weight matrix, the MoEITS method employs NMI to measure the amount of redundant information in the matrix expressions of their weights. Suppose that within a block, we have two experts e_1 and e_2, defined by respective Linear layers L_1^{e1} and L_2^{e1} for e_1; L_1^{e2} and L_2^{e2} for e_2. Let $W_{L_i}^{e_j}$, with $i, j \in \{1, 2\}$ the weight matrix for each Linear layer and expert. The level of redundancy R between the two experts is defined as:

$$r_1 = \text{NMI}(W_{L_1}^{e_1}, W_{L_1}^{e_2}) \tag{7}$$

$$r_2 = \text{NMI}(W_{L_2}^{e_1}, W_{L_2}^{e_2}) \tag{8}$$

$$R = \frac{r_1 + r_2}{2} \tag{9}$$

In general, for every pair of experts e_k, e_o within a block, if each expert is formed by n layers

$$R = \frac{\sum_{i=1}^{n} r_i}{n}, \tag{10}$$

with

$$r_i = \text{NMI}(W_{L_i}^{e_k}, W_{L_i}^{e_o}) \tag{11}$$

The higher the value of R, the more redundant the experts are. The preceding decisions assume that, as is generally the case in most models, the experts are composed of layers that are equal. However, if this premise were not met, these definitions could be adjusted for size, and, in the case of the expression (9), introduce a weighted mean based on relevance. For simplicity, the arithmetic mean is used.

4.2 Experts pruning process

The previous section elucidates how the MoEITS method employs NMI to quantify expert redundancy. This step corresponds to step 4 of Algorithm 2. This section focuses on how experts are selected using NMI (step 5 of Algorithm 2 and Algorithm 3).

Given a block of experts within a layer of a MoE, the application of normalized mutual information to all pairs of experts generates a symmetric matrix, which will be referred to as the NMI matrix, $\mathcal{R} = [r_{ij}]$. This matrix contains, at the i-th row and j-th column, the redundancy value between experts e_i and e_j, r_{ij}. Using this matrix, the MoEITS method selects which experts are least redundant from a collective perspective. For this purpose, an iterative algorithm is defined, which, inspired by the idea of using the interquartile range (IQR) for anomaly detection [10, 44], establishes a redundancy limit called ρ. This limit is calculated as the mean of the NMI matrix plus its interquartile range multiplied by τ, the threshold of the MoEITS method, which modulates the level of simplification applied to the network. The definition of ρ is inspired by the classical use of the IQR to measure the distance of a point from the center of the distribution to which it belongs. Given that the distribution of weights is very uniform across the layers of the experts, the mean is introduced to increase sensitivity to small variations and thus detect subtle but important differences among the experts.

Once the redundancy limit is established, the algorithm iterates over the NMI matrix, $\mathcal{R}$, searching for values that exceed ρ. Let us suppose that such a value exists and is located at position (i, j). This implies that experts e_i and e_j exhibit a level of redundancy higher than the required limit, and therefore, one of them could be removed. To decide whether e_i or e_j is the expert to prune, the redundancy they express with the rest is measured. The one that expresses greater redundancy with the other experts will be eliminated. To do this, the mean of the values in the i-th and j-th rows is calculated (it is also possible by columns, since $\mathcal{R}$ is a symmetric matrix), that is, the level of redundancy of experts e_i and e_j with the remaining experts. Let us suppose, without loss of generality, that the most redundant expert of this iteration e_i. Then the i-th row and column are removed from the NMI matrix, so that this expert is no longer considered. This process ends when there is no value in the $\mathcal{R}$ that exceeds the redundancy limit. Consequently, considering how ρ is calculated, it can be concluded that high values of τ generate very reduced simplifications, and conversely, small values —or values close to 0— of τ generate notable reductions in size. More details can be found in Algorithm 3.

Algorithm 2 Building the simplified model

```
 1: procedure SIMPLIFIED MODEL(model, $\tau$)
 2:     experts ← []
 3:     for $l$ in model.layers do
 4:         NMI_expert_block ← $\mathcal{R}$ for $l$-th expert block
 5:                                                    ▷ [See subsection 4.1)]
 6:         relevant_experts ← relevant experts in the $l$-th block
 7:                                  ▷ [Algorithm3(NMI_expert_block, $\tau$)]
 8:         experts.add(relevant_experts)
 9:     end for
10:     simplified model ← Create a new model including only relevant experts
11:                                            ▷ [relevant experts by layer]
12:     return simplified model
13: end procedure
```

Algorithm 3 Expert-Block Simplification

```
 1: procedure EXPERT-BLOCK SIMPLIFICATION($\mathcal{R}$, $\tau$)
 2:                              ▷ [Get non-redundant experts within a layer]
 3:     $m$ ← mean($\mathcal{R}$)
 4:     $iqr$ ← iqr($\mathcal{R}$)                         ▷ [Interquartile range]
 5:     $\rho \leftarrow m + iqr \times \tau$                  ▷ [Redundance limit]
 6:     while ($\exists\, r_{ij} \in \mathcal{R} > \rho$) do
 7: ▷ [Determine whether there exists any expert $i$ whose redundancy value with another expert $j$
    exceeds $\rho$.]
 8:         $e_1, e_2$ ← the two most similar experts
 9:         Measure the redundancy of $e_1, e_2$ with respect to the remaining experts
10:         Remove the most redundant expert $e_1$ or $e_2$
11:     end while
12:     return relevant experts
13: end procedure
```

The final step of Algorithm 1 involves assigning the weights of the original model to the simplified one. This represents the distillation of knowledge between the models. To achieve this, in those layers where simplifications have not been carried out, that is, all those unrelated to the experts, there is a direct assignment of weights from the original model to the simplified one. In the case of the experts, it is necessary to adjust the weight matrices by eliminating the rows and columns associated with the experts that have been removed.

This applies to both the unit preceding the experts, called the Gate, and the subsequent unit that collects their outputs.

4.3 Computational complexity analysis

This section analyzes the computational complexity of MoEITS. For this purpose, asymptotic Big-O notation, introduced in [24], is employed, representing a worst-case analysis. First of all, the following notation is established. Almost all of the published MoE-LLM models have the same number of experts per layer. Let us denote L as the number of layers in a MoE-LLM model, and e as the number of experts per layer. If the number of experts were not the same, then let $e = \max\{e_{l_1}, e_{l_2}, \ldots, e_{l_L}\}$. Let k be the number of layers per expert. It is again, typically a fixed number. If not, k is defined as the maximum. Finally, let ν be the maximum number of neurons in the layers of the experts.

As expressed in Algorithm 1, the MoEITS method comprises three distinct phases. The initial phase centers on the computation of normalized mutual information metrics across all experts. This computation is contingent upon the number of layers, the number of experts (scaling quadratically), the number of Linear layers per expert, and the number of neurons (also scaling quadratically). Consequently, the information-theoretic metric extraction phase exhibits a computational efficiency of O$(Le^2\nu^2k)$.

The second phase involves the construction of a simplified version of the original model, the details of which are delineated in Algorithms 2 and 3. As can be observed, Algorithm 2 exhibits a linear dependency on the number of layers, while Algorithm 3, in the worst-case scenario, performs e^2 iterations. Consequently, the generation of a new model also exhibits a linear dependency on L and a quadratic dependency on e. Therefore, the algorithmic complexity of this second phase is O(Le^2).

Finally, the algorithm's concluding stage is dedicated to the assignment and adjustment of weights within the simplified model. This process facilitates the transfer of knowledge from the original, pre-trained model to the simplified model. The computational time efficiency of this phase is O(Le^2).

In summary, taking into account the partial analysis of each stage of the algorithm, the computational complexity of MoEITS is O$(Le^2\nu^2k)$.

5 Empirical analysis

To assess the effectiveness of MoEITS, a thorough extensive experimentation has been designed and rigorously carried out. It is based on well established benchmarks in the literature. Furthermore, results are compared with those for other state-of-the-art techniques, both in terms of accuracy on each benchmark and the achieved reduction, whether in terms of the number of parameters, sparsity, or quantization.

5.1 Models, evaluation and datasets

Following the trend in other state-of-the-art proposals and to facilitate comparison with them, the Mixtral $8 \times$ 7B (46B) model is established as a reference for evaluating simplification. Nevertheless, MoEITS is a model-agnostic method, and thus it can be applied to any other MoE.

Similarly, to facilitate task-agnostic evaluation of the method, following the evaluation methodology proposed in [32], zero-shot task classification is performed across datasets such as BoolQ [6], HelloSwag [50], WinoGrande [34], ARC-e and ARC-c [7]. A zero-shot setting implies that the model is evaluated without being explicitly trained on the specific benchmark dataset. This tests the model's general knowledge and reasoning abilities. Table 1 contains more details about the tasks, including description, metrics and key focus. Accuracy is used as the prediction metric in all benchmarks. Additionally, the percentage of simplified parameters and sparsity, which refers to the number of weights set to zero after the simplification process, are included as measures to assess the level of simplification.

With respect to the state-of-the-art methodologies against which MoEITS is benchmarked, the comparative analysis incorporates the following techniques: MoE-I^2 [48] , MoE-Pruner [47], STUN [26], MC-MoE [18], and the method proposed in [30] by Lu *et al.* This selection of comparative methods serves to provide a comprehensive evaluation of MoEITS's performance relative to established and contemporary approaches in the field.

Regarding the experimentation, after a number of trial runs, it was found that $\tau = 0.75$ yielded the best value found across the totality of the benchmarks. Once τ was fixed, a model of 27B parameters is obtained (Figure 2 shows the distribution of experts along the layers). Besides, the number of shots utilized for each benchmark via the `DeepEval` library [1] was 3. Furthermore, to ensure the robustness of the results, the experiments were repeated three times, with the mean being taken as the definitive result.

5.2 Comparison with state-of-art methods

The empirical results obtained for the simplification of Mixtral 8×7B by MoEITS and the other methods from the state-of-the-art proposals are presented in Table 2. This table includes accuracy results for the various benchmarks. Simplification metrics are also included. In particular, the percentage of parameter reduction and sparsity. For both metrics, the higher the value, the greater the simplification.

As can be seen, MoEITS yields better results than the rest of the state-of-the-art methods for ARC-c, HelloSwag, ARC-c, and ARC-e. For WinoGrande and BoolQ, MoE-Pruner achieves higher accuracy. However, this is a method that uses sparsity metrics, and therefore, does not reduce the size of the model, making it non-applicable to systems with restricted hardware resources. In general terms, MoEITS achieves substantial results in terms of accuracy across the various benchmarks while simultaneously achieving a notable reduction in network size, specifically reducing to the 43% of the original number of parameters. In fact, as can be checked, the technique obtains the best average value of the reasoning metrics. Therefore, these results enable us to answer research question RQ2, as we can confirm that the presented methodology is effective for the simplification of MoE-LLMs.

6 Discussion

Considering the obtained results, detailed in the previous section, several topics for discussion are presented to further elucidate the key aspects of the proposed method. Specifically, a comparative analysis is introduced, contrasting simplification methods based on pruning and sparsity. Furthermore, an ablation study is performed, focusing on the parameter τ.

6.1 Sparsity and Pruning: Two Paths to Neural Network Efficiency

The comparison among neural network simplification methods must be conducted according to objective and well-recognized metrics. Firstly, metrics associated with the selected benchmarks are useful for assessing whether the models generated by the simplification methods are capable of solving learning tasks with adequate quality. Then, other fundamental metrics measure the level of simplification, that is, the extent to which the quantity of resources required to solve the referred to tasks has decreased. In this work, techniques based on sparsity as a simplification metric are compared. Other techniques quantify simplification by measuring the reduction in the number of parameters. Sparsity-inducing techniques, such as MoE-Pruner or STUN, operate by imposing constraints or penalties during the training process that encourage a subset of network weights to converge towards zero. This results in a sparse weight matrix, where a significant

Table 1: Overview of LLM benchmarks used for evaluating reasoning and comprehension capabilities for simplification methods, showing task descriptions and key focus. For all benchmarks, accuracy is the metric used.

Benchmark	**Task Description**	**Key Focus**
BoolQ	Question answering where the answer is either "yes" or "no." It tests a model's ability to understand a passage and answer a simple boolean question about it.	Reading comprehension, logical reasoning.
HelloSWag	Commonsense reasoning. Given a context, the model must choose the most likely ending from four options. It's designed to be challenging for AI by including adversarial examples.	Commonsense inference, context understanding.
WinoGrande	A large-scale dataset of pronoun resolution problems. The model must determine the referent of a pronoun in a sentence, requiring fine-grained understanding of context and world knowledge.	Pronoun resolution, coreference resolution, commonsense reasoning.
ARC-e	A set of multiple-choice science questions designed for elementary school level. It tests a model's ability to understand and reason about scientific concepts.	Scientific reasoning, knowledge application.
ARC-c	A more difficult subset of the ARC dataset, designed to require deeper reasoning and knowledge than ARC-e.	Advanced scientific reasoning, complex knowledge application.

Table 2: Results of pruning Mixtral 8×7B on BoolQ, ARC-e, ARC-c, WinoG (WinoGrande) and HelloS (HelloSwag) benchmarks. Av represents the average of the metrics. P $\downarrow$ represents the percentage reduction in the number of parameters. S (sparsity) refers to the portion of parameters that are zero. Best results are highlighted in bold.

Method	**BoolQ**	**ARC-e**	**ARC-c**	**WinoG**	**HellaS**	**Av**	**P** $\downarrow$	**S**
8x7B Or	85.4	84.0	57.2	75.9	84.4	77.4	-	-
MoE-I^2 [48]	82.6	78.2	52.2	71.5	61.1	69.1	**49%**	-
MoE-Pruner [47]	**86.0**	81.9	53.3	**75.5**	62.3	71.8	0%	50%
STUN [26]	-	80.0	51.5	-	59.9	63.8	0%	65%
MC-MoE [18]	80.6	73.1	48.4	71.3	74.9	69.6	-	-
Lu *et al.* [30]	84.8	79.9	53.9	73.8	60.1	70.5	40%	-
MoEITS	85.3	**82.1**	**57.1**	75.5	**74.9**	**74.9**	43%	-

proportion of connections are effectively nullified. However, it is crucial to recognize that these techniques perform a "logical simplification" rather than a physical one. The architectural structure of the network remains unchanged; the number of neurons and connections persists. Instead, the influence of certain connections is rendered negligible through the assignment of zero or near-zero weights. While this process yields a sparse weight matrix, its dimensions, $n \times m$, remain constant. Consequently, during inference, although multiplications involving zero weights do not contribute to the result, the computational overhead associated with these operations and the memory allocation for the entire weight matrix are not eliminated. Thus, sparsity-based methods provide a logical simplification, effectively deactivating certain connections without physically removing them. Unfortunately, they mean no contribution towards the Green AI effort, since the computation resources and energy required are not reduced.

In contrast, parameter pruning methods, exemplified by techniques such as MoEITS, execute a "physical simplification." These methods involve the explicit removal of redundant or inconsequential neurons and connections from the network architecture. The removal of parameters directly translates to a reduction in the computational and memory requirements. During inference, fewer operations are performed, and less memory is required to store the network's weights. Formally, the transition from the original model to the simplified one represents a structural modification of the network. This modification leads to a decrease in the number of floating-point operations required for inference and a reduction in the memory footprint.

6.2 Ablation study

The objective of this section is to analyze the evolution of the generative and reasoning capabilities of Mixtral 8×7B when different levels of simplification are applied. To this end, we select, without loss of generality, a benchmark, specifically BoolQ, and apply MoEITS with different threshold values or τ.

τ is used to control the level of simplification applied to the model. A decrease in this parameter corresponds to a more aggressive pruning. Figure 3 shows the evolution of the reasoning capacity of compressed Mixtral 8×7B models at different levels of compression. For values of τ ranging from 0.25 to 2.5, it displays in orange color (left axis) the size of the compressed models, measured in multiples of 10^{10}. The monotonic relationship between τ and the number of parameters is patent. On the other, the compressed models' accuracy is also plotted —in blue color (right axis). The accuracy generally follows the upward trend of τ, albeit not monotonically. For this benchmark, a peak value is observed around $\tau = 0.75$, where the model achieves a notably superior performance compared to its neighboring values. Subsequently, the performance undergoes a decline of approximately 10%, after which the accuracy increases in conjunction with τ, ultimately converging to the accuracy of the original pretrained model.

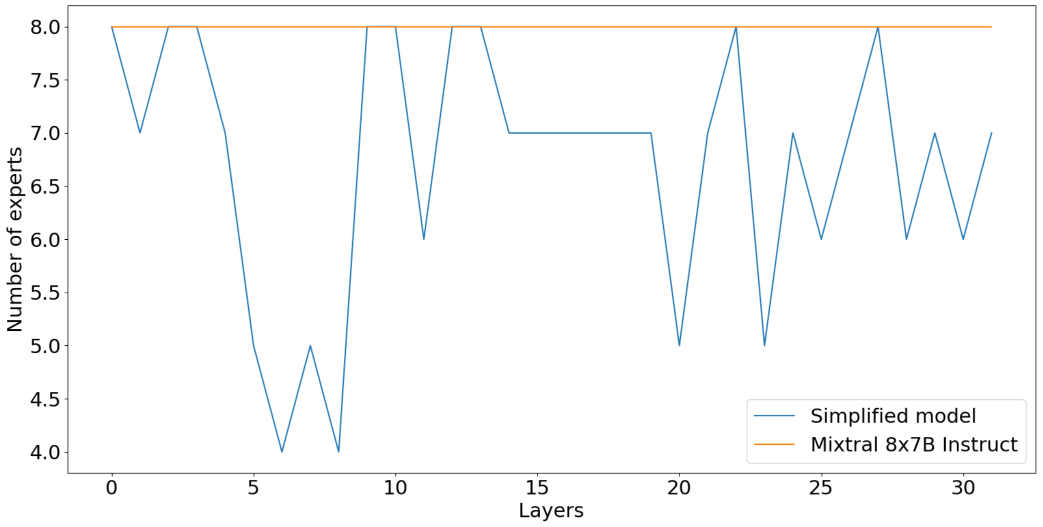


Figure 2: Distribution of experts per layer in the simplified model

Regarding research question RQ3, it is evident that for $\tau < 0.75$, the model's performance is drastically reduced in this case, even though the number of parameters is also significantly reduced (less than 50% of the original model's parameters). Consequently, parametric pruning algorithms, such as MoEITS, prove particularly valuable in identifying the optimal trade-off between performance and model size. Experiments like this show that a model can be readily generated to meet the hardware constraints of the target deployment device, while leveraging all available information regarding accuracy.

7 Conclusions

Large Language Models (LLMs), and particularly Mixture-of-Experts LLMs (MoE-LLMs), are models that have revolutionized both industry and academia by providing cutting-edge and effective solutions across virtually all disciplines of Machine Learning. However, not only has their training required vast amounts of data and computational capacity, but their deployment is also so infrastructure-demanding that it is within the reach of only a few companies and research centers. In this paper, MoEITS, a simplification method for MoE-LLMs based on information theory, is proposed. Specifically, it employs normalized mutual information to detect redundancy among the experts, enabling the removal of the most redundant ones and, consequently, significantly reducing the network with minimal loss in predictive quality.

To assess the effectiveness of MoEITS, a comprehensive analysis was conducted. On the one hand, from a theoretical perspective, it was established that its computational complexity is $\mathrm{O}(Le^2\nu^2k)$, where L represents the number of blocks, e indicates the number of experts per block, k represents the number of layers per expert and ν the maximum number of neurons in the layers of the experts. On the other hand, a thorough experimentation was designed and carried out. It includes the most established benchmarks in the literature, both at the model level using Mixtral 8×7B and at the task learning level. The results demonstrate that MoEITS is more effective and efficient than other state-of-the-art methods. Consequently, it can be asserted that NMI is a useful metric for measuring redundancy in experts (RQ1) and that MoEITS is a versatile tool for the simplification of MoE-LLMs (RQ2). Finally, an ablation study was conducted on the τ parameter to address (RQ3). The results demonstrate that beyond a certain simplification threshold, the model's reasoning

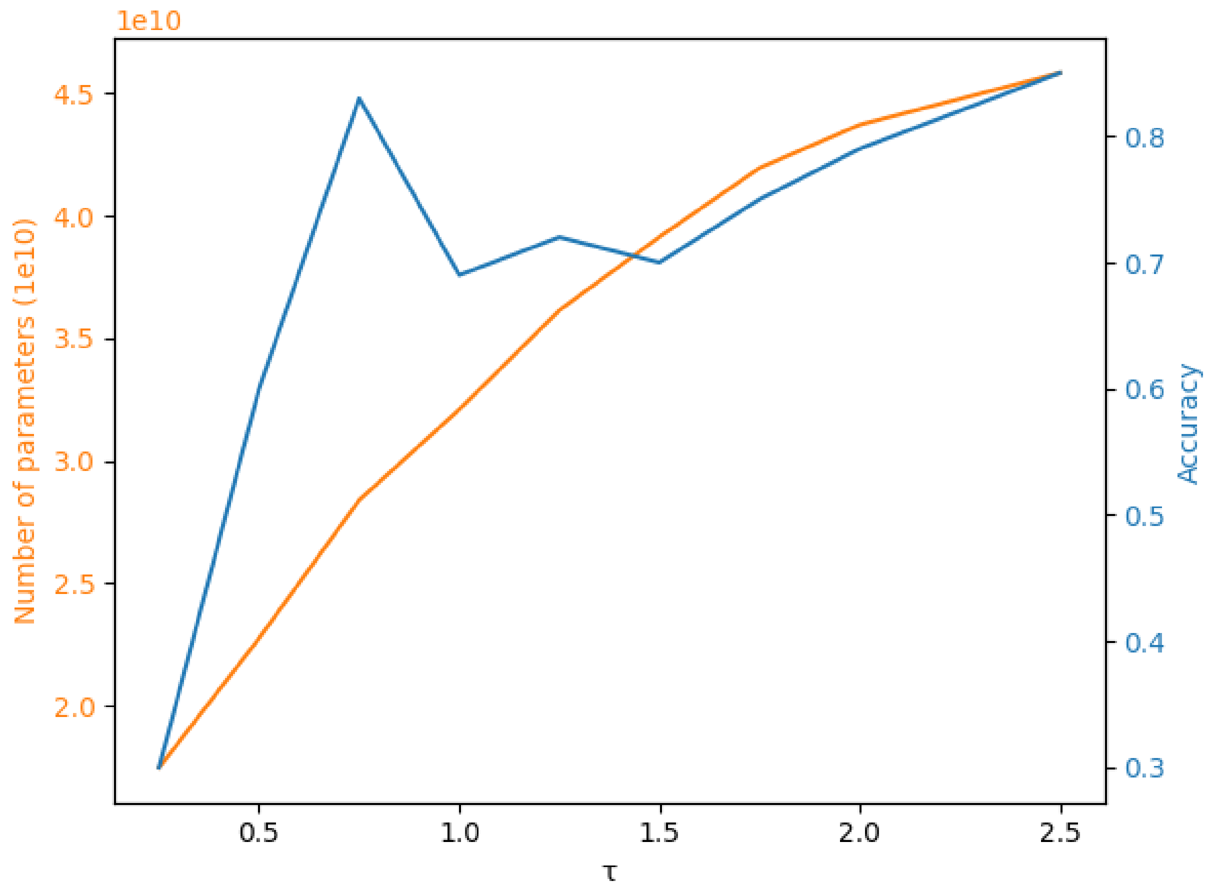


Figure 3: Evolution of the simplified Mixtral 8×7B model with MoEITS for different values of τ (number of parameters and accuracy). As can be observed, for $\tau = 0.75$, the performance is notably better, approaching the convergence value for the original model.

capacity undergoes a precipitous decline. Therefore, conducting such studies is crucial for determining the threshold value that aligns with the infrastructure requirements of the intended deployment environment.

Acknowledgment

This research has been partially supported by Proyecto PID2023-151336OB-I00 financiado por MICIU/AEI /10.13039/501100011033 y por FEDER, UE.

References

[1] Confident AI. Deepeval, 2024. https://docs.confident-ai.com/ [Accessed: 23/02/2025].

[2] Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. SliceGPT: Compress large language models by deleting rows and columns. In *The Twelfth International Conference on Learning Representations*, 2024.

[3] Wuzhida Bao, Yuting Cao, Yin Yang, Hangjun Che, Junjian Huang, and Shiping Wen. Data-driven stock forecasting models based on neural networks: A review. *Information Fusion*, 113:102616, 2025.

[4] Weilin Cai, Juyong Jiang, Fan Wang, Jing Tang, Sunghun Kim, and Jiayi Huang. A survey on mixture of experts, 2024.

[5] Mohammed Nowaz Rabbani Chowdhury, Meng Wang, Kaoutar El Maghraoui, Naigang Wang, Pin-Yu Chen, and Christopher Carothers. A provably effective method for pruning experts in fine-tuned sparse mixture-of-experts. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 8815–8847. PMLR, 21–27 Jul 2024.

[6] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[7] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018.

[8] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, USA, 2006.

[9] Damai Dai, Chengqi Deng, Chenggang Zhao, R. X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y. K. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models, 2024.

[10] Robert Dawson. How significant is a boxplot outlier? *Journal of Statistics Education*, 19(2), 2011.

[11] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: efficient finetuning of quantized llms. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

[12] Yi Ding and Tianyao Shi. Sustainable llm serving: Environmental implications, challenges, and opportunities : Invited paper. In *2024 IEEE 15th International Green and Sustainable Computing Conference (IGSC)*, pages 37–38, 2024.

[13] Tyrone E. Duncan. On the calculation of mutual information. *SIAM Journal on Applied Mathematics*, 19(1):215–220, 1970.

[14] Faramarz Farhangian, Rafael M.O. Cruz, and George D.C. Cavalcanti. Fake news detection: Taxonomy and comparative study. *Information Fusion*, 103:102140, 2024.

[15] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: scaling to trillion parameter models with simple and efficient sparsity. *J. Mach. Learn. Res.*, 23(1), January 2022.

[16] Shangqian Gao, Ting Hua, Reza Shirkavand, Chi-Heng Lin, Zhen Tang, Zhengao Li, Longge Yuan, Fangyi Li, Zeyu Zhang, Alireza Ganjdanesh, Lou Qian, Xu Jie, and Yen-Chang Hsu. Tomoe: Converting dense large language models to mixture-of-experts through dynamic structural pruning, 2025.

[17] Kai He, Rui Mao, Qika Lin, Yucheng Ruan, Xiang Lan, Mengling Feng, and Erik Cambria. A survey of large language models for healthcare: from data, technology, and applications to accountability and ethics. *Information Fusion*, 118:102963, 2025.

[18] Wei Huang, Yue Liao, Jianhui Liu, Ruifei He, Haoru Tan, Shiming Zhang, Hongsheng Li, Si Liu, and XIAOJUAN QI. MC-moe: Mixture compressor for mixture-of-experts LLMs gains more. In *The Thirteenth International Conference on Learning Representations*, 2025.

[19] Wei Huang, Yangdong Liu, Haotong Qin, Ying Li, Shiming Zhang, Xianglong Liu, Michele Magno, and Xiaojuan Qi. Billm: pushing the limit of post-training quantization for llms. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

[20] Wei Huang, Xudong Ma, Haotong Qin, Xingyu Zheng, Chengtao Lv, Hong Chen, Jie Luo, Xiaojuan Qi, Xianglong Liu, and Michele Magno. How good are low-bit quantized llama3 models? an empirical study. *CoRR*, abs/2404.14047, 2024.

[21] Md. Farhan Ishmam, Md. Sakib Hossain Shovon, M.F. Mridha, and Nilanjan Dey. From image to language: A critical analysis of visual question answering (vqa) approaches, challenges, and opportunities. *Information Fusion*, 106:102270, 2024.

[22] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

[23] Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts, 2024.

[24] Donald E Knuth. Big omicron and big omega and big theta. *ACM Sigact News*, 8(2):18–24, 1976.

[25] Tarald O. Kvålseth. On normalized mutual information: Measure derivations and properties. *Entropy*, 19(11), 2017.

[26] Jaeseong Lee, seung-won hwang, Aurick Qiao, Daniel F Campos, Zhewei Yao, and Yuxiong He. Stun: Structured-then-unstructured pruning for scalable moe pruning, 2024.

[27] Baohao Liao, Christian Herold, Shahram Khadivi, and Christof Monz. ApiQ: Finetuning of 2-bit quantized large language model. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 20996–21020, Miami, Florida, USA, November 2024. Association for Computational Linguistics.

[28] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. In P. Gibbons, G. Pekhimenko, and C. De Sa, editors, *Proceedings of Machine Learning and Systems*, volume 6, pages 87–100, 2024.

[29] Zhiwei Liu, Tianlin Zhang, Kailai Yang, Paul Thompson, Zeping Yu, and Sophia Ananiadou. Emotion detection for misinformation: A review. *Information Fusion*, 107:102300, 2024.

[30] Xudong Lu, Qi Liu, Yuhui Xu, Aojun Zhou, Siyuan Huang, Bo Zhang, Junchi Yan, and Hongsheng Li. Not all experts are equal: Efficient expert pruning and skipping for mixture-of-experts large language

models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6159–6172, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[31] Xinyin Ma, Gongfan Fang, and Xinchao Wang. LLM-pruner: On the structural pruning of large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[32] Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models, 2023.

[33] Zehua Pei, Lancheng Zou, Hui-Ling Zhen, Xianzhi Yu, Wulong Liu, Sinno Jialin Pan, Mingxuan Yuan, and Bei Yu. Cmoe: Fast carving of mixture-of-experts for efficient llm inference, 2025.

[34] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: an adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106, August 2021.

[35] Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. Green ai. *Commun. ACM*, 63(12):54–63, November 2020.

[36] C. Shannon. The lattice theory of information. *Transactions of the IRE Professional Group on Information Theory*, 1(1):105–107, 1953.

[37] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

[38] Noam Shazeer, *Azalia Mirhoseini, *Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017.

[39] Alexander Strehl and Joydeep Ghosh. Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3:583–617, 01 2002.

[40] Qwen Team. Qwen1.5-moe: Matching 7b model performance with 1/3 activated parameters", February 2024.

[41] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

[42] Tina Vartziotis, Maximilian Schmidt, George Dasoulas, Ippolyti Dellatolas, Stefano Attademo, Viet Dung Le, Anke Wiechmann, Tim Hoffmann, Michael Keckeisen, and Sotirios Kotsopoulos. Carbon footprint evaluation of code generation through llm as a service. In André Casal Kulzer, Hans-Christian Reuss, and Andreas Wagner, editors, *2024 Stuttgart International Symposium on Automotive and Engine Technology*, pages 230–241, Wiesbaden, 2024. Springer Fachmedien Wiesbaden.

[43] Roberto Verdecchia, June Sallou, and Luís Cruz. A systematic review of green ai. *WIREs Data Mining and Knowledge Discovery*, 13(4):e1507, 2023.

[44] H. P. Vinutha, B. Poornima, and B. M. Sagar. Detection of outliers using interquartile range technique from intrusion dataset. In Suresh Chandra Satapathy, Joao Manuel R.S. Tavares, Vikrant Bhateja, and J. R. Mohanty, editors, *Information and Decision Sciences*, pages 511–518, Singapore, 2018. Springer Singapore.

[45] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

[46] Zhenxiang Xiao, Yuzhong Chen, Junjie Yao, Lu Zhang, Zhengliang Liu, Zihao Wu, Xiaowei Yu, Yi Pan, Lin Zhao, Chong Ma, Xinyu Liu, Wei Liu, Xiang Li, Yixuan Yuan, Dinggang Shen, Dajiang Zhu, Dezhong Yao, Tianming Liu, and Xi Jiang. Instruction-vit: Multi-modal prompts for instruction learning in vision transformer. *Information Fusion*, 104:102204, 2024.

[47] Yanyue Xie, Zhi Zhang, Ding Zhou, Cong Xie, Ziang Song, Xin Liu, Yanzhi Wang, Xue Lin, and An Xu. Moe-pruner: Pruning mixture-of-experts large language model using the hints from its router, 2024.

[48] Cheng Yang, Yang Sui, Jinqi Xiao, Lingyi Huang, Yu Gong, Yuanlin Duan, Wenqi Jia, Miao Yin, Yu Cheng, and Bo Yuan. MoE-i^2: Compressing mixture of experts models through inter-expert pruning and intra-expert low-rank decomposition. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10456–10466, Miami, Florida, USA, November 2024. Association for Computational Linguistics.

[49] Zhihang Yuan, Yuzhang Shang, and Zhen Dong. PB-LLM: Partially binarized large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

[50] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In Anna Korhonen, David Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics.

[51] Kunpeng Zhang, Feng Zhou, Lan Wu, Na Xie, and Zhengbing He. Semantic understanding and prompt engineering for large-scale traffic data imputation. *Information Fusion*, 102:102038, 2024.

[52] Zeliang Zhang, Xiaodong Liu, Hao Cheng, Chenliang Xu, and Jianfeng Gao. Diversifying the expert knowledge for task-agnostic pruning in sparse mixture-of-experts, 2024.

[53] Xingchen Zou, Yibo Yan, Xixuan Hao, Yuehong Hu, Haomin Wen, Erdong Liu, Junbo Zhang, Yong Li, Tianrui Li, Yu Zheng, and Yuxuan Liang. Deep learning for cross-domain data fusion in urban computing: Taxonomy, advances, and outlook. *Information Fusion*, 113:102606, 2025.

5 An Efficient Green AI Approach to Time Series Forecasting Based on Deep Learning

- Balderas, L., Lastra, M., & Benítez, J. M. (2024). An Efficient Green AI Approach to Time Series Forecasting Based on Deep Learning. *Big Data and Cognitive Computing*, 8(9), 120. https://doi.org/10.3390/bdcc8090120
  - Status: **Published**.
  - Impact Factor (JCR 2023): **3.7**
  - Subject Category: **Computer Science, Theory & Methods**
  - Rank: **25/144**
  - Quartile: **Q1**

AN EFFICIENT GREEN AI APPROACH TO TIME SERIES FORECASTING BASED ON DEEP LEARNING

Luis Balderas
Department of Computer Science
and Artificial Intelligence
DiCITS, iMUDS, DaSCI
University of Granada, Granada, Spain 18071
luisbalru@ugr.es

Miguel Lastra
Department of Software Engineering
DiCITS, iMUDS, DaSCI
University of Granada, Granada, Spain 18071
mlastral@ugr.es

José M. Benítez
Department of Computer Science
and Artificial Intelligence
DiCITS, iMUDS, DaSCI
University of Granada, Granada, Spain 18071
J.M.Benitez@decsai.ugr.es

ABSTRACT

Time series forecasting is undoubtedly a key area in machine learning due to the numerous fields where it is crucial to estimate future data points of sequences based on a set of previously observed values. Deep Learning has been successfully applied to this area. On the other hand, growing concerns about the steady increase in the amount of resources required by deep learning based tools have made Green AI gain traction to move towards making machine learning more sustainable. In this paper we present a deep learning based time series forecasting technique, called GreeNNTSF, which is aimed to reduce the size of the resulting model, reducing therefore the associated computational and energetic costs, without giving up adequate forecasting performance. The methodology based on the ODF2NNA algorithm produces models which not only outperform state-of-the-art techniques in terms of prediction accuracy but also in terms of computational costs and memory footprint. To prove this claim we have tested our technique on a selection of forecasting problems which are commonly used as benchmarks.

Keywords Green AI, dense feed-forward neural network simplification, time series forecasting

1 Introduction

Time series forecasting is a discipline that permeates almost every facet of society, playing a significant role in both industry and academia. It has countless applications in various fields where it is a key player, such as in economics and financial markets [1, 2], climate prediction [3, 4], energy consumption [5, 6], and medical applications [7, 8]. For all these areas, building sufficiently accurate artificial intelligence models is crucial for making future decisions [9].

In the past decade, the advent of deep learning, along with advances in hardware, has made it possible to develop increasingly sophisticated solutions that address both one-step-ahead and multi-horizon time series forecasting. These solutions often rely on hybrid models that combine traditional statistical models with neural networks [10]. With the Transformer architecture introduction [11], the numerous advancements in natural language processing (NLP) [12], computer vision [13] or anomaly detection [14] among others, have spurred the application of this technology to time series forecasting problems, yielding great results due to the similarities between these learning tasks [15, 16].

However, the current trend to enhance the capabilities of these deep learning models focuses on increasing their size, whether in layers, neurons, or complexity, resulting in a rise in the number of parameters and floating-point operations. This leads to models that are very costly to train and maintain in terms of computation and memory footprint. Consequently, the use of these tools require high levels of energy consumption producing a notable adverse impact on the environment.

Contrary to this trend, in recent years, a paradigm called Green AI [17] has gained traction, aiming to continue the evolution of technology while minimizing environmental impact. In this article, we present a methodology to build deep learning models for time series forecasting following the Green-AI paradigm, called GreeNNTSF (Figure 1). Specifically, we propose a semi-automatic method for building efficient neural networks. The ODF2NNA algorithm [18] is one of the fundamental pillars of our methodology, as it automates the network simplification process. We will demonstrate that, compared to state-of-the-art techniques recognized as references for these problems due to their recent publications, it is possible to build solutions that are not only more efficient but also more effective. To achieve this, we will use prediction metrics, closely following the experimentation outlined in the reference articles, as well as efficiency metrics, such as the reduction in the number of model parameters. The objective of this article is to employ the methodology we present, applied to time series forecasting, to construct models that effectively solve widely prevalent real-world problems such as economy, weather, smart cities or medicine with the minimum number of parameters possible.

The structure of the article is as follows: In Section 2, we present the Green AI paradigm and the state-of-the-art methods that use deep learning for time series forecasting. In Section 3, we describe our methodology. In Section 4 we detail our empirical analysis and discussion. Finally, Section 5 contains the conclusions of the work.

2 Related Work

This section is dedicated, first, to introducing the fundamental concepts of Green AI as well as some metrics that justify the need for these techniques. Subsequently, we present some of the latest state-of-the-art methods in time series forecasting. These methods are applied to various sectors or industries, addressing real-world problems.

2.1 Green AI

As stated in [17], the term Green AI refers to AI research aimed at achieving scientific advancements while considering computational costs and aiming to reduce the resources required for development. One of the fundamental principles underpinning this paradigm is the balance between efficiency and effectiveness. To this end, when scientific advancements occur, such as new models solving problems, not only are effectiveness metrics considered, such as accuracy, F1-Score, or precision in classification, among others, but efficiency

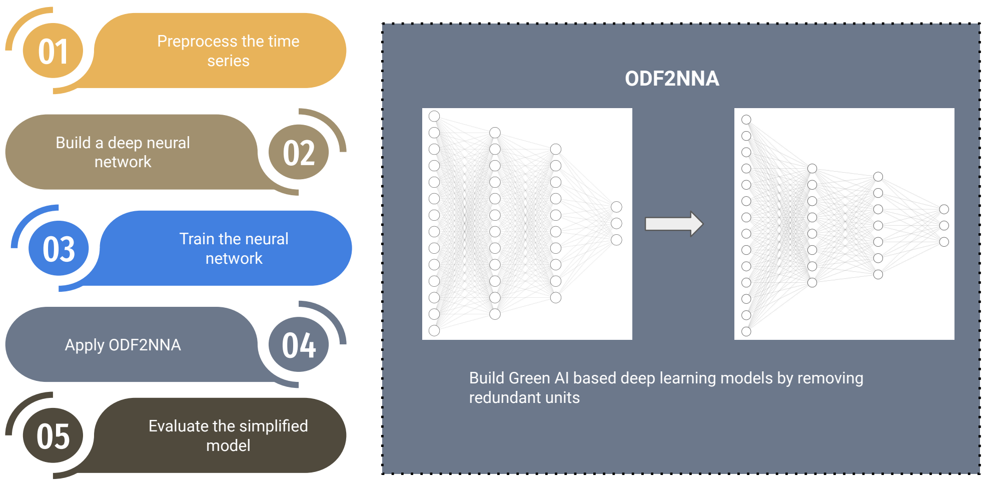

01
Preprocess the time series
Build a deep neural network
02
03
Train the neural network
Apply ODF2NNA
04
05
Evaluate the simplified model
ODF2NNA
Build Green AI based deep learning models by removing redundant units

Figure 1: GreeNNTSF, an end-to-end methodology to build deep learning models based on Green AI paradigm for time series forecasting

measures also take a prominent role. Efficiency metrics focus on measuring the amount of work, translated to computational resources, required for tasks such as training AI models or using them for inference. There are various efficiency measures, including carbon emissions, electricity usage, the time taken to generate the model or the number of parameters comprising it (which also serves as a measure of complexity). Henceforth, we will use the number of parameters as a reference efficiency metric, given that it is a metric that is easily measured (as opposed to others that exceed software capabilities, such as emissions or energy consumption) and is widely accepted and used in the literature.

The development of deep learning, combined with advances in hardware, is revolutionizing industry, academia, and the daily lives of citizens. Since the emergence of the Transformer architecture [11], generative AI models have been created that solve use cases in radically new ways, achieving unprecedented levels of quality and accuracy. However, training these models (and their subsequent use) incurs into very high CO2 emission costs. As shown in [19], training GPT-3 ([20]), with 175 billion parameters, resulted in emissions of 502 tons of CO2, which is eight times the emissions of a vehicle's entire lifespan or nearly 100 times more than what an average human emits in a year. Besides, in terms of energy consumption, [21] have estimated that training GlasPT-3 required an amount of energy equivalent to the consumption of 121 homes in the USA. Nonetheless, these costs are not restricted to training, taking into account that in inference time, each ChatGPT [22] query consumes 260.42 MWh per day [19]. Therefore, we must continue developing AI models that drive technological advancement while minimizing environmental impact. According to [19], the first step in building Green AI models is the optimization of algorithms by designing methods that reduce the resources needed for training or usage. Neural network pruning is one of the most widely used techniques in the literature due to its effectiveness in reducing network complexity and, consequently, associated computational costs.

2.2 Some Deep Learning methods for Time Series Forecasting

Time series forecasting is a widely practiced discipline that applies to numerous academic and industrial problems and has been approached from multiple perspectives. These approaches range from classical methods, such as autoregressive (AR) models, autoregressive moving average (ARMA) models, and autoregressive integrated moving average (ARIMA) models, to machine learning models like Random Forests or Support Vector Machines (SVMs). With the rise of deep learning, neural networks have become the reference solution, whether they are classical dense feed-forward networks or other types, such as recurrent neural networks (RNNs) in their various forms, like Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU), and convolutional neural networks (CNNs). In recent years, the Transformer architecture has been adopted due to the similarity between time series forecasting and language generation models. Intuitively, in both cases, there is a specific order in which the elements of a sentence (or a time series) are expressed to ensure coherence, such that the next word or value in a series largely depends on the preceding ones. This makes it natural to use similar technologies in solving these two seemingly different problems.

This work does not aim to exhaustively cover all approaches to time series forecasting. For an extensive review of recent developments in this field, see [10]. However, we will delve into certain recently published methods that provide different solutions to various problems and have been widely adopted as benchmarks in time series forecasting. These state-of-the-art problems and methods will be used in this work to compare and assess the effectiveness of our approach.

In [23], a new neural network architecture for time series forecasting, called Recurrent Graph Evolution Neural Network (REGENN), is presented, combining graph evolution and deep recurrent learning. Specifically, REGENN consists, in parallel, of a linear component with a feed-forward layer and a non-linear component with an auto-encoder. In the Encoder part, a non-conventional Transformer Encoder is introduced, followed by a Graph Soft Evolution layer. The decoder uses two sequence-to-sequence layers, specifically LSTM. REGENN utilizes three multivariate datasets to measure its effectiveness. The first, related to medicine, comes from Johns Hopkins University and focuses on SARS-CoV-2 [24]. The second, related to climate, is the Brazilian weather dataset. Finally, the third, also from the medical field, corresponds to the 2012 PhysioNet Computing in Cardiology Challenge [25]. As can be intuitively observed in Figures 5, 7, and 9 of the article [23] (exact values are not provided), REGENN achieves better results than other state-of-the-art techniques.

In [26], the problem of multi-step time series forecasting for non-stationary signals with sudden changes is addressed. To this end, the authors propose incorporating temporal and shape criteria in the training process, defining similarity and difference measures for both concepts. Using Dynamic Time Warping (DTW) and the Temporal Distortion Index (TDI), the article presents a new objective function called DILATE for probabilistic forecasting and a prediction framework called STRIPE++. STRIPE++ is based on an autoencoder, where the encoder summarizes the input into a latent vector. This vector is then transformed by the decoder into a trajectory, which facilitates the capture of variations in the upcoming values of the time series. STRIPE++ has been used on synthetic and Electricity datasets mentioned in the paper, which we could not access, and on the publicly available Traffic dataset.

Another paper that designs a deep learning-based approach for predicting electricity demand (Electricity ENTSO-E dataset) is [27]. By combining exponential smoothing (ES) and recurrent neural networks, they present the ES-dRNN method. The ES component dynamically extracts the components of each individual time series. As a result, each time series is deseasonalized, normalized and squashed. After that, a multi-layer RNN, along with a new type of neural unit called a dilated recurrent cell, extracts short- and long-term dependencies within the time series. Additionally, the model generates probability intervals to express the uncertainty of the predictions.

Finally, in [28], the problem of crude oil price forecasting is addressed. Using the West Texas Intermediate (WTI) daily closing oil prices from August 2, 2010, to December 31, 2019, two hybrid methods based on RNN using variational mode decomposition (VMD), sample entropy (SE), and gated recurrent units (GRUs) are presented. They point out that for predicting oil prices, GRU models perform exceptionally well, outperforming LSTM and DNN in terms of accuracy and execution time. Moreover, the combination of VMD, SE, and GRU achieves excellent and robust results.

3 Our proposal

In this article, we propose GreeNNTSF (Figure 1), a Green AI-based methodology for time series forecasting that constructs deep learning models to be more environmentally efficient without losing predictive effectiveness. Contrary to the trend of building increasingly larger networks to solve problems, the core idea of our approach lies in constructing a use case that includes the semi-automatic construction of a feed-forward neural network. This model will then be simplified, as our methodology allows for the automatic discovery of a subnetwork within the original neural network, resulting in a simplified model with fewer parameters without compromising the quality of the predictions.

Concretely, our methodology is an end-to-end process that starts with the problem, expressed through a dataset, that we aim to solve. First, we preprocess the instances and construct the final dataset by applying the sliding window technique, which involves transforming a time series from its original form, with values one after another, into a set of instances for training models. Specifically, by setting a time window, the window slides over the series, taking subsets of the same size from left to right. Each subset becomes an instance for the future model, consisting of the input used for training and the expected values to forecast.

Next, we propose a dense feed-forward neural network. Given that choosing the architecture, including layers and neurons per layer, is a complex task, our methodology favors simple network topologies with a sufficient number of layers and neurons. For simplicity of use, no hyperparameter tuning is required in our methodology.

After training the network, we apply the pruning algorithm ODF2NNA [18], a simplification algorithm for dense feed-forward neural networks based on pruning. It is very easy to use, as it only has one tolerance parameter, ϵ, which indicates the pruning intensity. This way, different versions of a more or less simplified model can be obtained. The algorithm works by measuring the redundancy of each neuron with respect to each instance in a dataset. If the redundancy level exceeds the tolerance, the neuron being evaluated is considered irrelevant to the prediction for that instance and can be removed. In contrast, if it has a low redundancy level, the neuron's contribution is deemed useful and is retained in the simplified model. ODF2NNA is compatible with both classification and regression problems, adapting to canonical metrics such as classification accuracy, F1-Score, or MSE, allowing for quantitative measurement of the predictive capacity of the simplified model. The resulting simplified model, adhering to Green AI standards, is ready to be used as a forecasting engine in production. Algorithm 1 summarizes all the steps that comprise our methodology.

Algorithm 1 Description of the GreeNNTSF methodology

```
function GREENNTSF(ts, $\epsilon$)
    Preprocess the ts. Use sliding window technique to build the definite version of our time series tsf
    Construct a dense feed-forward neural network for addressing the tsf time series forecasting task.
    Train the neural network
    Apply ODF2NNA: build a new model by pruning and refining the original model with $\epsilon$ network reduction level.
    Evaluate the simplified model using established benchmarks.
end function
```

In summary, GreeNNTSF is domain-agnostic and robust technique, as it can tackle real-world time series forecasting problems across various domains, impacting both industry and academia. It generates optimized models following the principles of Green AI. Additionally, it does not impose any restrictions on the problem to be solved nor does it require specific data preprocessing, making it completely flexible and adaptable to the needs of each use case.

To measure the effectiveness of our methodology, we have selected a set of real-world problems from various sectors recognized in the literature as benchmarks for time series forecasting. The empirical analysis conducted is detailed in the following section.

4 Empirical analysis

In this article, a methodology is proposed for building time series forecasting models following the Green AI paradigm. We have thoroughly evaluated our scheme considering real-world problems in fields such as economics, climate, smart cities, and medicine. The section is organized as follows: first, the different datasets that embody the problems to be solved are presented. Next, the metrics used to evaluate the suitability of the solution are introduced. Both the datasets and metrics have been meticulously chosen, following those used as benchmarks in the latest state-of-the-art publications in these fields. Subsequently, more details are provided about the procedure for constructing the neural networks and their subsequent simplification. Finally, the results obtained are discussed.

4.1 Real-world problems

For this work, datasets representing real-world problems have been chosen. Due to their diversity and the different approaches published in the state of the art for their resolution, these datasets constitute an appropriate benchmark for evaluating novel models. As mentioned, the problems described here are of interest to both industry and academia, being associated with economics, smart cities, medicine, or climate. The datasets are as follows:

- **Sars-Cov-2:** Dataset provided by the John Hopkins University [24]. It consists of three variables measured over 120 days in 188 different countries, representing the first four months of the pandemic. The features are the number of recovered patients, the number of infected patients, and the number of deaths.

- **Brazilian Weather:** Dataset generated by collecting data from 253 sensors over 1095 days [23]. It consists of four variables: minimum temperature, maximum temperature, solar radiation, and rainfall.

- **PhysioNet:** Dataset created for the 2012 Physionet Computing in Cardiology Challenge [25]. It consists of nine features collected over 48 hours from 11,988 ICU patients. The features are non-invasive diastolic arterial blood pressure, non-invasive systolic arterial blood pressure, invasive diastolic arterial blood pressure, non-invasive mean arterial blood pressure, invasive mean arterial blood pressure, urine output, heart rate, and weight.

- **WTI:** One of the most widely used benchmarks in the global oil market, available from the US Energy Information Administration [29]. For this experiment, 2446 WTI closing prices from August 2, 2010, to December 31, 2019, are included.

- **Traffic:** Dataset composed of the road occupancy rate from the California Department of Transportation, measured every hour over 48 months from 2015-2016 [26].

- **Electricity:** Includes the electricity demand of 35 European countries between 2016 and 2018, available at ENTSO-E webpage [30].

To ensure the experiments are comparable, the same training and test partitions indicated in each of the publications used for comparison have been employed. For full details on these partitions, please refer to the original publications.

4.2 Metrics

The metrics, like the datasets, have been rigorously selected according to those used in the articles presenting the techniques with which our time series forecasting method is compared. Most of them are classic in this discipline, such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE) or R^2. Others are somewhat less conventional, such as Mean Squared Logarithmic Error (MSLE) and Median Absolute Percentage Error (MdAPE) [31, 32]. Finally, some metrics are defined ad-hoc in the reviewed articles, such as DILATE, Distortion Loss including Shape (based on Dynamic Time Warping or DTW [33]) and Time (based on Time Distortion Index or TDI for temporal misalignment estimation [34]). DILATE is defined as follows. If $\hat{y}$ is the prediction and y^* the ground truth

$$L_{\text{DILATE}}(\hat{y}, y^*) = \alpha L_{\text{Shape}}(\hat{y}, y^*) + (1-\alpha) L_{\text{Time}}(\hat{y}, y^*)$$

with

$$L_{\text{Shape}} = \text{DTW}(\hat{y}, y^*)$$

$$L_{\text{Time}} = \text{TDI}(\hat{y}, y^*)$$

More details can be found in [26]. Table 1 summarizes the correspondence between datasets and error metrics used to compare different time series forecasting methods.

To apply the mentioned metrics, implementations available in specialized time series software packages, such as *sktime* [35, 36], are used when available. For DILATE, the implementation provided by the authors alongside their article is used. By using standard implementations and rigorously applying the criteria presented in the original works, we ensure that the comparison is fair and useful for drawing unbiased conclusions.

Table 1: Metrics used for each learning task or dataset. It helps in understanding how each dataset's forecasting performance is assessed using various error metrics, ensuring a comprehensive evaluation of the proposed models against existing state-of-the-art techniques.

Learning Task / Dataset	**Metrics**
SARS-CoV-2	MAE, RMSE, MSLE
PhysioNet	MAE, RMSE, MSLE
Brazilian Weather	MAE, RMSE, MSLE
Traffic	MSE, DILATE
Electricity	MAPE, RMSE, MdAPE
WTI	RMSE, MAE, MAPE, R^2

4.3 Dense Feed-Forward Neural Network definition and simplification

One of the major strengths of our methodology for time series forecasting is the simplicity of the networks we use. Unlike state-of-the-art reference methods, our neural networks are dense feed-forward networks with a relatively simple structure. Our approach is designed to be versatile, accommodating problems of very different natures. Consequently, the architecture and number of parameters of each network will vary depending on the specific problem. However, the overall procedure remains consistent: a dense feed-forward neural network is designed to solve a supervised learning task. This network is then simplified using ODF2NNA [18], resulting in a smaller network without losing accuracy—sometimes even improving it. After applying the algorithm, the simplified network is compared with state-of-the-art techniques to evaluate whether our solution is competitive.

Below are the different neural network architectures and configurations for each of the datasets described in Section 4.1. As mentioned before, simple yet effective neural networks architectures are chosen as part of our process in order to be simplified afterwards. Next, we present the original architecture proposed (before simplification) for each of the addressed problems:

- SARS-CoV-2: Neural Network with 6 layers: 300-200-100-50-20-3
- Brazilian Weather: Neural Network with 7 layers: 300-200-100-50-25-12-4
- PhysioNet: Neural Network with 7 layers: 500-300-200-100-50-20-9
- WTI: Neural Network with 6 layers: 300-200-100-50-20-1
- Traffic: Neural Network with 6 layers: 300-200-100-50-20-1
- Electricity: Neural Network with 6 layers: 500-250-200-100-50-24

4.4 Results and discussion

In this section, we present the results obtained for each of the experiments conducted. The objective is to assess the extent to which our approach based on Green AI is more suitable for time series forecasting problems. The results shown in the tables correspond to those generated by the models after the complete methodology has been applied. This means that the network is trained, simplified, and finally used to predict on the test set for each of the learning tasks. In all experiments, we have strictly followed the guidelines of the original articles for data preprocessing as well as the partitioning of the data by applying the sliding window technique for executing the corresponding trainings and validations. All approaches agree on using min-max normalization to achieve better training results. In every case, the training set was used to adjust the normalization, which was then applied to the test set. Finally, the model outputs were transformed back to their original scale before calculating the performance metrics.

The results of each experiment are presented and discussed below. Since we aggregate real-world problems addressed in various state-of-the-art articles, for the sake of clarity, we distinguish each experiment in a separate section. If multiple experiments are covered in the same reference article, they are addressed in the same section due to the similarity in the techniques used.

4.4.1 Experiment on SARS-CoV-2, Brazilian Weather and PhysioNet datasets

In this section, we present the results for experiments SARS-CoV-2, Brazilian Weather and PhysioNet datasets. For all of them, the most recent state-of-art technique is REGENN [23].

The results of the experiment on SARS-CoV-2 dataset are shown on Table 2. Given that the dataset contains information from 188 different countries, in contrast to REGENN, which proposes a single model, our approach involves creating a model for each country to generate results that are much more tailored to the specific circumstances of each region. Therefore, the results shown in the table represent, in our case, the average of the metrics obtained by the ensemble model for each test set.

With respect to the data preparation, a window size of 7 days is used for the training set, with 7 days reserved for validation, and finally, the goal is set to predict the values for the next 14 days. This 7-7-14 configuration follows medical expert knowledge regarding the virus incubation period, as it is stated in the original article.

The original work includes, in Figure 5, a graphical comparison of the reference method against other time series forecasting models, such as Linear, Autoregressive, LSTM, GRU, Decision Tree, KNeighbors, Random Forest or Exponential Smoothing approaches among others. Although the exact metric values cannot be precisely extracted, it is clear that REGENN outperforms the other techniques for MAE, RMSE and MSLE metrics. However, as summarized in the Table 2, we obtain a much more compact and efficient ensemble, given the fact that it achieves better results in both MAE (19% of reduction) and RMSE (81% of reduction), showing a notable reduction compared to the other experiments and especially against REGENN, which is the reference. For MSLE, we also obtain competitive results, although REGENN achieves the best performance.

Table 2: Results for Sars-CoV-2 experiment. MAE stands for Mean Absolute Error, RMSE stands for Root Mean Squared Error, and MSLE stands for Mean Squared Logarithmic Error. Although REGENN performs better in terms of the MSLE metric, our approach is much more accurate when considering metrics such as MAE and RMSE, with the latter being notably lower compared to other state-of-the-art methods. Best results are shown in bold.

Algorithm	**MAE**	**RMSE**	**MSLE**
REGENN [23]	165.41	915.92	**0.05**
GreeNNTSF	**134.13**	**174.25**	0.1

The results of the experiment on Brazilian Weather dataset are shown on Table 3. In this case, the dataset contains information from 253 different sensors. Thus, a model for each sensor is built to obtain more accurate predictions. With respect to the data preparation, following the seasonality of weather data, a 84 days window size is used, reserving 28 days for validation and predicting the next 56 days. Alike the last experiment, the results shown on table represent, in our case, the average of the metrics obtained by the ensemble model for each test set. GreeNNTSF obtains the best results for all metrics, obtaining a 13.5% of reduction for MAE, 40% of reduction for RMSE and 17% for MSLE.

Table 3: Results for Brazilian Weather experiment. MAE stands for Mean Absolute Error, RMSE stands for Root Mean Squared Error, and MSLE stands for Mean Squared Logarithmic Error. In this case, our methodology obtains the best results for all metrics. Best results are shown in bold.

Algorithm	MAE	RMSE	MSLE
REGENN	1.92	4.86	0.28
GreeNNTSF	**1.68**	**2.91**	**0.23**

Finally, Table 4 presents the results of PhysioNet dataset. To learn the behavior of the 11,988 ICU patients represented, an ensemble of 11,988 models is generated. The window size, following the guidelines of the original article, is set to 12 hours, with 6 hours reserved for validation, and the objective is to predict the next 6 hours of the dataset. Unlike previous datasets, a much shorter time period is established to track the patient hour by hour, rather than making long-term predictions. It can be observed that the results of our methodology are more competitive than the reference algorithm and the other proposed approaches (19% reduction for MAE, 14% for RMSE and 36% for MSLE).

Table 4: Results for PhysioNet experiment. MAE stands for Mean Absolute Error, RMSE stands for Root Mean Squared Error, and MSLE stands for Mean Squared Logarithmic Error. Again, GreeNNTSF obtains the best results for all metrics. Best results are shown in bold.

Algorithm	MAE	RMSE	MSLE
REGENN	18.22	47.31	1.37
GreeNNTSF	**14.77**	**40.52**	**0.88**

4.4.2 Experiment on the WTI dataset

Given that WTI crude oil is one of the primary price benchmarks in the global oil market, accurately predicting its value provides a significant competitive advantage, making it a highly relevant use case in economics. For setting up the experimentation from a data perspective, 2346 closing price values of WTI are taken, with the last 100 values reserved as the test set. As mentioned in [28], the data is smooth, autocorrelated, and does not follow a normal distribution, containing a substantial amount of noise.

As shown on Table 6, [28] proposes various approaches that combine variational modal decomposition, sample entropy, and gated recurrent units. There are no details provided regarding the number of parameters these different models have or indications of their complexity, so we cannot compare the results in that aspect. Nonetheless, we can see that GreeNNTSF yields better results in terms of RMSE, MAPE, and R^2 metrics, although the VMD-GRU configuration achieves a better MAE value. Finally, it can be verified that our method is effective, as there is another approach based on dense neural networks that yields significantly higher error rates (two to three times higher).

4.4.3 Experiment on the Traffic dataset

In the experiment with Traffic dataset, the ratio of road occupancy in California is measured hourly. Following the guidelines of the reference article [26], the time series, consisting of 17544 values, is divided by using 60% for training, the next 20% for validation, and the remaining 20% for testing. The objective is to predict the next 24 values, that is, the occupancy rate for the next day measured hour by hour, given the previous 168 points, which represent the traffic from the previous week.

Table 5: Results for predicting the WTI price. RMSE stands for Root Mean Squared Error, MAE stands for Mean Absolute Error, MAPE stands for Mean Absolute Percentage Error, and R^2 stands for the correlation coefficient. As shown, GreeNNTSF achieves better results across all metrics except for MAE, where VMD-GRU has the best result. Best results are shown in bold.

Algorithm	**RMSE**	**MAE**	**MAPE**	R^2
LSTM	1.402	0.835	1.646	0.685
GRU	1.268	0.816	1.43	0.742
DNN	1.481	0.962	1.723	0.626
BPNN	1.666	1.213	2.131	0.554
VDM-LSTM ([28])	0.681	0.462	0.815	0.825
VMD-GRU	0.654	**0.429**	0.756	0.931
VMD-SE-LSTM (univ)	0.817	0.556	0.982	0.892
VMD-SE-LSTM (multiv)	0.796	0.532	0.942	0.898
VMD-SE-GRU (univ)	0.669	0.451	0.793	0.928
VMD-SE-GRU (multiv)	0.783	0.523	0.924	0.901
GreeNNTSF	**0.612**	0.498	**0.587**	**0.932**

Table 6: Results for predicting the WTI price. RMSE stands for Root Mean Squared Error, MAE stands for Mean Absolute Error, MAPE stands for Mean Absolute Percentage Error, and R^2 stands for the correlation coefficient. As shown, GreeNNTSF achieves better results across all metrics except for MAE, where VMD-GRU has the best result. Best results are shown in bold.

Table 7: Results for the Traffic learning task. MSE stands for Mean Squared Error, and DILATE is introduced in Section 3.2. The lower the value, the better the model's performance. As shown, our approach yields better results than the other methods, including STRIPE++. Best results are shown in bold.

Algorithm	**MSE**	**DILATE**
STRIPE++ [26]	6.7	14.1
Diverse DPP [37]	6.9	14.7
Deep AR [38]	6.6	16.9
Nbeats [39]	17.1	17.8
Variety loss [40]	7.9	15.9
GreeNNTSF	**4.78**	**12.46**

Table 8: Results for the Traffic learning task. MSE stands for Mean Squared Error, and DILATE is introduced in Section 3.2. The lower the value, the better the model's performance. As shown, our approach yields better results than the other methods, including STRIPE++. Best results are shown in bold.

In [26], the authors do not provide details on the number of parameters of STRIPE++, although they explain that it is an autoencoder and specifies the number of GRU units and dense layers. Therefore, once again, we cannot compare the number of parameters between our approach and their algorithm. The table displays the prediction metric results. For both MSE, a classic forecasting metric, and DILATE, introduced in the reference, GreeNNTSF achieves better results compared to other state-of-the-art methods.

4.4.4 Experiment on the Electricity dataset

The experiment on Electricity dataset serves as a clear example of an applied problem in economics. The reference algorithm ES-eRNN [27] proposes a complex model that combines Exponential Smoothing and recurrent neural networks, with approximately 229000 parameters. Additionally, note that there are two proposals, ES-dRNN with a simple model and ES-dRNNe with an ensemble. The results are shown on Table 10. It can be observed that the reference model performs better than the other machine learning models proposed in the original work. However, our methodology achieves even better results across all three metrics (5% reduction for MAPE, 20% reduction for RMSE and 54% for MdAPE with respect to ES-dRNNe) with a simplified model. Initially starting with 280000 parameters, we reduced it to 184207, which represents a 35% reduction. Therefore, we not only generate a model with better performance but also make it more efficient, aligning with the principles of Green AI.

Table 9: Results for predicting the Electricity demand. RMSE stands for Root Mean Squared Error, MAE stands for Mean Absolute Error, MAPE stands for Mean Absolute Percentage Error, and MdAPE stands for Median Absolute Percentage Error. As we can see, our methodology outperforms all the other state-of-art methods in every metric. Best results are shown in bold.

Algorithm	MAPE	RMSE	MdAPE
Naive	5.08	704.34	4.84
ARIMA ([41])	3.3	475.09	3.01
ES ([41])	3.11	439.26	2.88
Prophet ([42])	4.53	619.39	4.32
K-Nnw ([41])	2.5	335.13	2.31
FNM ([41])	2.5	334.08	2.3
N-WE ([41])	2.49	332.49	2.28
GRNN ([43])	2.48	332.91	2.28
MLP ([43])	3.05	419.01	2.78
SVM ([44])	2.55	357.24	2.29
LSTM ([45])	2.76	381.76	2.57
ANFIS ([46])	3.65	507.08	3.17
MTGNN ([47])	2.99	405.18	2.74
ES-dRNN ([27])	2.36	326.06	2.15
ES-dRNNe ([27])	2.23	306.94	2.02
GreeNNTSF	**2.16**	**251.12**	**0.94**

Table 10: Results for predicting the Electricity demand. RMSE stands for Root Mean Squared Error, MAE stands for Mean Absolute Error, MAPE stands for Mean Absolute Percentage Error, and MdAPE stands for Median Absolute Percentage Error. As we can see, our methodology outperforms all the other state-of-art methods in every metric. Best results are shown in bold.

5 Conclusions

Time series forecasting is a discipline with enormous applications in both industry and academia. The literature presents numerous approaches that address problems of very different natures. The advent of deep learning has led to increasingly powerful solutions. However, the models generated tend to be very large, which, despite being very accurate, entail significant computational, energy, and environmental costs. Green AI is a paradigm that aims to introduce models with high predictive capacity that are more resource-efficient. In this work, we propose GreeNNTSF, an end-to-end methodology to build Green AI time series forecasting models that address real-world problems in economics, healthcare, smart cities, and climate. Through meticulous experimentation, including different datasets and forecasting metrics, we demonstrate that the models generated by our methodology are not only more efficient by reducing the number of parameters in their architecture but also more effective in prediction, outperforming other state-of-the-art approaches in almost all metrics.

Acknowledgment

This research has been partially supported by the Spanish Ministry of Economy, Industry and Competitiveness (PID2020-118224RB-I00 and PID2023-151336OB-I00) with the co-financing of the European Union (FEDER).

References

[1] Cheng Zhang, Nilam Nur Amir Sjarif, and Roslina Ibrahim. Deep learning models for price forecasting of financial time series: A review of recent advancements: 2020–2022. *WIREs Data Mining and Knowledge Discovery*, 14(1):e1519, 2024.

[2] Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. Financial time series forecasting with deep learning : A systematic literature review: 2005–2019. *Applied Soft Computing*, 90:106181, 2020.

[3] Soumik Ray, Soumitra Sankar Das, Pradeep Mishra, and Abdullah Mohammad Ghazi Al Khatib. Time series sarima modelling and forecasting of monthly rainfall and temperature in the south asian countries. *Earth Systems and Environment*, 5(3):531–546, Sep 2021.

[4] Yuchuan Lai and David A. Dzombak. Use of integrated global climate model simulations and statistical time series forecasting to project regional temperature and precipitation. *Journal of Applied Meteorology and Climatology*, 60(5):695 – 710, 2021.

[5] Chirag Deb, Fan Zhang, Junjing Yang, Siew Eang Lee, and Kwok Wei Shah. A review on time series forecasting techniques for building energy consumption. *Renewable and Sustainable Energy Reviews*, 74:902–924, 2017.

[6] Corne van Zyl, Xianming Ye, and Raj Naidoo. Harnessing explainable artificial intelligence for feature selection in time series energy forecasting: A comparative analysis of grad-cam and shap. *Applied Energy*, 353:122079, 2024.

[7] Bahman Rostami-Tabar and Rob J. Hyndman. Hierarchical time series forecasting in emergency medical services. *Journal of Service Research*, 0(0):10946705241232169, 0.

[8] Stanley Suan You Lim, Siao-Leu Phouratsamay, Zakaria Yahouni, and Eric Gascard. Medicine consumption demand forecasting in french hospitals using seasonal auto-regressive integrated moving average (sarima) models. In *2024 International Conference on Control, Automation and Diagnosis (ICCAD)*, pages 1–6, 2024.

[9] Christoph Bergmeir and José M. Benítez. On the use of cross-validation for time series predictor evaluation. *Information Sciences*, 191:192–213, 2012. Data Mining for Software Trustworthiness.

[10] Xinhe Liu and Wenmin Wang. Deep time series forecasting models: A comprehensive survey. *Mathematics*, 12(10), 2024.

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[12] Tong Xiao and Jingbo Zhu. Introduction to transformers: an nlp perspective, 2023.

[13] Sonain Jamil, Md. Jalil Piran, and Oh-Jin Kwon. A comprehensive survey of transformers for computer vision. *Drones*, 7(5), 2023.

[14] Mingrui Ma, Lansheng Han, and Chunjie Zhou. Research and application of transformer based anomaly detection model: A literature review, 2024.

[15] Sabeen Ahmed, Ian E. Nielsen, Aakash Tripathi, Shamoon Siddiqui, Ravi P. Ramachandran, and Ghulam Rasool. Transformers in time-series analysis: A tutorial. *Circuits, Systems, and Signal Processing*, 42(12):7433–7466, Dec 2023.

[16] Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. Transformers in time series: A survey, 2023.

[17] Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. Green ai. *Commun. ACM*, 63(12):54–63, nov 2020.

[18] Luis Balderas, Miguel Lastra, and José M. Benítez. Optimizing dense feed-forward neural networks. *Neural Networks*, 171:229–241, 2024.

[19] Verónica Bolón-Canedo, Laura Morán-Fernández, Brais Cancela, and Amparo Alonso-Betanzos. A review of green artificial intelligence: Towards a more sustainable future. *Neurocomputing*, 599:128096, 2024.

[20] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[21] David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. Carbon emissions and large neural network training, 2021.

[22] OpenAI. ChatGPT: A Large-Scale Generative Model for Open-Domain Chat. `https://github.com/openai/gpt-3`, 2021.

[23] Gabriel Spadon, Shenda Hong, Bruno Brandoli, Stan Matwin, Jose F. Rodrigues-Jr, and Jimeng Sun. Pay attention to evolution: Time series forecasting with deep graph-evolution learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):5368–5384, 2022.

[24] Ensheng Dong, Hongru Du, and Lauren Gardner. An interactive web-based dashboard to track covid-19 in real time, May 2020.

[25] Ikaro Silva, George Moody, Daniel J Scott, Leo A Celi, and Roger G Mark. Predicting In-Hospital mortality of ICU patients: The PhysioNet/Computing in cardiology challenge 2012. *Comput Cardiol (2010)*, 39:245–248, 2012.

[26] Vincent Le Guen and Nicolas Thome. Deep time series forecasting with shape and temporal criteria. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, 45(1):342–355, JAN 1 2023.

[27] Slawek Smyl, Grzegorz Dudek, and Pawel Pelka. Es-drnn: A hybrid exponential smoothing and dilated recurrent neural network model for short-term load forecasting. *IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS*, 2023 AUG 31 2023.

[28] Shiqi Zhang, Jing Luo, Shuyuan Wang, and Feng Liu. Oil price forecasting: A hybrid gru neural network based on decomposition–reconstruction methods. *Expert Systems with Applications*, 218:119617, 2023.

[29] Homepage - U.S. Energy Information Administration (EIA) — eia.doe.gov. www.eia.doe.gov. [Accessed 20-05-2024].

[30] Power Statistics — entsoe.eu. www.entsoe.eu/data/power-stats/. [Accessed 14-05-2024].

[31] Jan G. De Gooijer and Rob J. Hyndman. 25 years of time series forecasting. *International Journal of Forecasting*, 22(3):443–473, 2006. Twenty five years of forecasting.

[32] Rob J. Hyndman and Anne B. Koehler. Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4):679–688, 2006.

[33] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, 1978.

[34] Laura Frías-Paredes, Fermín Mallor, Teresa León, and Martín Gastón-Romeo. Introducing the temporal distortion index to perform a bidimensional analysis of renewable energy forecast. *Energy*, 94:180–194, 2016.

[35] Markus Löning, Anthony Bagnall, Sajaysurya Ganesh, Viktor Kazakov, Jason Lines, and Franz J. Király. sktime: A unified interface for machine learning with time series, 2019.

[36] Franz Király, Markus Löning, Tony Bagnall, Matthew Middlehurst, Sajaysurya Ganesh, Martin Walter, George Oastler, Anirban Ray, Jason Lines, ViktorKaz, Lukasz Mentel, Benedikt Heidrich, Sagar Mishra, chrisholder, Daniel Bartling, Leonidas Tsaprounis, RNKuhns, Ciaran Gilbert, Mirae Baichoo, Hazrul Akmal, Guzal, Taiwo Owoseni, Patrick Rockenschaub, eenticott shell, Sami Alavi, Armaghan, jesellier, Kishan Manani, and Patrick Schäfer. sktime/sktime: v0.31.0, 2024.

[37] Ye Yuan and Kris M. Kitani. Diverse trajectory forecasting with determinantal point processes. In *International Conference on Learning Representations*, 2020.

[38] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3):1181–1191, 2020.

[39] Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations*, 2020.

[40] Luca Thiede and Pratik Brahma. Analyzing the variety loss in the context of probabilistic trajectory prediction. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9953–9962, 2019.

[41] Grzegorz Dudek. Pattern similarity-based methods for short-term load forecasting – part 2: Models. *Applied Soft Computing*, 36:422–441, 2015.

[42] Sean J. Taylor and Benjamin Letham. Forecasting at scale. *The American Statistician*, 72(1):37–45, 2018.

[43] Grzegorz Dudek. Neural networks for pattern-based short-term load forecasting: A comparative study. *Neurocomputing*, 205:64–74, 2016.

[44] Paweł Pełka. Pattern-based forecasting of monthly electricity demand using support vector machine. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2021.

[45] Paweł Pełka and Grzegorz Dudek. Pattern-based long short-term memory for mid-term electrical load forecasting. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2020.

[46] Paweł Pełka and Grzegorz Dudek. Neuro-fuzzy system for medium-term electric energy demand forecasting. In Leszek Borzemski, Jerzy Świątek, and Zofia Wilimowska, editors, *Information Systems Architecture and Technology: Proceedings of 38th International Conference on Information Systems Architecture and Technology – ISAT 2017*, pages 38–47, Cham, 2018. Springer International Publishing.

[47] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, page 753–763, New York, NY, USA, 2020. Association for Computing Machinery.

6 On MRI-Based Central Nervous System Tumor Differentiation using Deep Neural Networks and Conformal Predictors

- Balderas, L., Pérez, F.J., Moreno de Castro, M., Lastra, M., Martínez, J.P., Lainez-Ramos, A.J., Arauzo, A., & Benítez, J.M. (2025). On MRI-Based Central Nervous System Tumor Differentiation using Deep Neural Networks and Conformal Predictors
  - Status: **Submitted**.
  - Impact Factor (JCR 2023): **4.9**
  - Subject Category: **Computer Science, Theory & Methods**
  - Rank: **0/144**
  - Quartile: **Q1**

On MRI-Based Central Nervous System Tumor Differentiation using Deep Neural Networks and Conformal Predictors

Luis Balderas
Department of Computer Science
and Artificial Intelligence
Instituto de Investigación Biosanitaria de Granada (ibs.Granada)
DiCITS, iMUDS, DaSCI
University of Granada, Granada, Spain 18071

Francisco Javier Pérez García
Instituto de Investigación Biosanitaria de Granada (ibs.Granada)
Department of Radiology
Hospital Universitario "Virgen de las Nieves"

María Moreno de Castro
Department of Computer Science
and Artificial Intelligence
DiCITS, iMUDS, DaSCI
University of Granada, Granada, Spain 18071

Miguel Lastra
Department of Software Engineering
DiCITS, iMUDS, DaSCI
University of Granada, Granada, Spain 18071

José Pablo Martínez Barbero
Instituto de Investigación Biosanitaria de Granada (ibs.Granada)
Department of Radiology
Hospital Universitario "Virgen de las Nieves"

Antonio Jesús Láinez-Ramos Bossini
Instituto de Investigación Biosanitaria de Granada (ibs.Granada)
Department of Radiology
Hospital Universitario "Virgen de las Nieves"
Department of Human Anatomy and Embryology
University of Granada, Granada, Spain 18071

Antonio Arauzo-Azofra
Department of Rural Engineering
DiCITS, iMUDS
University of Córdoba, Córdoba, Spain 14071

José M. Benítez
Department of Computer Science
Instituto de Investigación Biosanitaria de Granada (ibs.Granada)
and Artificial Intelligence
DiCITS, iMUDS, DaSCI
University of Granada, Granada, Spain 18071

Abstract

Central nervous system tumors (CNST) rank among the top 10 causes of cancer-related deaths worldwide. Gliomas compose a group of CNST demanding precise classification for effective treatment. While several previous studies have considered the application of Artificial Intelligence to differentiate between high and low-grade gliomas (HGG and LGG, respectively), all of them lack a reliable confidence measurement. We propose a deep learning-based technique for differentiating HGG and LGG using MRI imaging, which incorporates two novel approaches. First, we represent perfusion sequences as time series, which serve as inputs for a Deep Neural Network (DNN) classifier. This classifier is trained to distinguish between low-grade glioma (LGG) or high-grade glioma (HGG). Second, we employ Conformal Prediction to calibrate the results, ensuring they include the true category with a high probability (90%). The second problem addressed is gliomas grading according to WHO scale. LGG is decomposed into grades I and II, while HGG is subdivided into grades III and IV. A salient feature of this classification problems is their high ratio of unbalancedness. Again DNN achieve the highest performance. Our approach has been rigorously tested to assess its performance. Our experimental results demonstrate that our deep neural network not only provides accurate outputs but also ensures trustworthy predictions.

***K*eywords** Glioma differentiation, CNN, time series classification, conformal prediction

1 Introduction

Central nervous system (CNS) tumors represent a significant health burden worldwide. Among these, gliomas are one of the most common and aggressive types of brain tumors. Together with the fact that the incidence and mortality of glioma increase significantly with advancing age and considering the aging trend of occidental society, the burden associated with glioma may soon become a major challenge for healthcare systems and societes [33].

Brain gliomas are often difficult to treat due to their invasive nature and location, which limits surgical options and complicates treatment [56]. In addition, precise differentiation of glioma subtypes is essential for improving patient outcomes because timely and tailored treatments increase the chances of patient survival [50]. The current gold standard technique to classify brain gliomas is based on a combination of histological and molecular/genetic analysis [39], for which brain biopsy is required. However, this is an invasive technique that entails patient risks, including intracranial hemorrhage, infection, or neurological deficits [46].

Magnetic Resonance Imaging (MRI) is a non-invasive imaging technique widely used for diagnosing and monitoring CNS tumors, including gliomas. It uses strong magnetic fields and radiofrequency waves to create detailed images of the brain, allowing for the detection of abnormalities in soft tissues. Perfusion MRI makes use of exogenous and endogenous tracers for assessing blood flow dynamics within tissues by tracking the passage of a contrast agent (e.g. gadolinium) or using arterial spin labeling (a non-contrast technique). The resulting images can be analyzed to generate perfusion curves, which thus represent changes in signal intensity or contrast concentration over time in the region of interest. Perfusion MRI is a promising technique for tumor characterization and for the evaluation of neurodegenerative diseases [35].

On the other hand, recent advances in several Artificial Intelligence fields have boosted their application in medical diagnosis. Deep Learning (DL) is a subfield within Machine Learning, that in turn is a wide subfield within Artificial Intelligence, that deals with large Artificial Neural Networks (ANNs), usually networks with many layers (deep). ANNs began as models mimicking the way the human brain processes information and they can be used to solve a wide variety of problems, including image recognition. Techniques in DL excel at extracting meaningful insights from vast amounts of data. By leveraging our capacity to handle complex data, deep learning enables breakthroughs in many tasks and it has become a very important piece in many modern technological innovations.

Deep learning has demonstrated significant potential in medical imaging, including MRI processing tasks like reconstruction, contrast conversion, and quality enhancement. Perfusion MRI techniques have also advanced, enabling clinical applications that visualize tumor angiogenesis, blood-brain barrier breakdown, and quantifiable absolute cerebral blood perfusion. These techniques are used to study various diseases (e.g., brain tumors, strokes, Alzheimer's) and organs (e.g., brain, breast, prostate). However, traditional tracer-kinetic modeling for parametric maps is time-intensive and constrained by physical assumptions. Deep learning offers a faster, end-to-end, model-free approach for processing perfusion MRI, reducing processing time significantly while improving clinical feasibility [15].

Another relevant technical procedure that is increasingly receiving attention within the Machine Learning area is the application of Conformal Prediction (CP). Conformal Prediction is a statistical framework that provides reliable uncertainty estimates for machine learning predictions. Unlike traditional predictive models that give a single prediction, conformal prediction generates prediction sets or intervals, which are guaranteed to contain the true outcome with a user-specified confidence level (e.g., 95%). This is achieved without making strong assumptions about the underlying data distribution, making it a flexible and robust approach. The key idea is to use past experience to assess how well a model's predictions conform to the observed outcomes, based on a measure of prediction error. By calibrating the model on a separate dataset, conformal prediction adjusts the prediction sets to ensure they meet the desired confidence level [48]

This paper presents a novel approach based on perfusion curves to classifying gliomas into four categories and the use of conformal predictors to ensure guaranteed confidence levels in the results of this classification. Those are the two most relevant highlights of the research, namely, first, considering perfusion curves as time series and, then, building time series classifiers for gliomas diagnosis, and second, applying conformal calibration to enhance the confidence on the classifiers outcomes.

2 State-of-the-art

In this section, a review of recent literature related to the topics of the reported research is developed. Since the proposed methodology is grounded in deep learning and incorporates uncertainty quantification, this section is bifurcated into two parts: deep learning methodologies for the diagnosis of tumors, with a specific focus on gliomas; and the employment of conformal predictors as a mechanism for quantifying uncertainty in cancer detection.

2.1 Deep Learning-Based methods for Brain Tumor Diagnosis

MRI is a cornerstone in contemporary tumor diagnosis. Its non-invasive nature enables swift and efficient patient evaluation. When combined with the potent predictive capabilities and pattern recognition abilities inherent in deep learning models, MRI and artificial intelligence have become ubiquitous in scientific research. Moreover, as stated in [2], although deep learning techniques have demonstrated exceptional performance in voxel-wise evaluation, it is apparent that gadolinium-based contrast agents continue to be indispensable for the detection of smaller lesions, which often pose challenges for deep learning models. In conclusion, the synergistic integration of deep learning and gadolinium-enhanced imaging leads to the development of highly robust and adaptable predictive systems [8].

For glioma diagnosis and differentiation specifically, a multitude of approaches have been explored. A pivotal aspect involves denoising images to augment resolution and quality. Lee *et al.* [32] introduced a U-Net-based method for signal enhancement, thereby improving the differentiation of WHO glioma grades. Furthermore, Singh *et al.* [49] highlighted various methodologies that integrate deep learning and radiomics to extract radiomic features for targeted learning tasks [11, 13]. Notably, more advanced techniques leverage deep learning to generate synthetic or augmented radiomic features [29] or even harness automated segmentation to generate higher-quality data inputs. Besides, as highlighted in [47], deep learning methodologies have been employed to predict the genetic profile of LGGs and HGGs, incorporating markers such as IDH, PTEN, and 1p19q-codeletion [18, 12, 36, 41, 58, 42, 19].

Artificial intelligence has been used to exploit information from perfusion MRI in the context of brain tumors. For instance, Crisi *et al.* evaluated 92 quantitative features on rCBV and rCBF maps to predict the MGMT promoter methylation status status on brain glioma [16]. They found that the best-performing model (multilayer perceptron) using 5 most relevant imaging features showed good performance, with an AUC of 0.84. A similar approach was followed by Hashido *et al.* [25] to predict glioma grade (i.e., high vs low-grade glioma), who found that a radiomics model combined with rCBV from T2*-DSC achieved excellent performance (AUC = 0.962) to differentiate between low and high-grade glioma. Interestingly, they also found that a similar model based on arterial spin labeling perfusion MRI instead of T2*-DSC MRI showed non-significantly lower results. In addition, Sudre *et al.* [51] applied a random forest model to combined radiomics-T2*-DSC perfusion MRI metrics, achieving good results to predict the IDH mutation status and glioma grade (WHO II-IV), with overall specificity of 77% and sensitivity of 65%, but worse results were obtained for WHO glioma grade classification (53% of gliomas were correctly classified).

Similarly, Park *et al.* developed an autoencoder to analyze perfusion heterogeneity in brain tumors, achieving excellent performance in differentiating glioblastoma from primary central nervous lymphoma and metastases [40]. In a different clinical challenge, namely differentiating between brain glioma progression and pseudoprogression, authors like Elshafeey *et al.* and Kim *et al.* applied radiomics to MRI perfusion parameter maps, achieving excellent outcomes [20, 31]. Nevertheless, a comprehensive review of the literature has not revealed any studies that leverage perfusion curves as a data source for this purpose.

Moreover, recent studies have focused on the analysis of specific glioma subtypes, investigating their associations with other genetic and molecular markers. Liu *et al.* [34] developed a hybrid approach, combining nnU-Net and K-means clustering, to predict IDH mutation status in HGGs. Heo *et al.* [26] proposed a model to forecast local progression in adult grade IV gliomas. Furthermore, Bachi *et al.* [3] introduced a convolutional neural network-based framework for the detection of HGGs using multi-sequence MRI. Nevertheless, to the best of our knowledge, no existing methodology has addressed the comprehensive differentiation of gliomas, encompassing both LGGS and HGGS as well as the entire spectrum of WHO grades, to the extent that our proposed method does.

2.2 Trustworthy AI: Conformal Prediction and cancer detection

Conformal Prediction, as outlined in [48], provides a robust framework for quantifying the uncertainty associated with predictions made by machine learning models. Given a training dataset comprising feature vectors and their corresponding labels, and a new, unlabeled instance, CP determines the plausibility of assigning each possible label to the new instance by evaluating whether it could have been sampled from the same underlying probability distribution as the training data. This is achieved by constructing a conformity score for each label and comparing it to a nonconformity measure computed from the training data. A prediction is considered reliable if the conformity score falls below a pre-specified threshold, indicating that the predicted label is consistent with the observed patterns in the training data [44]. Within the family of conformal predictors, Venn-ABERS stands out as a particularly effective method for binary classification problems [54]. It generates probabilistic predictions for unseen instances, guaranteeing that these probabilities are well-calibrated under the assumption of independent and identically distributed data. In particular, the Inductive Venn-ABERS (IVAP) variant necessitates a distinct calibration set and thereby ensures that the conformalization process is agnostic of the underlying base model [43].

A comprehensive review of the literature reveals no existing state-of-the-art method that leverages conformal prediction for quantifying uncertainty in glioma differentiation. Nevertheless, CP has been successfully applied in various clinical contexts [53, 23], particularly in the prediction of other tumor types, within the last few years. In [37], Millar *et al.* utilize CP to quantify uncertainty in breast cancer risk prediction. Similarly, [1] introduces a method for breast cancer prediction using histopathological images, combining deep learning and CP. Specifically, an optimized version of the DenseNet-121 model is employed through transfer learning, with conformal prediction incorporated to quantify the uncertainty of the model's outputs. Beyond breast cancer prediction, this technique is also applied to enhance treatment outcomes. In particular, Garcia-Galindo *et al.* propose a methodology in [24] to quantify the effectiveness of Neoadjuvant Therapy (NAT), considered the most effective preoperative treatment for reducing tumor charge in breast cancer. However, CP applications extend beyond breast cancer. In [21], Gade *et al.* present a hybrid method combining deep learning and CP to segment prostates in MRIs, thereby facilitating cancer diagnosis. For more general segmentation applications, Brunekreef *et al.* propose an efficient method in [10] for calibrating image segmentation algorithms. Returning to cancer applications, Kapuria *et al.* propose a framework in [30] that fosters collaboration between clinicians and AI for colorectal cancer diagnosis. As evident, CP is a technique gaining traction due to its robustness and versatility in quantifying uncertainty. Nevertheless, as noted in [55], it has not yet become the predominant technique in the scientific literature.

3 Tumor differentiation problem

Tumor differentiation is a problem closely related to tumor diagnosis, with significant implications in treatment and prognosis. Once a tumor has been detected, identifying its exact nature is necessary because the specific treatment depends on it. For instance, tumors with a higher tendency to grow fast exhibit distinct molecular or histological features and can benefit from more aggressive therapies. At the same time they usually carry a worse prognosis. Therefore, it is necessary to clearly identify the specific type of tumor, and this is common to any malignancy in the human body, although the specific criteria used for their classification may vary depending on factors such as the anatomical location.

Gliomas represent the most frequent primary malignant tumors of the brain, which has motivated extensive research to improve their characterization. Particular emphasis has been put into finding molecular, histological, and even radiological variables that can be useful for prognostic purposes. Traditionally, the most relevant factors considered in glioma classification, established by the World Health Organization, were based on histological characteristics. However, a fundamental paradigm shift occurred in the 2016 WHO classification, which preconized molecular biomarkers in the classification of brain gliomas. This paradigm shift was consolidated and further extended in the most recent WHO classification (2021). Accordingly, molecular biomarkers such as the Isocitrate Dehydrogenase (IDH) gene status or 1p19q codeletion are now systematically assessed to classify brain gliomas.

Despite these relatively recent changes in the criteria used in the WHO classification, gliomas are still divided into four grades (I-IV), with a worse prognosis as the grade increases. Thus, WHO I gliomas carry an excellent prognosis, whereas WHO IV gliomas (e.g., glioblastomas) are associated with very poor survival outcomes. For management purposes, these four grades are commonly regrouped into a binary classification, namely LGG (WHO I and WHO II) and HGG (WHO III and WHO IV). Each of these two groups share similar prognosis and, although not so fine-tuned as the corresponding WHO grades, they are useful to decide appropriate patient management.

As one could expect, considering the need for assessing tumor biomarkers in order to establish the WHO grade of a glioma, invasive methods to obtain a histological specimen of the tumor are necessary. This is usually performed through stereotactic biopsies, which entail patient risks. Therefore, the search for reliable non-invasive techniques to characterize brain gliomas has not ceased. In this setting, MRI has become the focus of a number of studies aiming to correlate specific imaging findings with glioma grading. As a first approach to this problem, most studies have focused on differentiating LGG from HGG, but increasing interest on predicting the exact WHO grade is becoming apparent in recent studies.

In this setting, Artificial Intelligence and, particularly, Machine Learning approaches, are promising, since they can capture a lot of information that escapes human interpreters. On top of that, continuous improvement in MRI techniques, allowing to obtain larger amount of data with increased spatial and temporal resolution, has created an excellent environment for the development of robust non-invasive characterization methods. Notably, perfusion MRI offers an excellent opportunity for the development of machine learning methods, because not only does it provide spatial information but also —and most importantly— temporal information. The latter is of particular importance in the context of brain gliomas due to certain physiological particularities in the behavior of tumoral tissue perfusion within the brain (e.g., disruption of the blood-brain barrier). In clinical practice, the first step is obtaining an MRI from the patient's brain. Although a variety of MRI protocols exist for suspected or confirmed brain tumor, most include at least T1, T2, T2 FLAIR, DWI/ADC and perfusion sequences should be present for appropriate tumor characterization. As mentioned above, the most relevant particularity of perfusion MRI lies in obtaining imaging changes over time.

Therefore, when approaching the characterization of a brain tumor, different features from MRI perfusion sequences are analyzed. One frequently considered is the set of perfusion curves. There is one such curve for each frame. These curves are composed by calculating a value on the frame for each timepoint of the MRI sequence, so that the curve is composed of the ordered sequence of values. For example, if the MRI includes images captured at 40 successive time steps, the corresponding perfusion curve is composed of 40 points. The actual shape of this curve encodes information that can be relevant for an effective diagnosis. A number of MRI perfusion —also termed 'hemodynamic' parameters have been described in the literature. Some of the most consistently used in clinical practice and research include cerebral blood volume, cerebral blood flow, mean transit time and time to peak, and they can be obtained from the MRI perfusion curve. For instance, cerebral blood volume can be obtained by integrating the time-intensity curve of the MRI perfusion sequence. Despite these MRI parameters have been proved useful for tumor differentiation [14], a lot of information contained in the perfusion curve could be further exploited using, for instance, machine learning algorithms.

4 Our proposal

In this paper, we present a comprehensive pipeline for glioma differentiation. We address the problem of glioma differentiation at two level of precision. The first one is concerned with differentiation between low-grade gliomas (LGG) and high-grade gliomas (HGG), which is an important heuristic dichotomization with relevant clinical implications in patient management. The second one goes on step ahead by trying to categorize gliomas according to WHO grades, a widely used standard in the global medical community that aids physicians in formulating an effective diagnosis and treatment for the disease.

To this end, we start from a certain sequence of the patient's 3D MRI. After segmentation, specific information in the shape of a time series is extracted. This information is referred to as the perfusion curve of the MRI, which will be the fundamental information for the machine learning models. Given the particular relationship between the different types of gliomas and WHO grades, we approach the learning task as hierarchical classification, combining highly complex deep learning models with state-of-the-art methods for time series classification. Finally, we utilize conformal prediction to quantify the uncertainty in the models' outputs. A graphical representation of the overall procedure can be found in Figure fig:pipeline. A detailed explanation of each phase is provided below.

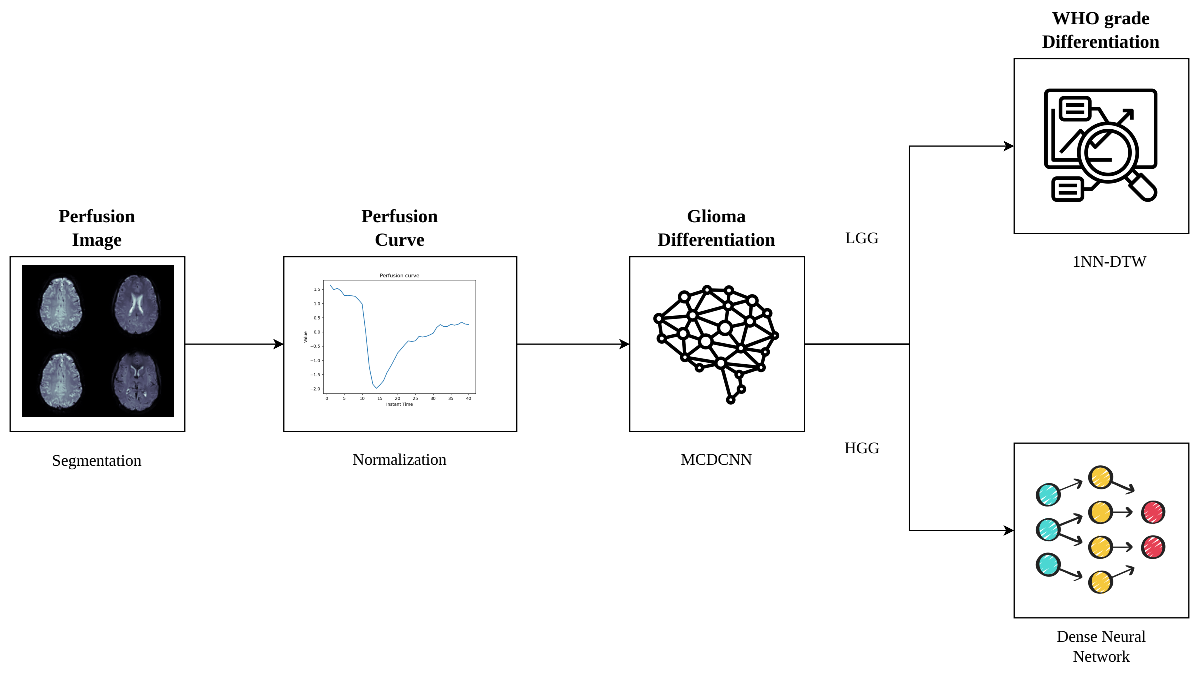


Figure 1: Complete pipeline of our glioma differentiation system. The perfusion sequence is normalized, and the perfusion curve is extracted from it, which will be the input to our predictor. Given that tumor differentiation is divided into two phases (first, distinction between high and low grade; second, differentiation between the four WHO grades), we construct a hierarchical classifier. The time series enters the first classifier, whose model is MCDCNN, to differentiate between LGG and HGG. If the case is categorized as LGG, the series is transferred to a 1NN-DTW model, which is used to differentiate between WHO grades I and II. Conversely, if it is classified as HGG, a dense neural network is used to differentiate between WHO grades III and IV.

4.1 Sequence Selection and Image Segmentation

To address this classification problem, we focused on the information contained in T2*-dynamic susceptibility contrast (DSC) MR perfusion images. Briefly, this sequence exploits the changes in the T2* relaxation times of protons following the intravenous administration of a gadolinium-based contrast agent to generate dynamic (i.e., time-dependent) images of an anatomical region. As opposed to other MRI perfusion modalities, such as T1 dynamic contrast enhancement, the changes in T2*-DSC MRI signal intensity lead to "negative" (i.e., darker than background) image contrast in the regions through which the contrast-agent passes. Consequently, the graphical representation of signal intensity for a given pixel across different timepoints generates a curve with a negative slope (Figure fig:pipeline). Further information regarding T2*-DSC perfusion MRI can be consulted in [6, 7]. The examinations were performed on a Philips Ingenia CX 3T system using a 2D gradient-echo/echo-planar sequence. The specific imaging parameters for DSC-MRI were as follows: TR/TE, 2059/40 ms; section thickness, 4 mm; matrix, 128×128; number of excitations, 1; and flip angle, 75°. Total acquisition time: 88.5 s. Total temporal sequences: 40.

Following the selection of the appropriate sequence, the image segmentation phase commences. While various initiatives for automatic tumor segmentation have been proposed [38, 22, 57, 28], to the best of our knowledge, no effective automated system exists for perfusion sequence segmentation. Therefore, to ensure optimal segmentation quality, the radiologists team performed the segmentation manually. In particular, using 3D Slicer® (v. 5.8.0), a neuroradiologist with more than 10 years of experience segmented the glioma and a region of normal-appearing white matter (NAWM) on the T2*-DSC MRI sequence after examining contrast-enhanced T1-weighted images for appropriate lesion location. Despite the high cost associated with this approach, it was feasible given the moderate size of the dataset.

4.2 Perfusion curves: from images to time series

As previously mentioned, the perfusion sequence contains a number of snapshots that allow the study of contrast flow over time. This is expressed in terms of image luminance. Therefore, by calculating the mean luminance value of voxels at each time instant, a time series, called perfusion curve, is generated. It encapsulates the information of the perfusion sequence and is consequently very useful for tumor differentiation. Finally, perfusion curves are normalized to a common scale before being fed into machine learning models. As a result, the glioma differentiation problem, originally conceived as an image classification problem, can be casted as a time series classification task, and, as a result, it can be addressed with state-of-art techniques known in Artificial Intelligence.

4.3 Addressing a hierarchical classification challenge: Distinguishing between HGG and LGG and assigning WHO grades

The normalized perfusion sequence serves as input to machine learning models to address two distinct classification problems. The first aims to differentiate gliomas into low-grade gliomas (LGG) and high-grade gliomas (HGG). The second objective is to assign a WHO grade (I-IV) to each case. These problems, however, are not independent; rather, they exhibit a close relationship: a low-grade glioma can be categorized as WHO grade I or II, while a high-grade glioma can be classified as WHO grade III or IV. Consequently, despite being binary and multiclass classification problems, respectively, we establish a hierarchical classification system. In the first stage, utilizing a MCDCNN model, we distinguish between LGG and HGG. Subsequently, to determine the WHO grade, we employ a specialized 1NN-DTW model to differentiate between grades I and II for cases previously labeled as LGG. Conversely, if the preceding model classifies the case as HGG, we utilize a dense neural network to discriminate between WHO grades III and IV.

4.4 Uncertainty Quantification: Conformal Prediction

A point of the utmost relevance is assessing the confidence on the output of classifiers. Many classifiers can provide a probability, that, however, is not usually well calibrated. An innovative approach can be considered instead, conformal predictors. Specifically, since the aforementioned scheme involves three distinct binary classifiers, we employ the Inductive Venn-Abers conformal predictor [54].

This approach ensures that the models, namely MCDCNN, 1NN-DTW, and a dense deep neural network, generate calibrated probabilities. This tool is crucial as it enhances model predictive accuracy while simultaneously enabling clinicians to assess the confidence of each model at various predictive stages. In conclusion, the proposed pipeline for glioma differentiation is both accurate, trustworthy and reliable, combining state-of-the-art models with robust uncertainty quantification tools.

5 Empirical analysis

A thorough experimental study has been designed to assess the efficacy of our novel methodology for differentiating gliomas into low-grade, high-grade, and specific WHO grade categories (I-IV). This section outlines the experimental goals, the dataset, evaluation metrics, and the employed classifiers.

5.1 Experimental goals

This experimentation has broad and ambitious goals. Primarily, we seek to create a robust and effective pipeline for solving two time series classification problems: differentiating between LGGs and HGGs, and then further classifying tumors according to WHO grades. Recognizing the real-world complexities of this biomedical problem, as discussed in the Dataset section, this is a significant challenge. Our approach not only involves employing highly predictive models like convolutional neural networks but also aims to provide clinicians with tools to quantify the uncertainty associated with the system's predictions, thereby facilitating more informed decision-making. To achieve this, we incorporate Conformal Predictor and specific metrics that evaluate both predictive performance and the quality of conformalization. Subsequent sections provide in-depth details about these metrics and models. In essence, this experimentation strives to find the best configuration to solve this problem in the most efficient, effective and informative manner, with the goal of developing a computer-aided diagnosis (CAD) system for radiologists.

5.2 Dataset

A cohort of 106 patients from the "Hospital Universitario Virgen de las Nieves" (Granada, Spain), selected for high-quality, representative, and homogeneous MRIs, was assembled. As explained in the previous section, a two-level hierarchical classification problem has been addressed: LGG vs. HGG, followed by WHO grade differentiation within each group. Figures fig:lgg-hgg and fig:who-degree illustrate the class distribution for both classification tasks.

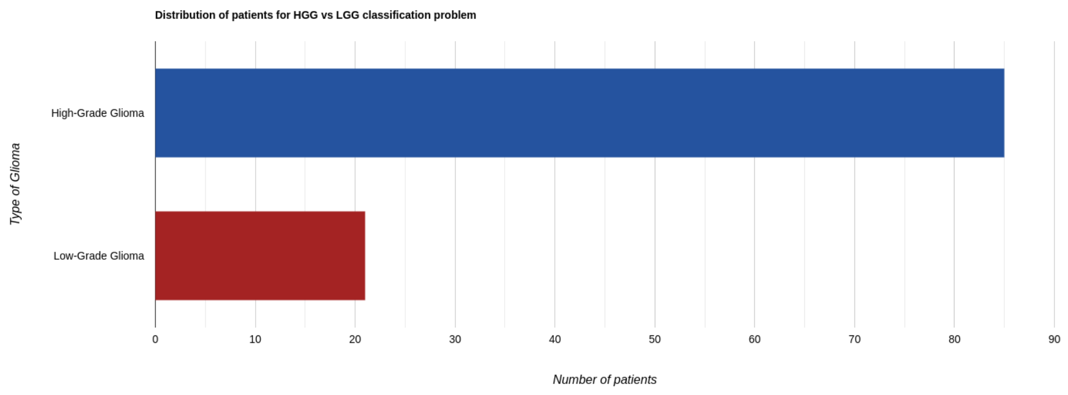


Figure 2: Distribution of patients for HGG vs. LGG classification problem

As it can be noticed in the figures, the dataset exhibits a significant class imbalance, mirroring real-world disease prevalence [39]. Specifically, 70% of our cohort were diagnosed with HGG, while 30% had LGG. Regarding WHO grades, 62% were grade IV, followed by 18% grade III, 15% grade II, and 5% grade I. This substantial class imbalance poses a significant challenge in building accurate classifiers for the less represented classes.

Given that our approach involves uncertainty quantification through conformal predictors, the dataset is divided into training, calibration, and test sets. For the construction of the training set, we implemented a data augmentation strategy. Specifically, for each patient, we generated all possible perfusion curves by traversing the z-axis along the segmented MRI.

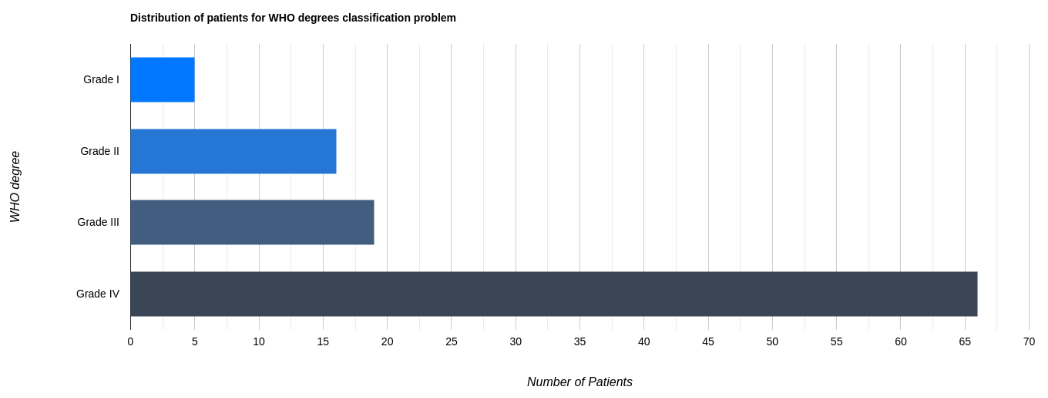


Figure 3: Distribution of patients for WHO grades classification problem

5.3 Metrics

Our methodology employs a hybrid approach combining deep learning, traditional time series classification models, and conformal prediction. Consequently, we combine accuracy metrics such as accuracy and F1-score with calibration metrics like Brier Score (BS) [9] and Log Loss (LL). On the one hand, Brier Loss is defined as

$$BS = \frac{1}{n}\sum_{i=1}^{n}(p_i - o_i)^2, \tag{1}$$

where p is the prediction probability of occurrence of the event i. Besides, o_i is defined as:

$$o_i = \begin{cases} 1, & \text{if } i \text{ happened} \\ 0, & \text{if } i \text{ not happened} \end{cases} \tag{2}$$

On the other hand, Log Loss is defined as

$$LL(y,p) = -(y\log(p) + (1-y)\log(1-p)) \tag{3}$$

with $y \in \{0,1\}$ and $p = P(y=1)$.

5.4 Classifiers

To address the time series classification problems that we are facing effectively, a variety of state-of-the-art techniques are employed. Specifically, Random Forest [17], 1NN-DTW [45], MCDCNN (a multi-channel convolutional model specially designed for multivariate time series) [59], Shapelet (an effective and interpretable approach for time series classification by focusing on distinctive subsequences within the data called shapelets) [27], SVC (an adapted version of support vector classifier for time series) [52], and dense neural networks are included in the experimental setup. Additionally, both conformalized and non-conformalized versions of these models are investigated, utilizing the Conformal Predictor Inductive Venn-ABERS and the metrics outlined in the preceding section. In the case of MCDCNN and DNN, the original models were not incorporated into the pipeline. Instead, simplified and optimized versions were used, resulting from the application of techniques OCNNA [4] and ODF2NNA [5] respectively.

Our analysis, presented in the results and discussion section, indicates that the most effective pipeline configuration utilizes a simplified MCDCNN model to distinguish between HGG and LGG tumors. Following this initial classification, we employ 1NN-DTW for further differentiating between WHO grades I and II (LGG), and a DNN for grades III and IV (HGG). The final, optimized pipeline structure is depicted in Figure fig:pipeline.

6 Results analysis and discussion

The results obtained for each of the candidate classifiers considered are shown in Tables HGGvsLGG, 1vs2 and 3vs4. Each cell of these tables contains two numbers: the first one refers to the unconformalized version of the model and the second one to the conformalized one and all of them are obtained averaging the results from a stratified 5-fold cross-validation process.

Table 1: Results for the differentiation between HGG and LGG. The metrics, established by columns, are Accuracy (Acc), F1-Score (F1), Log Loss (LL), and Brier Loss (BR). Additionally, the models are compared in their unconformalized version and conformalized versions (first value for unconformalized, second for conformalized). We consider the conformalized models. As a 5-fold stratified cross-validation was performed, the values shown are the average of the metrics obtained in the test set for each fold. MCDCNN* and DNN* models indicates that they have been simplified using the OCNNA [4] and ODF2NNA [5] techniques, respectively. As can be seen, the conformalized MCDCNN achieves better performance in terms of accuracy and conformalization metrics compared to the other approaches.

Model	**Acc**	**F1**	**BL**	**LL**
RF	0.762 / 0.760	0.863 / 0.86	0.557 / 0.556	0.176 / 0.182
1NN-DTW	0.737 / 0.742	0.837 / 0.847	9.486 / 0.546	0.263 / 0.181
MCDCNN*	0.698 / **0.778**	0.803 / **0.876**	3.265 / **0.528**	0.288 / **0.173**
Shapelet	0.703 / 0.777	0.815 / 0.874	5.0373 / 0.531	0.235 / 0.173
SVC	0.779 / 0.777	0.875 / 0.875	0.522 / 0.531	0.170 / 0.174
DNN*	0.661 / 0.772	0.766 / 0.870	1.585 / 0.564	0.300 / 0.182

The results presented in Table HGGvsLGG support the decision of selecting MCDCNN to perform the first stage of the classification process by distinguishing between HGG and LGG. This classification model shows the best performance according to the four metrics considered in this paper. It is also worth noting that all the techniques considered, except Random Forest, benefit from the use of conformalization. Moreover, without the use of conformalization the choice of the best performing model for this stage would have been different.

Regarding LLG classification into grade 1 and 2, Table 1vs2 shows 1NN-DTW is the best choice. Conformalization boosts the performance of this model making it achieve better results than the rest by a larger margin than in the previous stage.

Finally, HGG classification into grade 3 and 4 is done using a DNN model because it achieves the best results in terms of F1-score, Brier Loss and Log Loss and performs practically almost as the winning techniques in terms of accuracy (Shapelet and SVC) as shown in Table 3vs4.

Figure test$_probillustrateswithdetailtheeffectofconformalizationonthefirstclassificationstage(atonerandoml$
$validationiteration), wheresamplesareclassifiedintoHGGorLGG.Thesamplesforwhichthenon-$
$calibratedprediction(turquoisepoint)doesnotmatchthetruelabel(blackpoint)wouldbelongtothemisclassifiedsam$
$6, 7, 8and9arecorrectlyclassifiedandthisisnotthecasewhenthead-hocmodelisused.Samples\#1and\#2, evenusi$

Table 2: Results for the differentiation between WHO grades 1 and 2. The metrics, established by columns, are Accuracy (Acc), F1-Score (F1), Log Loss (LL), and Brier Loss (BR). Additionally, the models are compared in their unconformalized version and conformalized versions (first value for unconformalized, second for conformalized). We consider the conformalized models. As a 5-fold stratified cross-validation was performed, the values shown are the average of the metrics obtained in the test set for each fold. MCDCNN* and DNN* models indicates that they have been simplified using the OCNNA ([4]) and ODF2NNA ([5]) techniques, respectively. As can be seen, the conformalized 1NN-DTW achieves better performance in terms of accuracy and conformalization metrics compared to the other approaches.

Model	**Acc**	**F1**	**BL**	**LL**
RF	0.584 / 0.555	0.547 / 0.704	1.289 / 0.744	0.319 / 0.262
1NN-DTW	0.453 / **0.748**	0.472 / **0.853**	19.715 / **0.651**	0.547 / **0.215**
MCDCNN*	0.454 / 0.627	0.603 / 0.746	2.687 / 0.723	0.489 / 0.249
Shapelet	0.587 / 0.668	0.561 / 0.780	8.437 / 0.676	0.352 / 0.230
SVC	0.748 / 0.684	0.644 / 0.808	0.654 / 0.748	0.225 / 0.245
DNN*	0.574 / 0.635	0.724 / 0.763	6.265 / 0.694	0.420 / 0.239

Table 3: Results for the differentiation between WHO grades 3 and 4. The metrics, established by columns, are Accuracy (Acc), F1-Score (F1), Log Loss (LL), and Brier Loss (BR). Additionally, the models are compared in their unconformalized version and conformalized versions (first value for unconformalized, second for conformalized). We consider the conformalized models. As a 5-fold stratified cross-validation was performed, the values shown are the average of the metrics obtained in the test set for each fold. MCDCNN* and DNN* models indicates that they have been simplified using the OCNNA ([4]) and ODF2NNA ([5]) techniques, respectively. As can be seen, the conformalized dense neural network achieves better performance in terms of accuracy and conformalization metrics compared to the other approaches.

Model	**Acc**	**F1**	**BL**	**LL**
RF	0.699 / 0.755	0.640 / 0.860	0.952 / 0.559	0.243 / 0.185
1NN-DTW	0.619 / 0.752	0.609 / 0.858	13.720 / 0.552	0.381 / 0.183
MCDCNN*	0.585 / 0.760	0.725 / 0.863	2.003 / 0.593	0.357 / 0.195
Shapelet	0.662 / **0.767**	0.634 / 0.868	4.810 / 0.562	0.268 / 0.187
SVC	0.774 / 0.767	0.676 / 0.867	0.597 / 0.607	0.191 / 0.203
DNN*	0.735 / 0.766	0.846 / **0.869**	3.481 / **0.550**	0.257 / **0.182**

7 Conclusions

We have addressed the brain glioma differentiation problem by using brain T2*-DSC MRI perfusion sequences of patients as inputs. The goal is to precisely characterize the glioma subtype at two precision levels, HGG vs. LGG and WHO I-IV. The innovations of our approach are focusing on the perfusion curves, each of which is actually a time series, and formulate the differentiation problem as a time series classification problem.

Taking the perfusion curve as input, our proposal prescribes the construction of a classifier to predict the diagnosis. Different state-of-the-art classifiers have been considered. After a thorough empirical evaluation, the highest performance has been obtained with a Deep Neural Network. In addition, conformal predictors have been developed to bound the uncertainty on the prediction. The use of conformal predictions showed a significant improvement in the predictive capacity of the applied deep learning models, as well as a high degree of certainty.

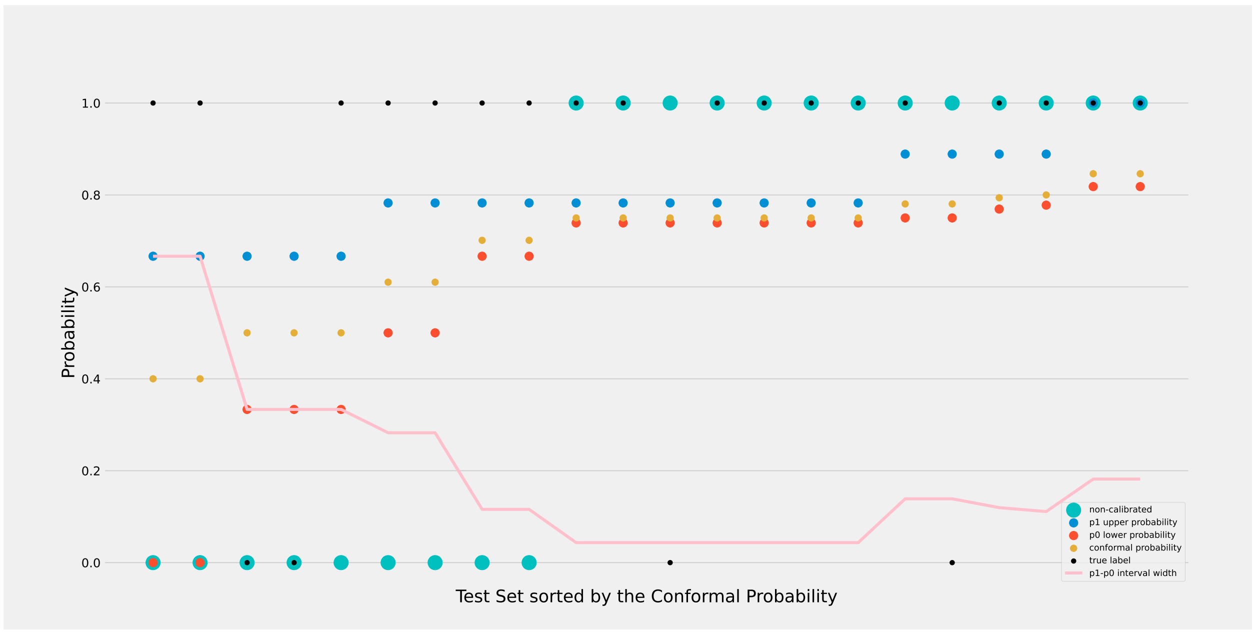


Figure 4: Analysis of the predictions for HGG and LGG differentiation. The x-axis represents the test cases. The y-axis shows the probability of each prediction. For each case, black points indicate the true label, turquoise points (thicker) represent the prediction of the model in its unconformalized version. Finally, red, light blue, and yellow points represent, respectively, the lower probability p_0, the upper probability p_1, and the conformalized probability. The distance between p_0 and p_1, illustrated by the pink line, indicates the model's uncertainty about each prediction: the smaller the distance, the more confident the predictions; the larger the distance, the greater the uncertainty. Thanks to conformalization, the sixth, seventh, eighth, and ninth cases from the left are now correctly classified (compared to the unconformalized model). Additionally, the great utility of this tool is observed for the first two cases from the left, since despite the incorrect prediction, a high prediction uncertainty is shown.

Ethics Statement

The study was conducted in accordance with the Declaration of Helsinki and approved by our Institutional Review Board. (approval code: IANeuro24; approval date: 29 June 2021). Due to the retrospective nature of the study, written informed consent was waived.

Acknowledgment

This research has been partially supported by Proyecto PID2023-151336OB-I00 financiado por MICIU/AEI /10.13039/501100011033 y por FEDER, UE.

References

References

[1] Bandar Almaslukh. A reliable breast cancer diagnosis approach using an optimized deep learning and conformal prediction. *Biomedical Signal Processing and Control*, 98, DEC 2024.

[2] Samy Ammari, Alexandre Bône, Corinne Balleyguier, Eric Moulton, Émilie Chouzenoux, Andreas Volk, Yves Menu, François Bidault, François Nicolas, Philippe Robert, Marc-Michel Rohé, and Nathalie Lassau. Can deep learning replace gadolinium in neuro-oncology? A reader study. *Investigative Radiology*, 57(2), 2022.

[3] Stephen Bacchi, Toby Zerner, John Dongas, Adon Toru Asahina, Amal Abou-Hamden, Sophia Otto, Luke Oakden-Rayner, and Sandy Patel. Deep learning in the detection of high-grade glioma recurrence using multiple mri sequences: A pilot study. *Journal of Clinical Neuroscience*, 70:11–13, DEC 2019.

[4] Luis Balderas, Miguel Lastra, and José M. Benítez. Optimizing convolutional neural network architectures. *Mathematics*, 12(19), 2024.

[5] Luis Balderas, Miguel Lastra, and José M. Benítez. Optimizing dense feed-forward neural networks. *Neural Networks*, 171:229–241, 2024.

[6] Roland Bammer and Shalini A Amukotuwa. Dynamic susceptibility contrast perfusion, Part 1: The fundamentals. *Magn Reson Imaging Clin N Am*, 32(1):1–23, Feb 2024.

[7] Roland Bammer and Shalini A Amukotuwa. Dynamic susceptibility contrast perfusion, Part 2: Deployment with and without contrast leakage present. *Magn Reson Imaging Clin N Am*, 32(1):25–45, Feb 2024.

[8] Jayendra M Bhalodiya, Sarah N Lim Choi Keung, and Theodoros N Arvanitis. Magnetic resonance image-based brain tumour segmentation methods: A systematic review. *Digital Health*, 8:20552076221074122, 2022. PMID: 35340900.

[9] Glenn W. Brier. Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, 78(1):1 – 3, 1950.

[10] Joren Brunekreef, Eric Marcus, Ray Sheombarsing, Jan-Jakob Sonke, and Jonas Teuwen. Kandinsky conformal prediction: Efficient calibration of image segmentation algorithms. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024*, IEEE Conference on Computer Vision and Pattern Recognition, pages 4135–4143. IEEE; IEEE Comp Soc; CVF, 2024. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, JUN 16-22, 2024.

[11] Evan Calabrese, Jeffrey D Rudie, Andreas M Rauschecker, Javier E Villanueva-Meyer, Jennifer L Clarke, David A Solomon, and Soonmee Cha. Combining radiomics and deep convolutional neural network features from preoperative mri for predicting clinically relevant genetic biomarkers in glioblastoma. *Neuro-Oncology Advances*, 4(1):vdac060, 04 2022.

[12] P Chang, J Grinband, B D Weinberg, M Bardis, M Khy, G Cadena, M-Y Su, S Cha, C G Filippi, D Bota, P Baldi, L M Poisson, R Jain, and D Chow. Deep-learning convolutional neural networks accurately classify genetic mutations in gliomas. *AJNR Am. J. Neuroradiol.*, 39(7):1201–1207, July 2018.

[13] Hongyu Chen, Fuhua Lin, Jinming Zhang, Xiaofei Lv, Jian Zhou, Zhi-Cheng Li, and Yinsheng Chen. Deep learning radiomics to predict PTEN mutation status from magnetic resonance imaging in patients with glioma. *Front. Oncol.*, 11:734433, October 2021.

[14] S.K. Cho, D.G. Na, J.W. Ryoo, H.G. Roh, C.H. Moon, H.S. Byun, and J.H. Kim. Perfusion MR imaging: clinical utility for the differential diagnosis of various brain tumors. *Korean J. Radiol.*, 3(3):171–179, 2002.

[15] Kyu Sung Choi. Deep learning applications in perfusion MRI: recent advances and current challenges. *Investigative Magnetic Resonance Imaging*, 26(4):246–255, 2022.

[16] Girolamo Crisi and Silvano Filice. Predicting mgmt promoter methylation of glioblastoma from dynamic susceptibility contrast perfusion: A radiomic approach. *Journal of Neuroimaging*, 30(4):458–462, May 2020.

[17] Houtao Deng, George Runger, Eugene Tuv, and Martyanov Vladimir. A time series forest for classification and feature extraction. *Information Sciences*, 239:142–153, 2013.

[18] Paul Dequidt, Pascal Bourdon, Olfa Ben Ahmed, Benoit Tremblais, Carole Guillevin, Mathieu Naudin, Christine Fernandez-Maloigne, and Remy Guillevin. Recent advances in glioma grade classification using machine and deep learning on MR data. In *2019 Fifth International Conference on Advances in Biomedical Engineering (ICABME)*, International Conference on Advances in Biomedical Engineering, pages 20–23, 2019. 5th International Conference on Advances in Biomedical Engineering (ICABME), Tripoli, LEBANON, OCT 17-19, 2019.

[19] Gianfranco Di Salle, Lorenzo Tumminello, Maria Elena Laino, Sherif Shalaby, Gayane Aghakhanyan, Salvatore Claudio Fanni, Maria Febi, Jorge Eduardo Shortrede, Mario Miccoli, Lorenzo Faggioni, Mirco Cosottini, and Emanuele Neri. Accuracy of radiomics in predicting IDH mutation status in diffuse gliomas: A bivariate meta-analysis. *Radiology: Artificial Intelligence*, 6(1):e220257, 2024. PMID: 38231039.

[20] Nabil Elshafeey, Aikaterini Kotrotsou, Ahmed Hassan, Nancy Elshafei, Islam Hassan, Sara Ahmed, Srishti Abrol, Anand Agarwal, Kamel El Salek, Samuel Bergamaschi, Jay Acharya, Fanny E. Moron, Meng Law, Gregory N. Fuller, Jason T. Huse, Pascal O. Zinn, and Rivka R. Colen. Multicenter study demonstrates radiomic features derived from magnetic resonance perfusion images identify pseudoprogression in glioblastoma. *Nature Communications*, 10(1):3170, 2019.

[21] Marius Gade, Kevin Mekhaphan Nguyen, Sol Gedde, and Alvaro Fernandez-Quilez. Impact of uncertainty quantification through conformal prediction on volume assessment from deep learning-based MRI prostate segmentation. *Insights into Imaging*, 15(1), NOV 29 2024.

[22] Louis Gagnon, Diviya Gupta, George Mastorakos, Nathan White, Vanessa Goodwill, Carrie R. McDonald, Thomas Beaumont, Christopher Conlin, Tyler M. Seibert, Uyen Nguyen, Jona Hattangadi-Gluth, Santosh Kesari, Jessica D. Schulte, David Piccioni, Kathleen M. Schmainda, Nikdokht Farid, Anders M. Dale, and Jeffrey D. Rudie. Deep learning segmentation of infiltrative and enhancing cellular tumor at pre- and posttreatment multishell diffusion MRI of glioblastoma. *Radiology: Artificial Intelligence*, 6(5):e230489, 2024. PMID: 39166970.

[23] Cooper Gamble, Shahriar Faghani, and Bradley J. Erickson. Applying conformal prediction to a deep learning model for intracranial hemorrhage detection to improve trustworthiness. *Radiology: Artificial Intelligence*, 0(ja):e240032, 0. PMID: 39601654.

[24] Alberto Garcia-Galindo, Marcos Lopez-De-Castro, and Ruben Armananzes. An uncertainty-aware sequential approach for predicting response to neoadjuvant therapy in breast cancer. In H Papadopoulos, K AnNguyen, H Bostrom, and L Carlsson, editors, *Conformal and Probabilistic Prediction with Applications, Vol 204*, volume 204 of *Proceedings of Machine Learning Research*, pages 74–88. Frederick Univ; Albourne; Mitie; Cyta; Kroo, 2023. 12th Symposium on Conformal and Probabilistic Prediction with Applications (COPA), Limassol, CYPRUS, SEP 13-15, 2023.

[25] Takashi Hashido, Shigeyoshi Saito, and Takayuki Ishida. A radiomics-based comparative study on arterial spin labeling and dynamic susceptibility contrast perfusion-weighted imaging in gliomas. *Scientific Reports*, 10(1), April 2020.

[26] Donggeon Heo, Jisoo Lee, Roh-Eul Yoo, Seung Hong Choi, Tae Min Kim, Chul-Kee Park, Sung-Hye Park, Jae-Kyung Won, Joo Ho Lee, Soon Tae Lee, Kyu Sung Choi, Ji Ye Lee, Inpyeong Hwang, Koung Mi Kang, and Tae Jin Yun. Deep learning based on dynamic susceptibility contrast MR imaging for prediction of local progression in adult-type diffuse glioma (grade 4). *Scientific Reports*, 13(1), AUG 24 2023.

[27] Jon Hills, Jason Lines, Edgaras Baranauskas, James Mapp, and Anthony Bagnall. Classification of time series by shapelet transformation. *Data Mining and Knowledge Discovery*, 28(4):851–881, Jul 2014.

[28] Katharina V. Hoebel, Christopher P. Bridge, Sara Ahmed, Oluwatosin Akintola, Caroline Chung, Raymond Y. Huang, Jason M. Johnson, Albert Kim, K. Ina Ly, Ken Chang, Jay Patel, Marco Pinho, Tracy T. Batchelor, Bruce R. Rosen, Elizabeth R. Gerstner, and Jayashree Kalpathy-Cramer. Expert-centered evaluation of deep learning algorithms for brain tumor segmentation. *Radiology: Artificial Intelligence*, 6(1):e220231, 2024.

[29] Seyyed Ali Hosseini, Elahe Hosseini, Ghasem Hajianfar, Isaac Shiri, Stijn Servaes, Pedro Rosa-Neto, Laiz Godoy, MacLean P. Nasrallah, Donald M. O'Rourke, Suyash Mohan, and Sanjeev Chawla. MRI-based radiomics combined with deep learning for distinguishing IDH-mutant WHO grade 4 astrocytomas from idh-wild-type glioblastomas. *Cancers*, 15(3), 2023.

[30] Siddhartha Kapuria, Patrick Minot, Ariel Kapusta, Naruhiko Ikoma, and Farshid Alambeigi. A novel dual layer cascade reliability framework for an informed and intuitive clinician-AI interaction in diagnosis of colorectal cancer polyps. *IEEE Journal of Biomedical and Health Informatics*, 28(4):2326–2337, APR 2024.

[31] J.Y. Kim, J.E. Park, Y. Jo, W.H. Shim, S.J. Nam, J.H. Kim, R.E. Yoo, S.H. Choi, and H.S. Kim. Incorporating diffusion- and perfusion-weighted MRI into a radiomics model improves diagnostic performance for pseudoprogression in glioblastoma patients. *Neuro Oncol.*, 21(3):404–414, 2019.

[32] Junhyeok Lee, Woojin Jung, Seungwook Yang, Jung Hyun Park, Inpyeong Hwang, Jin Wook Chung, Seung Hong Choi, and Kyu Sung Choi. Deep learning-based super-resolution and denoising algorithm improves reliability of dynamic contrast-enhanced MRI in diffuse glioma. *Scientific Reports*, 14(1):25349, Oct 2024.

[33] Dongdong Lin, Ming Wang, Yan Chen, Jie Gong, Liang Chen, Xiaoyong Shi, Fujun Lan, Zhongliang Chen, Tao Xiong, Hu Sun, et al. Trends in intracranial glioma incidence and mortality in the united states, 1975-2018. *Frontiers in Oncology*, 11:748061, 2021.

[34] J. Liu, C. Cong, J. Zhang, J. Qiao, H. Guo, H. Wu, Z. Sang, H. Kang, J. Fang, and W. Zhang. Multimodel habitats constructed by perfusion and/or diffusion MRI predict isocitrate dehydrogenase mutation status and prognosis in high-grade gliomas. *Clinical Radiology*, 79(1):e127–e136, Jan 2024.

[35] Robert Luypaert, S Boujraf, Steven Sourbron, and Michel Osteaux. Diffusion and perfusion MRI: basic physics. *European journal of radiology*, 38(1):19–27, 2001.

[36] Georgios C Manikis, Georgios S Ioannidis, Loizos Siakallis, Katerina Nikiforaki, Michael Iv, Diana Vozlic, Katarina Surlan-Popovic, Max Wintermark, Sotirios Bisdas, and Kostas Marias. Multicenter DSC-MRI-based radiomics predict IDH mutation in gliomas. *Cancers (Basel)*, 13(16):3965, August 2021.

[37] Alexander S. Millar, John Arnn, Sam Himes, and Julio C. Facelli. Uncertainty in breast cancer risk prediction: A conformal prediction study of race stratification. In J Bichel-Findlay, P Otero, P Scott, and E Huesing, editors, *MEDINFO 2023 – The Future is Accesible*, volume 310 of *Studies in Health Technology and Informatics*, pages 991–995, 2024. 19th World Congress on Medical and Health Informatics (MEDINFO), Sydney, Australia, JUL 08-12, 2023.

[38] Shervin Minaee, Yuri Boykov, Fatih Porikli, Antonio Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3523–3542, 2022.

[39] L.B. Nabors, J. Portnow, M. Ahluwalia, J. Baehring, H. Brem, S. Brem, N. Butowski, J.L. Campian, S.W. Clark, A.J. Fabiano, P. Forsyth, J. Hattangadi-Gluth, M. Holdhoff, C. Horbinski, L. Junck, T. Kaley, P. Kumthekar, J.S. Loeffler, M.M. Mrugala, S. Nagpal, M. Pandey, I. Parney, K. Peters, V.K. Puduvalli, I. Robins, J. Rockhill, C. Rusthoven, N. Shonka, D.C. Shrieve, L.J. Swinnen, S. Weiss, P.Y. Wen, N.E. Willmarth, M.A. Bergman, and S.D. Darlow. Central nervous system cancers, version 3-2020, nccn clinical practice guidelines in oncology. *J Natl Compr Canc Netw.*, 18(11):1537–1570, 2022.

[40] Ji Eun Park, Ho Sung Kim, Junkyu Lee, E. Nae Cheong, Ilah Shin, Sung Soo Ahn, and Woo Hyun Shim. Deep-learned time-signal intensity pattern analysis using an autoencoder captures magnetic resonance perfusion heterogeneity for brain tumor differentiation. *Scientific Reports*, 10(1):21485, 2020.

[41] Y W Park, K Han, S S Ahn, S Bae, Y S Choi, J H Chang, S H Kim, S-G Kang, and S-K Lee. Prediction of*IDH1*-mutation and 1p/19q-codeletion status using preoperative MR imaging phenotypes in lower grade gliomas. *AJNR Am. J. Neuroradiol.*, 39(1):37–42, January 2018.

[42] Luca Pasquini, Antonio Napolitano, Emanuela Tagliente, Francesco Dellepiane, Martina Lucignani, Antonello Vidiri, Giulio Ranazzi, Antonella Stoppacciaro, Giulia Moltoni, Matteo Nicolai, Andrea Romano, Alberto Di Napoli, and Alessandro Bozzao. Deep learning can differentiate idh-mutant from idh-wild gbm. *Journal of Personalized Medicine*, 11(4), APR 2021.

[43] Telma Pereira, Sandra Cardoso, Manuela Guerreiro, Alexandre Mendonça, and Sara C. Madeira. Targeting the uncertainty of predictions at patient-level using an ensemble of classifiers coupled with calibration methods, Venn-ABERS, and conformal predictors: A case study in AD. *Journal of Biomedical Informatics*, 101:103350, 2020.

[44] Telma Pereira, Sandra Cardoso, Dina Silva, Alexandre de Mendonça, Manuela Guerreiro, and Sara C. Madeira. Towards trustworthy predictions of conversion from mild cognitive impairment to dementia: A conformal prediction approach. In Florentino Fdez-Riverola, Mohd Saberi Mohamad, Miguel Rocha, Juan F. De Paz, and Tiago Pinto, editors, *11th International Conference on Practical Applications of Computational Biology & Bioinformatics*, pages 155–163, Cham, 2017. Springer International Publishing.

[45] Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh. Addressing big data time series: Mining trillions of time series subsequences under dynamic time warping. *ACM Trans. Knowl. Discov. Data*, 7(3), September 2013.

[46] M. Riche, P. Marijon, A. Amelot, F. Bielle, K. Mokhtari, M.P. Chambrun, A.L. Joncour, A. Idbaih, M. Touat, C.H. Do, M. Deme, R. Pasqualotto, A. Jacquens, V. Degos, E. Shotar, L. Chougar, A. Carpentier, and B. Mathon. Severity, timeline, and management of complications after stereotactic brain biopsy. *J Neurosurg*, 136(3):867–876, 2021.

[47] Paniz Sabeghi, Paniz Zarand, Sina Zargham, Batis Golestany, Arya Shariat, Myles Chang, Evan Yang, Priya Rajagopalan, Daniel Chang Phung, and Ali Gholamrezanezhad. Advances in neuro-oncological imaging: An update on diagnostic approach to brain tumors. *Cancers*, 16(3), 2024.

[48] Glenn Shafer and Vladimir Vovk. A tutorial on conformal prediction. *J. Mach. Learn. Res.*, 9:371–421, June 2008.

[49] Gagandeep Singh, Annie Singh, Joseph Bae, Sunil Manjila, Vadim Spektor, Prateek Prasanna, and Angela Lignelli. New frontiers in domain-inspired radiomics and radiogenomics: increasing role of molecular diagnostics in CNS tumor classification and grading following WHO CNS-5 updates. *Cancer Imaging*, 24(1):133, Oct 2024.

[50] R Stupp, M Brada, MJ Van Den Bent, J-C Tonn, and GESMO Pentheroudakis. High-grade glioma: Esmo clinical practice guidelines for diagnosis, treatment and follow-up. *Annals of Oncology*, 25:93–101, 2014.

[51] Carole H. Sudre, Jasmina Panovska-Griffiths, Eser Sanverdi, Sebastian Brandner, Vasileios K. Katsaros, George Stranjalis, Francesca B. Pizzini, Claudio Ghimenton, Katarina Surlan-Popovic, Jernej Avsenik, Maria Vittoria Spampinato, Mario Nigro, Arindam R. Chatterjee, Arnaud Attye, Sylvie Grand, Alexandre Krainik, Nicoletta Anzalone, Gian Marco Conte, Valeria Romeo, Lorenzo Ugga, Andrea Elefante, Elisa Francesca Ciceri, Elia Guadagno, Eftychia Kapsalaki, Diana Roettger, Javier Gonzalez, Timothé Boutelier, M. Jorge Cardoso, and Sotirios Bisdas. Machine learning assisted dsc-mri radiomics as a tool for glioma classification by grade and mutation status. *BMC Medical Informatics and Decision Making*, 20(1), July 2020.

[52] Shan Suthaharan. *Support Vector Machine*, pages 207–235. Springer US, Boston, MA, 2016.

[53] Janette Vazquez and Julio C. Facelli. Conformal prediction in clinical medical sciences. *Journal of Healthcare Informatics Research*, 6(3):241–252, Sep 2022.

[54] Vladimir Vovk and Ivan Petej. Venn-Abers predictors, 2014.

[55] Kareem A. Wahid, Zaphanlene Y. Kaffey, David P. Farris, Laia Humbert-Vidan, Amy C. Moreno, Mathis Rasmussen, Jintao Rend, Mohamed A. Naser, Tucker J. Netherton, Stine Korreman, Guha Balakrishnan, Clifton D. Fuller, David Fuentes, and Michael J. Dohopolski. Artificial intelligence uncertainty quantification in radiotherapy applications: A scoping review. *Radiotherapy and Oncology*, 201, DEC 2024.

[56] Michael Weller, Martin van den Bent, Matthias Preusser, Emilie Le Rhun, Jörg C Tonn, Giuseppe Minniti, Martin Bendszus, Carmen Balana, Olivier Chinot, Linda Dirven, et al. Eano guidelines on the diagnosis and treatment of diffuse gliomas of adulthood. *Nature Reviews Clinical Oncology*, 18(3):170–186, 2021.

[57] Shaocheng Wu, Hongyang Li, Daniel Quang, and Yuanfang Guan. Three-plane–assembled deep learning segmentation of gliomas. *Radiology: Artificial Intelligence*, 2(2):e190011, 2020. PMID: 32280947.

[58] Kai Zhao, Boyuan Li, Kai Zhang, Ruoyu Liu, Long Gao, Xujun Shu, Minghang Liu, Xuejun Yang, Shengbo Chen, and Guochen Sun. Automatic 1p/19q co-deletion identification of gliomas by MRI using deep learning U-net network. *Computers & Electrical Engineering*, 105, JAN 2023.

[59] Yi Zheng, Qi Liu, Enhong Chen, Yong Ge, and J. Leon Zhao. Time series classification using multi-channels deep convolutional neural networks. In Feifei Li, Guoliang Li, Seung-won Hwang, Bin Yao, and Zhenjie Zhang, editors, *Web-Age Information Management*, pages 298–310, Cham, 2014. Springer International Publishing.