DOCTORAL THESIS

Automatic identification of the protein fold type using representations from the amino acid sequence and deep learning techniques

University of Granada

Department of Signal Theory, Telematics and Communications

Doctoral Program in Information and Communication Technologies

Author:

Amelia Otilia Villegas Morcillo

Thesis supervisors:

Victoria Eugenia Sánchez Calle

Ángel Manuel Gómez García

Granada, September 2022

A mi familia: Amelia, Antonio y Juanan

Acknowledgments

There are many people I would like to thank for being part of this journey and for making it more enjoyable in one way or another. Whether your name is on this list or not, if we have crossed paths at any point, thank you!

To my Thesis advisors, Victoria and Ángel, for giving me the opportunity to do a PhD with you in an exciting field from which I have learned so much over these years. For helping me to grow professionally and always maintaining a healthy work environment, I am eternally grateful to you. To Juan Andrés (Juanito) and Paco, for helping me to start researching in this field, I hope we can collaborate again in the future!

To the rest of the SigMAT group: Antonio, José Luis, José Andrés, Iván, and my office colleagues and friends Juanma, Alejandro (already doctorates!), and Eros, for being supportive and providing helpful guidance during these years. Thank you for the daily coffee breaks, the Friday drinks and tapas, and the occasional ultra-mountain hiking routes with 'happy endings' for our bellies. I wish you all the best in your next endeavors!

To my colleagues at TUDelft: to Marcel, firstly, for allowing me to do a research visit in the group, and to Stavros, secondly, for actually wanting to collaborate with me. To people in the Pattern Recognition, Computer Vision, and Bioinformatics labs (yeah and Social BlaBla also), for the fun times in every Thursday 'borrel' and the rare board game nights. Special mention to my friends Ramin, Yeshwanth, Aysun, (and Stavros of course!) for making me look forward to returning to the Netherlands, my second home.

To my beloved Chirag, for always providing incredible advice and helping me to see life from a different perspective. I deeply appreciate all the discussions, and I feel immensely lucky for having you by my side. This Thesis is also yours.

To my InstaDeep colleagues: my folding team Tom, Louis and Benoit, and the rest of people in the London and Paris offices, for the good vibes and fun at work, making my jump from academia to industry a smooth one.

A mis padres Amelia y Antonio, por apoyarme siempre con mis estudios y abrazar cada una de las oportunidades que me han surgido. A mi hermano Juanan (a.k.a. nene), por ser una fuente de inspiración tanto personal como profesional, y por enseñarme las maravillas del mundo tech e incrementar mi curiosidad cada día.

A la familia que se elige, mis amigos. A los de toda la vida: mis "peligrosas", Marga, María, Irene y Elena, y mis "granaínos", Dani, Joseca y Ramón; y a mis "telecos": Marga (again), Mario, Borja, Vicente, Irene y Fran. Gracias por hacer posible que pasemos tiempo de desconexión y risas todos juntos aunque estemos dispersos por el mundo.

Abstract

Proteins are the building blocks of life as they are present in most of the biological processes of living organisms. The accurate determination of the protein three-dimensional structure is essential for many applications including drug development and protein design. However, the high cost of experimental methods has generated an increasing gap between the number of protein sequences and 3D structures available in public databases. Furthermore, although all the information needed to fold a protein is contained in its amino acid sequence, the computational determination of the protein structure is a challenging problem due to the complexity of the physicochemical interactions that define such structure. One step towards resolving this is the identification of the fold type the protein belongs to by comparing it to solved structures. However, this approach has recently been superseded by several deep learning methods that succeeded in producing highly accurate 3D structures from scratch. Despite this, it remains crucial to develop algorithms that identify sequential and structural similarities between proteins at a low computational cost. Since structures tend to be better conserved than sequences over the course of evolution, protein fold prediction is also a tool to find structurally related proteins that may not be similar in sequence. This could help to annotate rare proteins that are yet to be characterized.

The main objective of this Thesis is therefore to advance research on protein fold prediction methods by exploiting the information contained in the amino acid sequences using deep learning algorithms. The results are presented in this dissertation as a compendium of scientific papers that have been published during the doctoral period.

The proposed strategies explore different research directions with a common ground: the use of deep learning techniques to learn meaningful embedding representations of protein fold types. First, image representations of the protein have been evaluated for the fold recognition task, including estimated and enhanced contact maps, as well as native contact and categorical distance maps (from the 3D structure). Then, a convolutional-recurrent neural network architecture has been proposed for fold recognition, which successfully processes arbitrary-length protein sequences using amino acid residue-level features. Subsequently, more discriminative embedding spaces of protein fold classes have been learned by adjusting the training procedure of neural network models, in particular, the loss function and the use of prototype fold class vectors to guide the classification. Finally, the performance of several pre-trained protein language model embeddings has been analyzed for the fold recognition and fold classification tasks, which have shown promise and great potential for the field.

Resumen

Las proteínas son los componentes básicos de la vida ya que están presentes en la mayoría de procesos biológicos de los seres vivos. La determinación de la estructura tridimensional de la proteína es esencial para muchas aplicaciones incluyendo el desarrollo de fármacos y el diseño de proteínas. Sin embargo, el alto coste de los métodos experimentales ha generado una brecha entre el número de secuencias y estructuras 3D de proteínas disponibles en las bases de datos. Además, a pesar de que toda la información necesaria para plegar una proteína está contenida en su secuencia de aminoácidos, la determinación de la estructura por métodos computacionales es difícil debido a la complejidad de las interacciones físico-químicas que definen dicha estructura. Un paso hacia su resolución es la identificación del tipo de plegamiento (*fold*) mediante comparación con estructuras resueltas. Sin embargo, este enfoque ha sido superado recientemente por varios métodos basados en aprendizaje profundo, los cuales han logrado producir estructuras 3D muy precisas desde cero. A pesar de ello, sigue siendo crucial el desarrollo de algoritmos que identifiquen similitudes secuenciales y estructurales entre proteínas a un bajo coste computacional. Dado que las estructuras tienden a conservarse mejor que las secuencias a lo largo de la evolución, la predicción del tipo de plegamiento de la proteína es también una herramienta para encontrar proteínas relacionadas entre sí a nivel estructural sin necesidad de ser similares a nivel de secuencia. Esto podría ayudar en la anotación de proteínas poco comunes que están aún por caracterizar.

El objetivo principal de esta Tesis es, por tanto, avanzar en la investigación de los métodos de predicción del plegamiento de proteínas explotando la información contenida en las secuencias de aminoácidos mediante el uso de algoritmos de aprendizaje profundo. Los resultados se presentan en esta memoria como un compendio de artículos científicos que han sido publicados durante el periodo doctoral.

Las estrategias propuestas exploran diferentes direcciones de investigación con una base común: el uso de técnicas de aprendizaje profundo para aprender representaciones compactas (*embeddings*) significativas de los tipos de plegamiento de las proteínas. En primer lugar, se han evaluado representaciones en forma de imagen de la proteína para la tarea de reconocimiento del plegamiento, incluyendo los mapas de contactos estimados y mejorados, así como los mapas de contactos nativos y de distancias categorizadas (a partir de la estructura 3D). Seguidamente, se ha propuesto una arquitectura de red neuronal de tipo convolucional-recurrente para el reconocimiento del plegamiento, la cual procesa con éxito secuencias de proteínas de longitud arbitraria utilizando características a nivel de aminoácido. Posteriormente, se han aprendido espacios de *embedding* más discriminativos de los plegamientos mediante el ajuste del entrenamiento de las redes neuronales, en particular la función de pérdidas y el uso de vectores prototipo para cada clase con objeto de guiar la clasificación. Por último, se ha analizado el rendimiento de varios *embeddings* extraídos de modelos de lenguaje de proteínas para las tareas de reconocimiento y clasificación de pliegues, los cuales han demostrado ser prometedores y con gran potencial para el campo.

Contents

I

Introduction

1 Introduction

It would take ages to compute all possible conformations
of a typical protein by brute force.
Yet proteins fold spontaneously in nature,
some within milliseconds.

— Levinthal's Paradox

1.1 Background

Proteins are macromolecules essential for life that play critical roles in many biological processes [1]. They are composed by chains of varying length, which contain 20 different types of amino acids. The linear arrangement of these amino acids constitutes the primary structure of the protein. Moreover, the physicochemical properties of amino acids cause them to interact with one another. Therefore, this variable length sequence guides each protein to adopt a specific three-dimensional (3D) structure in the space, also called 'conformation'. In turn, the biological function performed by the protein is conditioned by its structure [2, 3], so its determination provides highly relevant information in the field of molecular biology.

Traditional procedures for obtaining the protein 3D structure have been based on experimental methods such as X-ray crystallography or nuclear magnetic resonance (NMR) spectroscopy. However, these methods are costly compared to massive genome sequencing techniques [4], which generate large amounts of uncharacterized protein sequences that are hosted in public databases [5]. This has led to extensive development of algorithms that exploit the information contained in the amino acid sequence to predict the protein 3D structure.

Historically, the computational estimation of the 3D structure at atomic level has been considered a highly complex problem [6]. The main developments of the last few decades have been driven by the CASP (Critical Assessment of Structure Prediction) challenges [7, 8] in which, until very recently, two main modeling approaches were adopted: template-based [9] and *ab initio* [10]. While *ab initio* techniques aim to build the structure from scratch, template-based modeling is used to infer the structure of a query protein from a set of templates with known structure, by looking for similarities in sequence or structure. In template-based modeling, one of the steps is to predict the fold type the protein belongs to, a problem known as *protein fold recognition* in the literature [11–13]. Fold categories group proteins that share the spatial arrangement of their secondary structure elements and the topological connections. The task can be seen as either a pairwise identification problem or a supervised classification problem, since the number of possible folds or structural motifs is thought to be limited in nature [14]. In fact, the pace of adding new folds to the different versions of the SCOPe (Structural Classification of Proteins—extended) database has been rather slow during the last decade [15, 16].

More recently, advances in machine learning techniques, particularly in deep learning [17], have led to major breakthroughs in the field during the two latest CASP challenges, in which the AlphaFold methods [18, 19] succeeded in producing accurate protein structures with high resolution at atomic level. Despite this, there is still interest in developing algorithms to automatically find sequential and structural similarities between proteins at reduced computational cost, since they can disclose evolutionary or functional relationships. Given that structures are usually better conserved than sequences [20], protein fold prediction can also be a means to discover proteins that are structurally related to each other without necessarily sharing high sequence similarities.

In this Thesis we address the protein fold prediction problem by taking advantage of the latest developments in deep neural networks (DNN) and representation learning techniques. To this end, we propose deep learning-based models adapted to different information from the protein, such as the evolutionary profile or PSSM (position-specific scoring matrix), the secondary structure (SS), or even the 3D structure in the form of distance and contact maps. In addition, while most of this information has traditionally been estimated from the multiple sequence alignment (MSA), recently proposed language models trained on amino acid sequences learn embeddings that are informative of protein attributes. Therefore, we evaluate the impact of using either traditional or learned protein representations as input to DNN models that are trained to map proteins into all possible fold types. These models also allow the extraction of new feature embeddings that are representative of the fold, which can be used to compare proteins using simple similarity metrics.

This doctoral Thesis is in agreement with the Doctoral Programme in Information and Communication Technologies of the University of Granada and is presented as a compendium of scientific papers—both journal publications and conference contributions. Three of these papers have been published in indexed journals and are included here without modification. In total there are seven chapters in this dissertation. This Chapter 1 introduces the research topic and its background, describes all the elements necessary to understand this Thesis as a whole, and also covers the objectives, contributions and publications resulting from this Thesis. The core experimental work is included in Chapters 2 to 6. First, Chapters 2 and 3 cover the initial work done with protein contact maps. Then, the three published papers on protein fold prediction (as part of the compendium) can be found in Chapters 4, 5 and 6. Thus, all experimental chapters are self-contained and prepared to be read individually. Most of them include the following sections: Abstract, Introduction, Materials and Methods, Results, Conclusion, and References. Therefore, given the nature of the compendium format, some unavoidable repetitions may be found in this manuscript. Finally, Chapter 7 brings together all the conclusions drawn from this Thesis, and summarizes possible lines of future work.

Antecedentes

Las proteínas son macromoléculas esenciales para la vida, las cuales juegan un papel fundamental en muchos procesos biológicos [1]. Están compuestas por cadenas de longitud variable, formadas por la combinación de 20 tipos distintos de aminoácidos. La disposición lineal de estos aminoácidos constituye la estructura primaria de la proteína. Además, las propiedades físico-químicas de los aminoácidos hacen que estos interactúen entre sí. Por tanto, esta secuencia de longitud variable es la que guía a cada proteína a adoptar una estructura tridimensional (3D) específica, también denominada “conformación”. A su vez, la función biológica que realiza la proteína está condicionada por su estructura [2, 3], por lo que su determinación aporta información altamente relevante en el campo de la biología molecular.

Los procedimientos tradicionales para la obtención de la estructura tridimensional se han basado en métodos experimentales como la cristalografía de rayos X o la espectroscopía de resonancia magnética nuclear (NMR). Sin embargo, estos métodos son costosos en comparación con las técnicas de secuenciación masiva de genomas [4], las cuales generan una gran cantidad de secuencias de proteína sin caracterizar que están siendo alojadas en bases de datos públicas [5]. Esto ha propiciado un amplio desarrollo de algoritmos que explotan la información contenida en la secuencia de aminoácidos para predecir la estructura 3D de la proteína.

Históricamente, la estimación por medios computacionales de la estructura 3D a nivel atómico se ha considerado un problema altamente complejo [6]. Los principales avances de las últimas décadas han sido impulsados por los retos CASP (*Critical Assessment of*

Structure Prediction) [7, 8] en los cuales, hasta hace muy poco, se adoptaron dos enfoques principales para el modelado de estructuras: *template-based* [9] y *ab initio* [10]. Mientras que las técnicas *ab initio* pretenden construir la estructura desde cero, el modelado de tipo *template-based* se utiliza para inferir la estructura de una proteína a partir de un conjunto de modelos con estructura conocida, buscando similitudes en la secuencia o la estructura. En el modelado de tipo *template-based*, uno de los pasos es predecir el tipo de plegamiento al que pertenece la proteína, un problema conocido en la literatura como *protein fold recognition* [11–13]. Las categorías de plegamientos agrupan aquellas proteínas que comparten la disposición espacial de sus elementos de estructura secundaria, así como las conexiones topológicas entre ellos. La tarea puede verse como un problema de identificación por pares o de clasificación supervisada, ya que se cree que el número de posibles plegamientos o motivos estructurales está limitado en la naturaleza [14]. De hecho, el ritmo de incoporación de nuevos plegamientos en las diferentes versiones de la base de datos SCOPe (*Structural Classification of Proteins-extended*) ha sido bastante lento durante la última década [15, 16].

Más recientemente, los avances en las técnicas de aprendizaje automático, especialmente en el aprendizaje profundo [17], han dado lugar a importantes adelantos en este campo durante los dos últimos retos CASP, en los que los métodos AlphaFold [18, 19] lograron producir estructuras de proteína precisas y con alta resolución a nivel atómico. A pesar de ello, sigue habiendo interés en desarrollar algoritmos para encontrar automáticamente similitudes secuenciales y estructurales entre proteínas a un coste computacional reducido, ya que podrían revelar relaciones evolutivas o funcionales. Dado que las estructuras suelen estar mejor conservadas que las secuencias [20], la predicción del tipo de plegamiento de la proteína también puede ser un medio para descubrir proteínas relacionadas entre sí a nivel estructural, sin que necesariamente posean grandes similitudes de secuencia.

En esta Tesis se aborda el problema de la predicción del tipo de plegamiento de las proteínas haciendo uso de los últimos avances en redes neuronales profundas (DNN) y técnicas de aprendizaje de representaciones. Para ello, se proponen modelos basados en *deep learning* y adaptados a diferentes tipos de información de la proteína, como el perfil evolutivo PSSM (*position-specific scoring matrix*), la estructura secundaria (SS), o incluso la estructura 3D en forma de mapas de distancias y contactos. Además, aunque la mayor parte de esta información se ha estimado tradicionalmente a partir del alineamiento múltiple de secuencias (MSA), los recientemente propuestos modelos de lenguaje entrenados con secuencias de aminoácidos aprenden representaciones (*embeddings*) que contienen información acerca de distintos atributos de las proteínas. Por lo tanto, tambien se evalúa el impacto de utilizar representaciones de proteínas tradicionales o aprendidas como entrada a los modelos DNN entrenados para clasificar las proteínas en todos los tipos de plegamiento posibles. Estos modelos también permiten la extracción de nuevos *embeddings* de características que son representativos del plegamiento, los cuales pueden utilizarse para comparar proteínas utilizando métricas de similitud simples.

Esta Tesis doctoral se ajusta al Programa de Doctorado en Tecnologías de la Información y la Comunicación de la Universidad de Granada, y se presenta como un compendio de artículos científicos—tanto publicaciones en revista como contribuciones a congreso. Tres de estos artículos han sido publicados en revistas indexadas y se incluyen aquí sin ningún tipo de modificación. En total hay siete capítulos en esta Tesis. Este Capítulo 1 presenta el tema de investigación y sus antecedentes, describe todos los elementos necesarios para entender esta Tesis en su conjunto, y también cubre los objetivos y las contribuciones derivadas de esta Tesis. El trabajo experimental principal se incluye en los Capítulos 2 a 6. En primer lugar, los Capítulos 2 y 3 abarcan el trabajo inicial realizado con mapas de contactos. A continuación, los tres artículos publicados del compendio se encuentran en los Capítulos 4, 5 y 6, respectivamente. Así, todos los capítulos experimentales son independientes y están preparados para ser leídos de manera individual. La mayoría de ellos incluyen las siguientes secciones: Resumen, Introducción, Materiales y Métodos, Resultados, y Referencias. Por lo tanto, dada la naturaleza del formato de compendio, se pueden encontrar algunas repeticiones inevitables en este manuscrito. Finalmente, el Capítulo 7 recoge todas las conclusiones extraídas de esta Tesis, y resume las posibles líneas de trabajo futuras.

1.2 Fundamentals of protein structure

Levels of structure

Proteins are composed of one or more long chains of amino acids (polypeptides) linked together through peptide bonds in a specific order, that fold into a specific three-dimensional shape. There are four levels involved in the protein structure: primary, secondary, tertiary, and quaternary (Figure 1.1) [21, 22].

(i) The *primary structure* (Figure 1.1a) is defined by the linear sequence of amino acids that compose the polypeptide chain. This information is determined by the DNA sequence

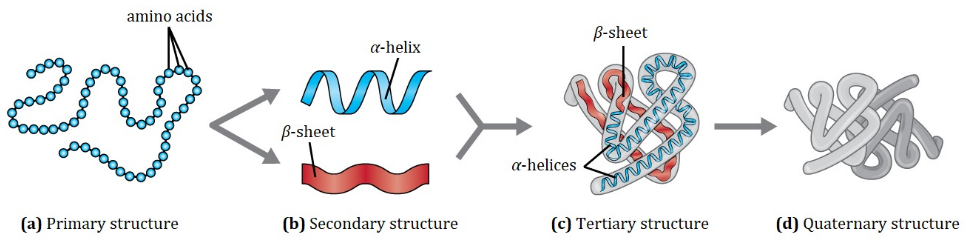

amino acids
α-helix
β-sheet
β-sheet
α-helices
(a) Primary structure
(b) Secondary structure
(c) Tertiary structure
(d) Quaternary structure

Figure 1.1: Four levels of protein structure: **(a)** primary structure, **(b)** secondary structure, **(c)** tertiary structure, and **(d)** quaternary structure. Adapted from [21].

Table 1.1: The 20 common amino acid types, their three- and one-letter abbreviation codes and one attribute of their side-chain: polar (hydrophilic) and nonpolar (hydrophobic). From [1].

Polar amino acids			*Nonpolar amino acids*		
Amino acid	3-letter code	1-letter code	Amino acid	3-letter code	1-letter code
Aspartic acid	Asp	D	Alanine	Ala	A
Glutamic acid	Glu	E	Glycine	Gly	G
Arginine	Arg	R	Valine	Val	V
Lysine	Lys	K	Leucine	Leu	L
Histidine	His	H	Isoleucine	Ile	I
Asparagine	Asn	N	Proline	Pro	P
Glutamine	Gln	Q	Phenylalanine	Phe	F
Serine	Ser	S	Methionine	Met	M
Threonine	Thr	T	Tryptophan	Trp	W
Tyrosine	Tyr	Y	Cysteine	Cys	C

of the gene that encodes the protein. There are 20 common different types of amino acids (Table 1.1), all of them having an amino group (H_2N) and a carboxyl group (CO_2H), but a particular side-chain (R group), attached to their alpha-carbon (C_α) atom (Figure 1.2). To maintain the primary structure, covalent peptide bonds connect the amino acids together, from the C atom in the carboxyl group of one amino acid to the N atom in the amino group of the next one. This repeating sequence of core atoms ($N{-}C_\alpha{-}C$) constitutes the backbone of the protein, while the side-chain determines the unique properties of each amino acid. The group of backbone atoms and side-chain is commonly referred to as a *residue*. Following the order of backbone atoms for each amino acid in the sequence, the protein chain folds from the N-terminus to the C-terminus.

(ii) The *secondary structure* (Figure 1.1b) refers to local elements that fold due to interactions between atoms of the polypeptide backbone. The secondary structure adopts two common forms: α-helix and β-sheet. The shape of these local structures is held by hydrogen bonds formed between the carboxyl group of one amino acid and the amino group of another. α-helices are formed when one amino acid is hydrogen bonded to another separated by four amino acids in the chain. This results in a right-handed helical shape containing 3.6 residues per turn (Figure 1.3a). By contrast, β-sheets are composed of at least two adjacent β-strands that are hydrogen bonded. Each individual β-strand is a segment of polypeptide chain with an almost fully extended conformation. Two particular forms of β-sheet are usually found: parallel, where all strands follow the same direction from N- to C-terminus; and antiparallel, where the strands point in opposite directions (Figure 1.3b). Although the side-chains (R groups) are not directly involved in the formation of α-helices and β-strands, they can influence the type of secondary structure element that is created. An example is the amino acid proline, which cannot form part of a helix because its R group forms a ring with the amino group, bending the chain in that region. In addition, some segments of the protein chain acquire simpler shapes, known as loops or coils, which

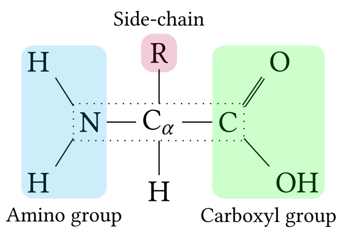


Figure 1.2: Molecular composition of atoms in the amino acids, formed by an amino group (blue), a carboxyl group (green), the backbone atoms (dotted), and the side-chain (red) attached to the central C_α atom.

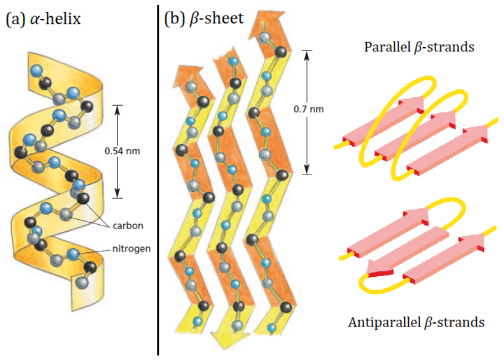


Figure 1.3: Secondary structure elements: **(a)** α-helix and **(b)** β-sheet (parallel and antiparallel β-strands). From [1].

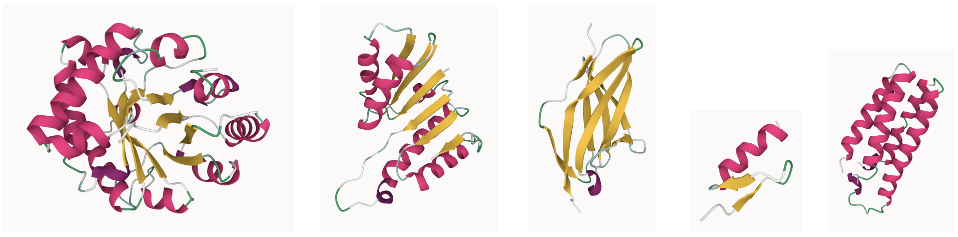
are responsible for connecting helices and strands in protein structures.

(iii) The *tertiary structure* (Figure 1.1c) is the overall three-dimensional structure of the polypeptide chain described by the spatial location of all backbone and side-chain atoms, as well as the spatial arrangement of secondary structure elements. In this case, the R groups are fully involved in the formation of the tertiary structure. The R groups interact with each other contributing to the stabilization of the protein through hydrogen bonds, disulfide bonds, electrostatic forces, and Van der Waals forces. It is this tertiary structure that gives polypeptide chains a very specific shape, which allows proteins to interact with other molecules providing them with a unique function.

(iv) The *quaternary structure* (Figure 1.1d) is the spatial arrangement of subunits within a protein formed as a complex of multiple polypeptide chains.

Domains, folds, and motifs

The tertiary structure of a protein chain can be further divided into evolutionary, functional segments called *domains*. These units can fold independently into compact, stable three-dimensional structures that can be associated to specific functions. Protein domains usually comprise between 40 and 350 residues, and are considered the building blocks for many larger proteins. While small proteins usually contain one single domain, larger proteins can be composed of multiple domains. The term *fold* refers to a particular spatial arrangement of secondary structure elements (α-helices, β-sheets, or loops) into a domain structure, which is common to different proteins. In contrast, *structural motifs* are generally smaller than folds and can be contained within them. Examples of folds and motifs can be found in Figure 1.4. While some structural motifs are observed in many unrelated proteins with large variability, others appear in protein domains with similar function and are therefore highly conserved. These share a characteristic amino acid sequence called *sequence motif*,



(a) TIM-barrel **(b)** Rossmann **(c)** Beta-sandwich **(d)** Zinc-finger **(e)** Helix bundle

Figure 1.4: Graphical visualization (cartoon mode) of protein folds and motifs. For each domain fold α-helices are shown in pink, β-strands in yellow, and loop regions in light green/gray. **(a)** TIM-barrel fold formed by a β-strand followed by an α-helix, repeated eight times (PDB ID: 1SQ7_A). **(b)** Rossmann fold containing an alpha/beta twist that usually binds NAD cofactors (PDB ID: 1EMD_A). **(c)** Immunoglobulin-like beta-sandwich fold consisting of 7 or 9 antiparallel β-strands arranged in two β-sheets with a Greek-key motif topology (PDB ID: 2AW2_A). **(d)** Zinc-finger motif formed by an α-helix and an antiparallel β-strand, usually found in DNA-binding proteins (PDB ID: 1AAY_A). **(e)** Four-helix bundle motif (PDB ID: 1M6T_A). The structure information for each domain was taken from the SCOPe version 2.08 database [16], and the figures were created with Mol* Viewer [24].

although not all structural motifs have a unique amino acid sequence. It should be noted that the terms described here—domain, fold and motif—sometimes have blurred distinctions and are used interchangeably, especially in the case of large, highly preserved structural motifs that can be identified as a particular type of domain or fold [1, 22, 23].

1.3 Representations of proteins

The sequence-to-structure nature of proteins allows for feature representations derived from two different modalities: sequence (attributes of each amino acid) and structure (3D location of atoms within each residue). To illustrate the different representations, in the following we use the small protein ubiquitin (76 amino acids) as an example. Information on its sequence and structure has been obtained from the PDB website (`https://rcsb.org/structure/1ubi`).

Sequence features

The amino acid sequence is commonly represented as a string of variable length L, where each position can take one of the 20 unique characters in Table 1.1 (usually following the order ARNDCEQGHILKMFPSTWYV). Protein sequences can also contain ambiguous (B=`[N,D]`, J=`[I,L]`, Z=`[E,Q]`), or unknown (X) amino acids. An illustration of amino acid sequence stored in a FASTA file format can be seen in Example 1.1. The most straightforward way to encode this sequence is to use a one-hot representation for each native amino acid [25]. This protein encoding is orthogonal and sparse, with a total size $L \times 20$, but lacks any information about the attributes of individual amino acids, let alone

Example 1.1: FASTA file content for the protein with PDB ID: 1UBI_A (`1ubi.fasta`).

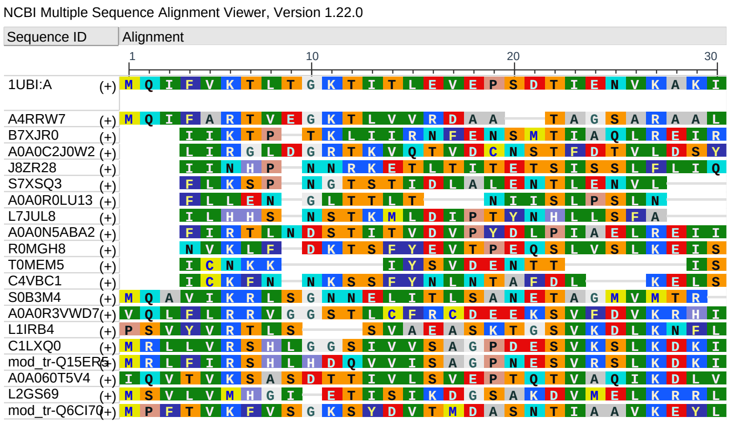
```
>1UBI_1|Chain A|UBIQUITIN|Homo sapiens (9606)
MQIFVKTLTGKTITLEVEPSDTIENVKAKIQDKEGIPPDQQRLIFAGKQLEDGRTLSDYNIQKESTLHLVLRLRGG
```

the relative position of the amino acids in the sequence. To solve the first, each position of the sequence can be instead represented as a vector of biochemical properties for the amino acid, which can range from 3 values—the hydrophobicity, the hydrophilicity, and the side-chain mass [26]—to the more than 500 physicochemical properties collected in the AAIndex database [27]. To account for evolutionary information, the scoring matrices PAM (point accepted mutation) [28] and BLOSUM (blocks substitution matrix) [29] can be used. Both matrices are obtained by aligning homologous proteins, i.e. those that share common ancestors in the evolutionary tree and therefore have similar sequences (more than 25% shared sequence identity), and then measuring the frequencies of amino acid substitutions in those related proteins. The final matrices contain the log-odds ratio for each of the 210 possible substitution pairs of the 20 standard amino acids. While PAM matrices are constructed from global alignments, BLOSUM matrices are developed from local conserved blocks or clusters of aligned proteins. An example is the widely used BLOSUM62 matrix, built from amino acid sequences with a maximum pairwise identity of 62% within each cluster. Table 1.2 provides an example of sequence representation using one-hot encoding, 3 physicochemical properties [26], and evolutionary relationships (BLOSUM62 matrix).

In fact, these substitution matrices play an important role in the building of *multiple sequence alignments* (MSA) for proteins (see Figure 1.5), as they are used to evaluate the quality of the alignment in tools such as BLAST [30]. The idea behind these tools is to search for homologous proteins that are similar to a query protein (globally or locally) in a sequence database, and sequentially add them to the alignment including gaps if required. Then, the position-specific scoring matrix (PSSM) profile is usually computed from this alignment. This is a frequency matrix of the number of occurrences of each residue at each position of the alignment. The scores are computed as the log-likelihood ratio of the

Table 1.2: Partial sequence representations for the protein with PDB ID: 1UBI_A.

AA seq	One-hot encoding ($L \times 20$)	Physicochemical properties ($L \times 3$)	BLOSUM62 log-odds ($L \times 20$)
M	[0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0]	[0.64, −1.3, 75.0]	[−1,−1,−2,−3,−1, 0,−2,−3,−2, 1, 2,−1, 5, 0,−2,−1,−1,−1,−1, 1]
Q	[0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0]	[−0.85, 0.2, 72.0]	[−1, 1, 0, 0,−3, 5, 2,−2, 0,−3,−2, 1, 0,−3,−1, 0,−1,−2,−1,−2]
I	[0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0]	[1.38, −1.8, 57.0]	[−1,−3,−3,−3,−1,−3,−3,−4,−3, 4, 2,−3, 1, 0,−3,−2,−1,−3,−1, 3]
F	[0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0]	[1.19, −2.5, 91.0]	[−2,−3,−3,−3,−2,−3,−3,−3,−1, 0, 0,−3, 0, 6,−4,−2,−2, 1, 3,−1]
V	[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1]	[1.08, −1.5, 43.0]	[0,−3,−3,−3,−1,−2,−2,−3,−3, 3, 1,−2, 1,−1,−2,−2, 0,−3,−1, 4]
⋮	⋮	⋮	⋮
G	[0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0]	[0.48, 0.0, 1.0]	[0,−2, 0,−1,−3,−2,−2, 6,−2,−4,−4,−2,−3,−3,−2, 0,−2,−2,−3,−3]



```
NCBI Multiple Sequence Alignment Viewer, Version 1.22.0
Sequence ID       Alignment
                  1                 10                  20                  30
1UBI:A        (+) M Q I F V K T L T G K T I T L E V E P S D T I E N V K A K I

A4RRW7        (+) M Q I F A R T V E G K T L V V R D A A - - T A G S A R A A L
B7XJR0        (+)       I I K T P - T K L I I R N F E N S M T I A Q L R E I R
A0A0C2J0W2    (+)       L I R G L D G R T K V Q T V D C N S T F D T V L D S Y
J8ZR28        (+)       I I N H P - N N R K E T L T I T E T S I S S L F L I Q
S7XSQ3        (+)       F L K S P - N G T S T I D L A L E N T L E N V L - - -
A0A0R0LU13    (+)       F L L E N - G L T T L T - - - N I I S L P S L N - - -
L7JUL8        (+)       I L H H S - N S T K M L D I P T Y N H L L S F A - - -
A0A0N5ABA2    (+)       F I R T L N D S T I T V D V P Y D L P I A E L R E I I
R0MGH8        (+)       N V K L F - D K T S F Y E V T P E Q S L V S L K E I S
T0MEM5        (+)       I C N K K - - - - - I Y S V D E N T T - - - - - - I S
C4VBC1        (+)       I C K F N - N K S S F Y N L N T A F D L - - - K E L S
S0B3M4        (+) M Q A V I K R L S G N N E L I T L S A N E T A G M V M T R -
A0A0R3VWD7    (+) V Q L F L R R V G G S T L C F R C D E E K S V F D V K R H I
L1IRB4        (+) P S V Y V R T L S - - - S V A E A S K T G S V K D L K N F L
C1LXQ0        (+) M R L L V R S H L G G S I V V S A G P D E S V K S L K D K I
mod_tr-Q15ER5 (+) M R L F I R S H L H D Q V V I S A G P N E S V R S L K D K I
A0A060T5V4    (+) I Q V T V K S A S D T T I V L S V E P T Q T V A Q I K D L V
L2GS69        (+) M S V L V M H G I - E T I S I K D G S A K D V M E L K R R L
mod_tr-Q6CI70 (+) M P F T V K F V S G K S Y D V T M D A S N T I A A V K E Y L
```

Figure 1.5: Partial MSA representation for the protein with PDB ID: 1UBI_A (amino acids 1 to 30).

observed to the expected frequencies. PSI-BLAST [31], for example, is a well-known tool to compute and refine PSSM matrices in an iterative manner, using previously computed matrices to repeatedly search for sequences in a database. As a result, the PSI-BLAST PSSM matrix is a customized scoring matrix that is more sensitive than PAM or BLOSUM for a specific query. Probabilistic models, such as hidden Markov models (HMM), can be also built from MSAs using the HMMER [32] and HHblits tools [33]. HMM profiles extend PSSM scores by adding position-specific penalties for insertions and deletions. Furthermore, the profile matrices generated by these tools are used to search for remote homologous sequences in databases with pre-computed profiles, and so build more diverse multiple sequences alignments.

To include information about the context of amino acids in the sequence, several protein-level representations have been proposed. An example is the k-mer composition [34, 35], which computes the occurrence of each possible short subsequence of length k amino acids in the sequence. This representation is high dimensional, with size 20^k. When $k = 1$, it corresponds to the amino acid composition, which simply computes the frequency of each individual amino acid in the sequence. Other examples are the PseAAC (pseudo amino acid composition) descriptor [26, 36] and the PSSM auto-covariance (PSSM-AC) [37]. While both measure the correlation between pairs of amino acids separated by different distances along the sequence, the PseAAC uses biochemical properties and the PSSM-AC uses evolutionary information from the PSSM profiles.

In recent years there has been a boom in deep-learned representations for proteins, borrowing concepts from the field of NLP (natural language processing). In particular, language models (LM) have been adapted to protein sequences, where amino acids are equivalent to words (tokens) and the protein sequence to a sentence. Currently there exist many examples of *protein language models* (protein-LM) [38], starting from the word2vec models [39, 40], followed by the contextualized auto-regressive ELMo model [41, 42], and more recently the transformer-based language model BERT and its variants [43–45]. These models are trained in a self-supervised manner by either predicting the context of an amino acid, the following amino acid, or randomly masked amino acids in a sequence. From the trained protein-LMs, deep representations known as *embeddings* are extracted for every amino acid in the protein, which can be averaged along the sequence dimension to obtain protein-level embeddings.

Structure features

As opposed to sequence features that are extracted from the amino acid sequence alone, structure features take into account the position and relation of the residues in the space. The PDB (protein data bank) [46] file format is usually used to store the 3D coordinates for every residue atom (backbone and side-chain) in the protein (see Example 1.2), along with other information about the determination of the protein structure.

Given the 3D position of atoms in a protein, the secondary structure assignment for every residue can be determined by identification of the hydrogen bonds as, for example, the DSSP program [47] does. DSSP provides 8 states for the secondary structure: H (α-helix), E (β-strand), C (random coil), G (3_{10}-helix), I (π-helix), B (β-bridge), S (bend), and T (H-bonded turn); which can be further assigned into one of the 3 common states: H (helix: H, G and I), E (strand: E and B), and C (loop: C, S and T). Together with the secondary structure states, the angles between bonds within a residue and in consecutive residues can be also computed. In a protein, each residue has two dihedral or rotational angles [48]: ϕ about the N–C_α bond and ψ about the C_α–C bond (see Figure 1.2). The third angle ω between the carboxyl and amino groups (C–N bond) is restricted to 180° (mostly) or 0° due to rigid planar peptide bonds. In addition, two other angles (θ and τ) are essential to define the overall backbone structure. For the i-th residue in the sequence, θ_i is the angle between

Example 1.2: Partial PDB file content for the protein with PDB ID: 1UBI_A (`1ubi.pdb`).

```
HEADER     CHROMOSOMAL PROTEIN     03-FEB-94     1UBI
ATOM       1  N    MET A    1     27.343  24.294   2.683  1.00 14.70     N
ATOM       2  CA   MET A    1     26.381  25.361   2.894  1.00  9.58     C
ATOM       3  C    MET A    1     26.997  26.557   3.583  1.00  6.78     C
ATOM       4  O    MET A    1     27.973  26.439   4.351  1.00 11.04     O
ATOM       5  CB   MET A    1     25.112  24.879   3.647  1.00 15.53     C
ATOM       6  CG   MET A    1     25.341  24.685   5.142  1.00 18.33     C
ATOM       7  SD   MET A    1     23.944  23.887   5.986  1.00 16.83     S
ATOM       8  CE   MET A    1     24.546  23.890   7.694  1.00  9.81     C
ATOM       9  N    GLN A    2     26.410  27.694   3.332  1.00 10.15     N
ATOM      10  CA   GLN A    2     26.865  28.934   3.898  1.00  8.89     C
....
```

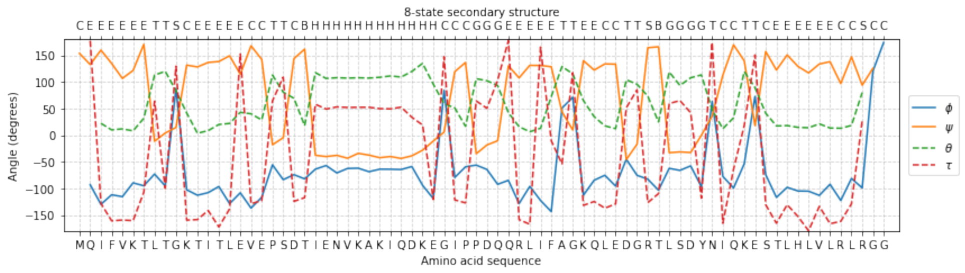
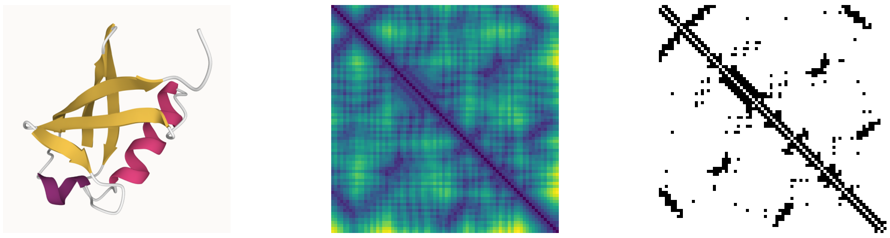


Figure 1.6: Secondary structure assignment (8-state) and backbone angles for the protein with PDB ID: 1UBI_A.

atoms $C_{\alpha_{i-1}}$–C_{α_i}–$C_{\alpha_{i+1}}$, and τ_i is the dihedral angle rotated about the C_{α_i}–$C_{\alpha_{i+1}}$ bond. An example of the 8-state secondary structure assignment and the torsion angles given by DSSP is shown in Figure 1.6. We observe that the angles take on more constant values for consecutive H (α-helix) and E (β-strand) states. This can be compared to the graphical visualization of the 3D structure in Figure 1.7a, in which the α-helix and β-strand elements can be easily identified. Another property related to the secondary structure is the solvent accessible surface area [49] which is a function of the protein structure and can be used to classify the amino acid residues as solvent exposed or buried (non-exposed).

Tertiary structure features of a protein are often calculated as spatial distances between atoms in the protein backbone. This representation is known as *distance map* (Figure 1.7b), which is an $L \times L$ real-valued symmetric matrix containing the Euclidean distance between each pair of beta-carbon (C_β) atoms within the protein. The C_β is the first atom of the side-chain in an amino acid residue (since the amino acid Glycine lacks a C_β atom, its C_α atom is used instead for the computation). From this matrix, the binary *contact map* (Figure 1.7c) is computed by applying a distance threshold over the distance map, usually from 6 to 12 Å (Angstroms). The contact map indicates which residues are close enough in space to form an interaction, and therefore contains more information about the 3D structure than the secondary structure representation. Moreover, it is a translation and rotation invariant representation that can be used as a constraint to reconstruct the 3D structure of the protein [50].

It is important to note that the number of experimentally determined protein structures is less abundant than the number of existing amino acid sequences. Therefore, there is a special interest in predicting structural attributes from the information contained in the protein sequences. Most methods rely on evolutionary information from the MSAs to extract representations that are useful for protein structure prediction. One example is the PSSM and HMM profiles, which have proven to be highly relevant in the prediction of 1D structural features (i.e. secondary structure, solvent accessibility, and backbone angles) [51]. Another example is the prediction of contact maps by identifying co-evolutionary



(a) PDB structure **(b)** Distance map **(c)** Contact map

Figure 1.7: Tertiary structure representations in image form for the protein with PDB ID: 1UBI_A.

signals in pairs of MSA columns [52, 53]. These positions could indicate residues that interact in space and thus may undergo correlated mutations during evolution—i.e. if one residue mutates, its counterpart has to mutate as well to preserve the overall 3D structure.

1.4 Protein fold prediction tasks

The main goal of this Thesis is to develop methods to predict the fold type of protein domains, particularly from their amino acid sequence. To formalize this task, we start by defining the input and output of our prediction models. The input is a protein domain with L amino acid residues. Depending on the representation we choose, we can denote the protein domain either as $\mathbf{X} \in \mathbb{R}^{L \times d}$ for residue-level sequential features (e.g. one-hot encoding, PSSM matrix, secondary structure assignment, backbone angles, solvent accessibility, etc.); $\mathbf{X} \in \mathbb{R}^{d}$ for protein-level features (e.g. PseAAC, PSSM-AC, average protein-LM embeddings, etc.); or $\mathbf{X} \in \mathbb{R}^{L \times L}$ for distance and contact maps. In all cases, L denotes the sequence length and d the dimension of the feature vectors representing each amino acid residue. The output is a fold class c taken from a pre-defined closed set of fold classes $c = \{c_1, c_2, \ldots, c_N\}$.

The protein fold prediction task can be handled in two different ways. On the one hand, it can be seen as a multi-class classification problem that can be tackled using common machine learning approaches. In this case, we aim to directly map the input query domain into one of the N fold classes using a classification model. We refer to this task as *direct fold*

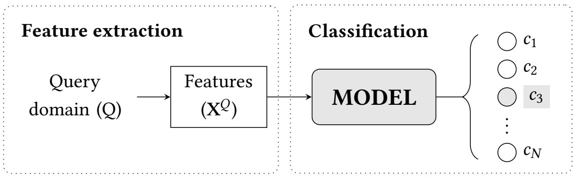


Figure 1.8: Direct fold classification (DFC) task. The features for the query domain are fed to a model that computes likelihoods for each one of the fold classes, and then the most likely class is assigned to the query.

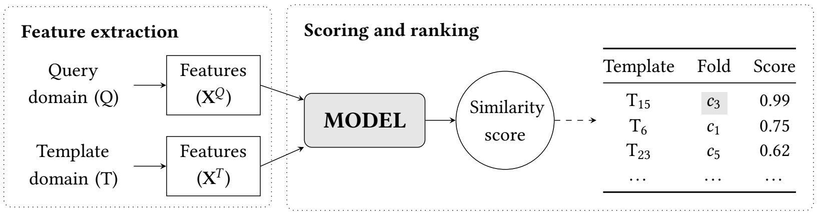


Figure 1.9: Pairwise fold recognition (PFR) task. The model processes the features for the query domain and compares to the features for a pool of template domains with known fold. The similarity scores computed between each pair of query-template domains are then used to rank all templates. Finally, the fold class of the most similar template is assigned to the query.

classification or DFC (see Figure 1.8). On the other hand, instead of directly outputting the fold class, the model could measure the similarity degree between pairs of protein domains. In such case, the query domain is compared to a pool of template domains for which we know their fold class. For each pair of query-template domains, the model computes a score that is used to rank the templates by similarity with respect to the query domain. Then, the most similar template provides the predicted fold class for the query domain. We denote this task as *pairwise fold recognition* or PFR (see Figure 1.9).

1.5 Databases

Protein sequences and structures

Protein sequence databases are collections of amino acid sequences, which are usually translated from nucleotide sequences found in resources that contain genome, gene, and transcript sequence data. Sequence data is provided in FASTA format (see Example 1.1) and is crucial to search for homologous proteins when building multiple sequence alignments (MSAs).

The UniProt Knowledgebase (UniProtKB) [5, 54] provides both sequence data and associated function information. UniProtKB (`https://uniprot.org`) comprises two main sections. First, the translations from EMBL (European Molecular Biology Laboratory) nucleotide entries (`https://ebi.ac.uk/ena`) are computer-annotated and automatically transferred to the TrEMBL section (unreviewed). Then, some of these records are selected for manual annotation and then integrated into the Swiss-Prot section (reviewed). The Swiss-Prot database provides protein annotations such as function descriptions, the domain structure, post-translational modifications and variants, by cross-linked integration with other external databases. The latest release of UniProtKB (`2022_02`) contains 231.9M protein sequences, out of which only 567.5k entries (0.24%) are stored in Swiss-Prot.

As part of UniProt we also find the UniProt Archive (UniParc), which collects non-redundant sequences from most of the publicly available sequence databases, including

UniProtKB, RefSeq (`https://ncbi.nlm.nih.gov/refseq`), and the Protein Data Bank (PDB, `https://rcsb.org`). As of July 2022, UniParc contains 505.7M protein sequences. Moreover, the UniProt Reference Clusters (UniRef) [55] provides clustered sets from UniProtKB and selected UniParc records representing the sequence space at several resolutions. Three sets have been created by removing sequence redundancy using the MMseqs2 algorithm [56]: (i) UniRef100, which combines identical sequences into a single entry, (ii) UniRef90, which clusters UniRef100 sequences together up to 90% sequence identity, and (iii) UniRef50, which clusters UniRef90 to 50% maximum sequence identity.

In order to further increase the number of protein sequences for MSA generation, metagenomic data collected from environmental samples [57] generates billions of protein sequences that are stored in databases such as MGnify [58] and the Big Fantastic Database (BFD) [19]. MGnify (`https://ebi.ac.uk/metagenomics`) provides a non-redundant protein set generated from thousands of publicly available studies, which contains more than 1 billion sequences. BFD (`https://bfd.mmseqs.com`) was constructed by collecting 2.5B protein sequences from UniprotKB (Swiss-Prot and TrEMBL), MetaClust [59], as well as the Soil and the Marine Eukaryotic Reference Catalogs assembled by PLASS [60].

Complementary to sequence databases, *protein structure databases* process and distribute experimentally determined 3D structure data. The best known repository is the Protein Data Bank (PDB) [46, 61], operated by the Research Collaboratory for Structural Bioinformatics (RCSB), which contains large molecules of proteins, nucleic acids and other biological macromolecules. Structure data is usually provided in PDB format (see Example 1.2), and can be visualized using dedicated graphics software such as Mol* Viewer (open, `https://molstar.org`) or PyMOL (commercial, `https://pymol.org`). As of July 2022, PDB contains 188.8k protein structures (`https://rcsb.org`), mainly determined by X-ray crystallography (87.2%), nuclear magnetic resonance (NMR) spectroscopy (6.5%) and electron microscopy (6.1%). Moreover, out of these, 59k structures have direct correspondence with UniProtKB entries.

To bridge the gap between available sequence and structure data, several repositories provide access to a large number of predicted structures. An example is the SWISS-MODEL Repository [62], which contains 3D protein models generated by automated homology modeling for relevant model organisms. Currently the repository contains over 2.2M predicted structures for UniProtKB entries (`https://swissmodel.expasy.org/repository`). More recently, the AlphaFold DB [63] has been released and includes 3D structures estimated by the AlphaFold2 deep learning-based model [19] for 992.3k proteins (as of July 2022), covering 48 model organisms (including the human proteome), as well as the majority of Swiss-Prot entries (`https://alphafold.ebi.ac.uk`).

Classification of domain structures

To record the structural similarities between proteins and their common evolutionary origins, proteins with solved structure in the PDB have been further split into domains and hierarchically classified according to structural and sequence similarities. This has led to several *protein structure classification databases* such as FSSP (Families of Structurally Similar Proteins) [64], SCOP (Structural Classification of Proteins) [15, 16, 65], CATH (Class, Architecture, Topology, Homology) [66, 67], and ECOD (Evolutionary Classification of Protein Domains) [68]. These databases differ in the methods of classifying structural data: FSSP is fully automated, SCOP is almost completely manually derived, and CATH and ECOD employ both automated procedures and manual annotations [68, 69]. Unlike SCOP and CATH, which assign protein domains into folds or superfamilies, FSSP and ECOD group domain structures according to evolutionary relationships alone, paying especial attention to distant relations [70].

Throughout this Thesis, we focus on the use of domain and fold definitions provided by SCOP [65] (manually derived) and its extended version SCOPe [15] (`https://scop.berkeley.edu`). SCOP is a hierarchical database in which protein domains are grouped by their secondary structure properties into four main levels: *structural class, fold, superfamily*, and *family* (see Figure 1.10). Top levels group domains with increasing structural similarities and do not imply homology (i.e. structural class, fold, superfamily), whereas bottom levels are based on evolutionary similarities (i.e. superfamily, family, protein). Starting at the bottom, the main levels of SCOP have the following characteristics:

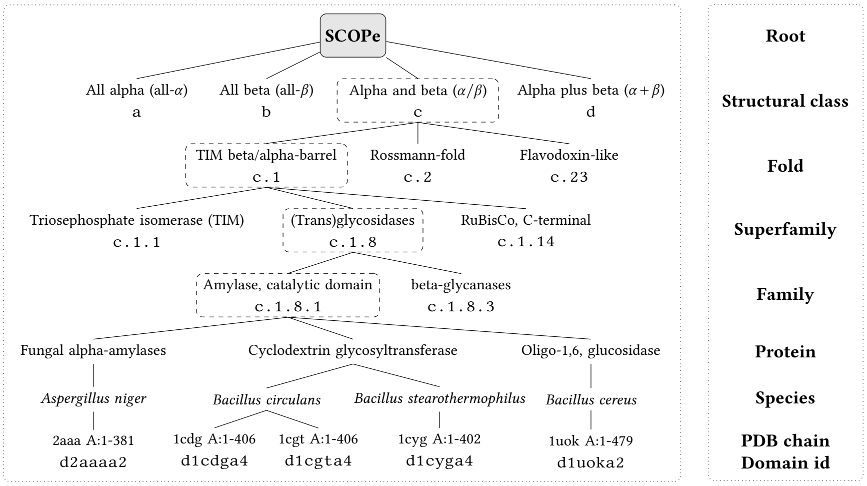


Figure 1.10: Example of the SCOPe hierarchy (adapted from `https://scop.berkeley.edu` for SCOPe version 2.08 [16]). At each level, we specify the identification codes for the corresponding structural classes, folds, superfamilies, and families. We also include the unique identifiers for the final domains at the bottom of the tree.

- **Family**. The family level groups proteins that have a clear common evolutionary origin, measured as either sharing more than 30% sequence identity or having similar functions and structures.
- **Superfamily**. The superfamily level groups families whose proteins have low sequence identities but their structures and functional features suggest that a common evolutionary origin is probable.
- **Fold**: The fold level groups superfamilies with proteins that have the same major secondary structures in the same arrangement and with the same topological connections. These structural similarities might arise from the physics and chemistry of proteins favoring certain motifs and chain topologies, or an unclear common evolutionary origin where the structure has been better conserved than the sequence.
- **Structural class**: The structural class level groups folds according to their secondary structure elements and organization. There are 7 major structural classes in SCOPe, termed from a to g. The most important ones are the following (Figure 1.10):
    a. All alpha (all-α): proteins whose structure is formed by α-helices.
    b. All beta (all-β): proteins whose structure is formed by β-sheets.
    c. Alpha and beta (α/β): proteins with α-helices and β-strands that are largely interleaved, usually arranged in beta-alpha-beta units.
    d. Alpha plus beta ($\alpha+\beta$): proteins in which α-helices and β-strands are spatially segregated in different parts of the protein.

Figure 1.11 shows the number of protein domains and folds contained in the different versions of the SCOP and SCOPe databases (from 2001 to 2021). In the case of SCOPe, we observe that while the number of domains has more than doubled, the number of folds has

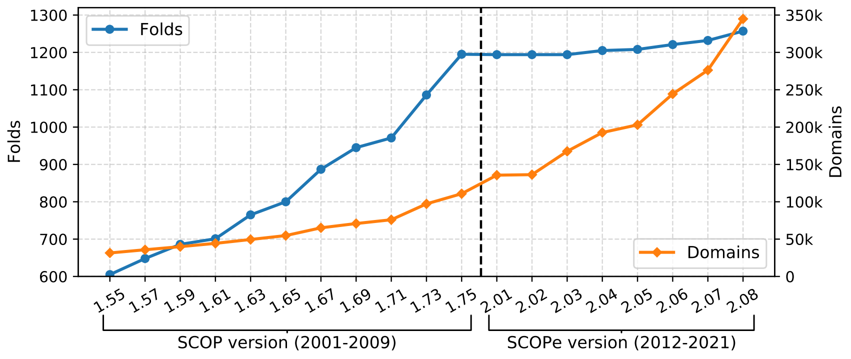


Figure 1.11: Number of protein domains and folds stored in the different versions of the SCOP (from 2001 to 2009) and SCOPe (from 2012 to 2021) databases. The information was taken from `https://scop.berkeley.edu`.

remained almost constant at around 1200 (from 1194 in v2.01 [15] to 1257 in v2.08 [16]). This is consistent with related literature suggesting that the number of folds is limited in nature [71–73].

1.6 Common approaches and open challenges

The protein fold identification task [74–77] has been tackled following two broad approaches: *alignment-based fold recognition*, in which the fold class of the query protein is inferred by finding similarities with templates with known structure; and *taxonomy-based fold classification*, in which machine learning methods are used to directly map the protein domains into their fold class.

Early alignment-based approaches tried to find proteins with the same fold by using sequence alignment methods [13] or by comparing the PSSM profiles, HMM profiles [78], or Markov random fields [79]. This is known in the literature as *homology modeling*, since it relies on finding sequences that are homologous to a query protein. While sequence alignment is conceptually simple and an effective method for certain sequences, it can easily fail when no close homologs exist for the query protein. Profile comparison approaches slightly remedy such a drawback by constructing an overall profile from summary statistics computed on the MSAs. However, they are also bottlenecked by the quality of the MSA, which can be again hard for those proteins without close homologous sequences. Consequently, *threading* methods [80–86] have been proposed to improve performance in remote homology. Here, structural properties can be used along with sequence profiles to perform sequence-to-structure alignment.

Taxonomy-based classification approaches use traditional machine learning (ML) methods such as support vector machines (SVM), random forests (RF) or neural networks (NN) to directly map an input protein domain into its fold class. The advantage is that once trained, these models offer significant speed gains in detecting fold types compared to alignment-based methods. However, a crucial limitation of traditional ML approaches is that they only consider the most populated fold classes: 27 or 30 in contrast to the more than 1000 folds in the SCOP database [37, 87–95].

While taxonomy-based methods view the task as a multi-class classification problem, another set of methods attempt to instead predict whether two protein domains belong to the same fold, casting the task into a binary classification problem [96–98]. To improve performance, ensemble methods have also been proposed, which combine multiple protein feature representations and prediction models [92, 99–101]. However all such models are limited to using protein-level representations, usually derived from evolutionary information (e.g. PSSM-AC). While the proteins themselves are variable in length, which itself carries information, these representations are computed to be fixed-sized vectors in order to work with traditional ML approaches.

A more promising approach lies in using deep representation learning methods. First, by scaling well with data, they allow for classification over the full set of SCOP folds (as demonstrated in the DeepSF method [102]), overcoming the limitations imposed by considering only populated folds. Second, the classification process can be a means to learn deep embeddings that are representative of the fold type. This strategy was proposed by the DeepFR [103] method and followed by more recent approaches [104–108]. These *fold-representative embeddings* can then be used to measure the structural similarity between two protein domains, providing a better score than previous alignment and threading methods. However, despite these improvements, existing deep learning approaches suffer from several drawbacks. The use of sampling/padding, cropping, or adaptive pooling strategies to fix the size previous to the classification layer discards information that could be useful for classification. In addition, while increasing performance for the hardest cases (proteins with same fold but different sequence), deep learning methods usually perform worse than alignment methods at the family level (i.e. when the sequences are similar).

1.7 Objectives

In this Thesis, we build upon the advantages of deep learning approaches to develop algorithms that improve the prediction accuracy of the protein fold type by leveraging sequence and structure information from different representations of the protein. In order to accomplish this, the following sub-objectives are defined:

1. To analyze the existing **protein datasets** from public databases of sequences (UniProt [54, 109]), structures (PDB [46]), and structural classification (SCOPe [15, 65]), as well as to adapt them to the training and evaluation tasks of the developed systems.
2. To design **deep neural network architectures** that process different representations of the protein—either sequential (for each amino acid residue), in image form (residue–residue pairs), or as a fixed-size vector (at the protein level).
3. To implement **representation learning strategies** for the neural network models to generate new features or embedding vectors for each protein that contain information about the protein fold in a compact form.
4. To develop and evaluate systems that improve accuracy in the **prediction of the protein fold type**, using machine learning algorithms on the learned fold-related features in combination with other representations extracted from the protein.

1.8 Contributions

This section presents the contributions derived from this Thesis and their correspondence with the different Chapters of this dissertation.

Contact maps for protein fold recognition. In Chapter 2 we analyze two approaches to reduce Gaussian noise in estimated protein contact maps. The best performing method trains a fully-convolutional neural network (FCN) for image denoising. This strategy is used in Chapter 3 to generate enhanced contact and categorical maps, which are then compared against the native maps obtained from the protein 3D structure in terms of fold recognition performance. To do so, we use a deep convolutional neural network (DCNN) model that processes the input images through several 2D convolutional layers and then performs the classification into fold types through fully-connected layers. Even though the fold-related features extracted from native contact maps are effective at recognizing proteins with the same fold, there is still room for improvement which could be achieved by using other types of information.

Neural network architectures adapted to variable-length protein sequences. Protein sequences are inherently variable in length. Previous approaches for protein fold recognition have used fixed-size features as input to the machine learning models. We argue that such representations result in a loss of information and the entire protein sequence should be used instead. In Chapter 4 we introduce a method to tackle this issue by training a neural network model on protein residue-level features—i.e. one feature vector for each amino acid residue in the sequence including evolutionary and secondary structure information. The architecture of the model consists of a convolutional part (CNN), followed by a bidirectional gated recurrent unit-based (BGRU) layer that generates a fixed-size representation, and fully-connected layers to perform the final classification from this representation. As a result, the fold-related embeddings extracted from the CNN-BGRU architecture perform better at recognizing the fold class than those learned from estimated contact maps. Furthermore, we observe a performance gain in our CNN-BGRU-RF+ method, which combines the pairwise scores calculated from the two sources (residue-level features and contact maps) with other similarity measures in a random forest (RF) model.

Learning strategies for better fold-discriminative embeddings. When analyzing the state-of-the-art results on protein fold recognition, we find that there exists a performance gap at the fold level compared to the relatively easier family level. This suggests that it might be possible to learn an embedding space that better discriminates between protein folds. To that end, in Chapter 5 we focus on the learning part of the neural network and propose a two-stage training procedure called FoldHSphere. The method first obtains prototype vectors for each fold class that are maximally separated in hyperspherical space.

We then train a residual-convolutional and recurrent (ResCNN-BGRU) neural network by minimizing the angular large margin cosine loss (LMCL) to learn protein embeddings clustered around the corresponding fold prototypes. The resulting hyperspherical embeddings are discriminative of the protein folds, as they are effective at recognizing the fold class by pairwise comparison. Moreover, the FoldHSpherePro method using an RF ensemble model yields high accuracy at the fold level, successfully bridging the performance gap between the different levels of evaluation.

Protein language model embeddings to predict fold types. Traditionally, information extracted from the multiple sequence alignment (MSA) has mainly been used as input source for different protein prediction tasks. In contrast, recent studies have shown that language models trained on millions of protein sequences are able to learn protein representations (protein-LM embeddings) containing relevant information about the protein and the contextual relationships between its amino acids. In Chapter 6 we analyze the performance of six different protein-LM embeddings on two fold-related tasks—pairwise fold recognition (PFR) and direct fold classification (DFC). We further train three neural network models to fine-tune the protein-LM embeddings and extract new fold-representative embeddings (PFR task) as well as perform the final fold classification (DFC task). These architectures include a multi-layer perceptron (MLP) working on protein-level embeddings, and the ResCNN-BGRU (RBG) and light-attention (LAT) models to process amino acid-level embeddings. The results indicate that the combination of transformer-based embeddings, particularly those obtained at amino acid-level, with the RBG and LAT fine-tuning models performs remarkably well in both tasks. Furthermore, ensemble strategies proposed for PFR and DFC provide a significant improvement over the current state-of-the-art results.

Extension to other tasks: protein function prediction. Although we do not present the published work on protein function prediction [110] explicitly as part of this dissertation, such work has contributed to the achievement of the Thesis objectives, and we summarize it here: Proteins can perform one or several functions, which are labeled according to terms in the Gene Ontology (GO) database. Therefore, the function prediction task can be seen as a multi-label classification problem that can be tackled using supervised machine learning algorithms. However, there is less labeled data than is needed to train complex models for this task. In [110] we argue that the use of deep protein representations from pre-trained protein language models (as in Chapter 6) could ease this problem. In particular, we analyze the performance of the protein-LM embeddings at predicting protein function, and compare against hand-crafted features extracted from the protein sequence (one-hot encoding of amino acids or k-mer counts) or the native structure (distance maps, secondary structure or backbone angles). We show that the protein-LM embeddings alone provide the best results using a simple classification model. Furthermore, the performance is not improved when combining these embeddings with distance map information in a graph convolutional network (GCN) model, although it does so when considering simple representations of the amino acids instead.

1.9 List of publications

All publications resulting from the work of this Thesis are listed below. The items are arranged in order of publication date, the latest first. For each journal, we also provide the Impact Factor (IF) computed by the Journal Citation Reports (JCR). Some journal papers have additionally been presented as an oral communication and/or a poster at international conferences. The symbol 🗎 indicates the journal papers supporting the compendium format in agreement with the Doctoral Programme.

🗎 1. **Amelia Villegas-Morcillo**, Angel M. Gomez, and Victoria Sanchez. An analysis of protein language model embeddings for fold prediction. *Briefings in Bioinformatics*, 23(3):bbac142, 2022.

   IF (JCR 2021): 13.994. Mathematical & Computational Biology. Rank 1/57 (D1).

🗎 2. **Amelia Villegas-Morcillo**, Victoria Sanchez, and Angel M. Gomez. FoldHSphere: deep hyperspherical embeddings for protein fold recognition. *BMC Bioinformatics*, 22(1):1-21, 2021.

   IF (JCR 2021): 3.328. Mathematical & Computational Biology. Rank 20/57 (Q2).

   Note: This work was also presented as a poster in the *29th International Conference on Intelligent Systems for Molecular Biology / 20th European Conference on Computational Biology (ISMB/ECCB)*, July 25-30, Virtual, 2021.

🗎 3. **Amelia Villegas-Morcillo**, Angel M. Gomez, Juan A. Morales-Cordovilla, and Victoria Sanchez. Protein fold recognition from sequences using convolutional and recurrent neural networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 18(6):2848-2854, 2021.

   IF (JCR 2021): 3.702. Mathematics, Interdisciplinary Applications. Rank 16/108 (Q1).

4. **Amelia Villegas-Morcillo**, Juan A. Morales-Cordovilla, Angel M. Gomez, and Victoria Sanchez. Improved protein residue–residue contact prediction using image denoising methods. In *Proceedings of 26th European Signal Processing Conference (EUSIPCO)*, pp. 1167-1171, September 3-7, Rome (Italy), 2018.

The following papers are not directly included in this dissertation, but were also developed during the doctoral period on related research topics:

5. **Amelia Villegas-Morcillo***, Stavros Makrodimitris*, Roeland C.H.J. van Ham, Angel M. Gomez, Victoria Sanchez, and Marcel J.T. Reinders. Unsupervised protein embeddings outperform hand-crafted sequence and structure features at predicting molecular function. *Bioinformatics*, 37(2):162-170, 2021.

   IF (JCR 2021): 6.931. Mathematical & Computational Biology. Rank 5/57 (D1).

   [*] Denotes equal contribution

   Note: This work was also presented as both oral communication and poster in the *28th International Conference on Intelligent Systems for Molecular Biology (ISMB)*, July 13-16, Virtual, 2020.

6. Francisco Gonzalez-Lopez, Juan A. Morales-Cordovilla, **Amelia Villegas-Morcillo**, Angel M. Gomez, and Victoria Sanchez. End-to-end prediction of protein-protein interaction based on embedding and recurrent neural networks. In *Proceedings of 2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pp. 2344-2350, December 3-6, Madrid (Spain), 2018.

References

[1] B. Alberts, D. Bray, K. Hopkin, et al. *Essential cell biology.* Garland Science, 2015.

[2] C. A. Wilson, J. Kreychman, and M. Gerstein. Assessing annotation transfer for genomics: quantifying the relations between protein sequence, structure and function through traditional and probabilistic scores. *Journal of Molecular Biology*, 297(1):233–249, 2000.

[3] M. I. Sadowski and D. T. Jones. The sequence–structure relationship and protein function prediction. *Current opinion in structural biology*, 19(3):357–362, 2009.

[4] M. Morey, A. Fernández-Marmiesse, D. Castiñeiras, et al. A glimpse into past, present, and future DNA sequencing. *Molecular Genetics and Metabolism*, 110(1-2):3–24, 2013.

[5] U. Consortium. UniProt: the universal protein knowledgebase in 2021. *Nucleic Acids Research*, 49(D1):D480–D489, 2021.

[6] B. Kuhlman and P. Bradley. Advances in protein structure prediction and design. *Nature Reviews Molecular Cell Biology*, 20(11):681–697, 2019.

[7] J. Moult. A decade of CASP: progress, bottlenecks and prognosis in protein structure prediction. *Current Opinion in Structural Biology*, 15(3):285–289, 2005.

[8] A. Kryshtafovych, T. Schwede, M. Topf, K. Fidelis, and J. Moult. Critical assessment of methods of protein structure prediction (CASP)–Round XIV. *Proteins: Structure, Function, and Bioinformatics*, 89(12):1607–1617, 2021.

[9] A. Fiser. Template-based protein structure modeling. In *Computational Biology*, pages 73–94. Humana Press, 2010.

[10] J. Lee, P. L. Freddolino, and Y. Zhang. Ab initio protein structure prediction. In *From Protein Structure to Function with Bioinformatics*, pages 3–35. Springer, 2017.

[11] C. Chothia and A. V. Finkelstein. The classification and origins of protein folding patterns. *Annual Review of Biochemistry*, 59(1):1007–1035, 1990.

[12] D. T. Jones, W. R. Taylor, and J. M. Thornton. A new approach to protein fold recognition. *Nature*, 358(6381):86, 1992.

[13] E. Lindahl and A. Elofsson. Identification of related proteins on family, superfamily and fold level. *Journal of Molecular Biology*, 295(3):613–625, 2000.

[14] R. D. Schaeffer and V. Daggett. Protein folds and protein folding. *Protein Engineering, Design & Selection*, 24(1-2):11–19, 2010.

[15] N. K. Fox, S. E. Brenner, and J.-M. Chandonia. SCOPe: Structural classification of proteins–extended, integrating SCOP and ASTRAL data and classification of new structures. *Nucleic Acids Research*, 42(D1):D304–D309, 2014.

[16] J.-M. Chandonia, L. Guan, S. Lin, et al. SCOPe: improvements to the structural classification of proteins–extended database to facilitate variant interpretation and machine learning. *Nucleic Acids Research*, 50(D1):D553–D559, 2022.

[17] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521:436–444, 2015.

[18] A. W. Senior, R. Evans, J. Jumper, et al. Improved protein structure prediction using potentials from deep learning. *Nature*, 577(7792):706–710, 2020.

[19] J. Jumper, R. Evans, A. Pritzel, et al. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, 2021.

[20] K. Illergård, D. H. Ardell, and A. Elofsson. Structure is three to ten times more conserved than sequence–A study of structural response in protein cores. *Proteins: Structure, Function, and Bioinformatics*, 77(3):499–508, 2009.

[21] OpenStax, Microbiology. OpenStax CNX. `https://openstax.org/books/microbiology/pages/7-4-proteins`. Mar 5, 2022.

[22] P. D. Sun, C. E. Foster, and J. C. Boyington. Overview of protein structural and functional folds. *Current Protocols in Protein Science*, 35(1):1–189, 2004.

[23] M. Levitt and C. Chothia. Structural patterns in globular proteins. *Nature*, 261(5561):552–558, 1976.

[24] D. Sehnal, S. Bittrich, M. Deshpande, et al. Mol* Viewer: modern web app for 3D visualization and analysis of large biomolecular structures. *Nucleic Acids Research*, 49(W1):W431–W437, 2021.

[25] K. Lin, A. C. May, and W. R. Taylor. Amino acid encoding schemes from protein structure alignments: Multi-dimensional vectors to describe residue types. *Journal of Theoretical Biology*, 216(3):361–365, 2002.

[26] K.-C. Chou. Prediction of protein cellular attributes using pseudo-amino acid composition. *Proteins: Structure, Function, and Bioinformatics*, 43(3):246–255, 2001.

[27] S. Kawashima and M. Kanehisa. AAindex: Amino Acid index database. *Nucleic Acids Research*, 28(1):374–374, 2000.

[28] M. O. Dayhoff, R. Schwartz, and B. Orcutt. A model of evolutionary change in proteins. *Atlas of Protein Sequence and Structure*, 5:345–352, 1978.

[29] S. Henikoff and J. G. Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences*, 89(22):10915–10919, 1992.

[30] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 1990.

[31] S. F. Altschul, T. L. Madden, A. A. Schäffer, et al. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402, 1997.

[32] S. R. Eddy. A new generation of homology search tools based on probabilistic inference. In *International Conference on Genome Informatics*, pages 205–211, 2009.

[33] M. Remmert, A. Biegert, A. Hauser, and J. Söding. HHblits: lightning-fast iterative protein sequence searching by HMM-HMM alignment. *Nature Methods*, 9(2):173–175, 2012.

[34] C. Leslie, E. Eskin, and W. S. Noble. The spectrum kernel: A string kernel for SVM protein classification. In *Biocomputing 2002*, pages 564–575. World Scientific, 2001.

[35] B. Liu, X. Wang, L. Lin, Q. Dong, and X. Wang. A discriminative method for protein remote homology detection and fold recognition combining Top-*n*-grams and latent semantic analysis. *BMC Bioinformatics*, 9(1):1–16, 2008.

[36] K.-C. Chou. Some remarks on protein attribute prediction and pseudo amino acid composition. *Journal of Theoretical Biology*, 273(1):236–247, 2011.

[37] Q. Dong, S. Zhou, and J. Guan. A new taxonomy-based protein fold recognition approach based on autocross-covariance transformation. *Bioinformatics*, 25(20):2655–2662, 2009.

[38] D. Ofer, N. Brandes, and M. Linial. The language of proteins: Nlp, machine learning & protein sequences. *Computational and Structural Biotechnology Journal*, 19:1750–1758, 2021.

[39] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

[40] E. Asgari and M. R. K. Mofrad. Continuous distributed representation of biological sequences for deep proteomics and genomics. *PLoS ONE*, 10(11):e0141287, 2015.

[41] M. E. Peters, M. Neumann, M. Iyyer, et al. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[42] M. Heinzinger, A. Elnaggar, Y. Wang, et al. Modeling aspects of the language of life through

transfer-learning protein sequences. *BMC Bioinformatics*, 20(1):1–17, 2019.

[43] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[44] A. Rives, J. Meier, T. Sercu, et al. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, 118(15):e2016239118, 2021.

[45] A. Elnaggar, M. Heinzinger, C. Dallago, et al. ProtTrans: Towards cracking the language of life's code through self-supervised deep learning and high performance computing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–16, 2021.

[46] H. M. Berman, J. Westbrook, Z. Feng, et al. The protein data bank. *Nucleic Acids Research*, 28(1):235–242, 2000.

[47] W. Kabsch and C. Sander. Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers: Original Research on Biomolecules*, 22(12):2577–2637, 1983.

[48] G. N. Ramachandran, C. Ramakrishnan, and V. Sasisekharan. Stereochemistry of polypeptide chain configurations. *Journal of Molecular Biology*, 7(1):95–99, 1963.

[49] C. Chothia. The nature of the accessible and buried surfaces in proteins. *Journal of molecular biology*, 105(1):1–12, 1976.

[50] L. Bartoli, E. Capriotti, P. Fariselli, P. L. Martelli, and R. Casadio. The pros and cons of predicting protein contact maps. *Methods in Molecular Biology*, 2007.

[51] J. Hanson, K. Paliwal, T. Litfin, Y. Yang, and Y. Zhou. Improving prediction of protein secondary structure, backbone angles, solvent accessibility and contact numbers by using predicted contact maps and an ensemble of recurrent and residual convolutional neural networks. *Bioinformatics*, 35(14):2403–2410, 2019.

[52] U. Göbel, C. Sander, R. Schneider, and A. Valencia. Correlated mutations and residue contacts in proteins. *Proteins: Structure, Function, and Bioinformatics*, 18(4):309–317, 1994.

[53] D. de Juan, F. Pazos, and A. Valencia. Emerging methods in protein co-evolution. *Nature Reviews Genetics*, 14(4):249–261, 2013.

[54] U. Consortium. The universal protein resource (UniProt). *Nucleic Acids Research*, 36:D190–D195, 2007.

[55] B. E. Suzek, Y. Wang, H. Huang, et al. UniRef clusters: a comprehensive and scalable alternative for improving sequence similarity searches. *Bioinformatics*, 31(6):926–932, 2015.

[56] M. Steinegger and J. Söding. MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nature Biotechnology*, 35(11):1026–1028, 2017.

[57] Q. Hou, F. Pucci, F. Pan, et al. Using metagenomic data to boost protein structure prediction and discovery. *Computational and Structural Biotechnology Journal*, 20:434–442, 2022.

[58] A. L. Mitchell, A. Almeida, M. Beracochea, et al. MGnify: the microbiome analysis resource in 2020. *Nucleic Acids Research*, 48(D1):D570–D578, 2020.

[59] M. Steinegger and J. Söding. Clustering huge protein sequence sets in linear time. *Nature Communications*, 9(1):1–8, 2018.

[60] M. Steinegger, M. Mirdita, and J. Söding. Protein-level assembly increases protein sequence recovery from metagenomic samples manyfold. *Nature Methods*, 16(7):603–606, 2019.

[61] S. K. Burley, C. Bhikadiya, C. Bi, et al. RCSB Protein Data Bank: powerful new tools for exploring 3d structures of biological macromolecules for basic and applied research and education in fundamental biology, biomedicine, biotechnology, bioengineering and energy sciences. *Nucleic Acids Research*, 49(D1):D437–D451, 2021.

[62] A. Waterhouse, M. Bertoni, S. Bienert, et al. SWISS-MODEL: homology modelling of protein structures and complexes. *Nucleic Acids Research*, 46(W1):W296–W303, 2018.

[63] M. Varadi, S. Anyango, M. Deshpande, et al. AlphaFold Protein Structure Database: massively expanding the structural coverage of protein-sequence space with high-accuracy models. *Nucleic Acids Research*, 2021.

[64] L. Holm and C. Sander. The FSSP database of structurally aligned protein fold families. *Nucleic Acids Research*, 22(17):3600, 1994.

[65] A. G. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia. SCOP: A structural classification of proteins database for the investigation of sequences and structures. *journal of Molecular Biology*, 247(4):536–540, 1995.

[66] C. A. Orengo, A. D. Michie, S. Jones, et al. CATH — a hierarchic classification of protein domain structures. *Structure*, 5(8):1093–1109, 1997.

[67] I. Sillitoe, N. Bordin, N. Dawson, et al. CATH: increased structural coverage of functional space. *Nucleic acids research*, 49(D1):D266–D273, 2021.

[68] H. Cheng, R. D. Schaeffer, Y. Liao, et al. ECOD: an evolutionary classification of protein domains. *PLoS Computational Biology*, 10(12):e1003926, 2014.

[69] C. Hadley and D. T. Jones. A systematic comparison of protein structure classifications: SCOP, CATH and FSSP. *Structure*, 7(9):1099–1112, 1999.

[70] R. D. Schaeffer, L. N. Kinch, J. Pei, K. E. Medvedev, and N. V. Grishin. Completeness and consistency in structural domain classifications. *ACS Omega*, 6(24):15698–15707, 2021.

[71] C. Chothia. One thousand families for the molecular biologist. *Nature*, 357(6379):543–544, 1992.

[72] S. Govindarajan, R. Recabarren, and R. A. Goldstein. Estimating the total number of protein folds. *Proteins: Structure, Function, and Bioinformatics*, 35(4):408–414, 1999.

[73] A. Grant, D. Lee, and C. Orengo. Progress towards mapping the universe of protein folds. *Genome Biology*, 5(5):1–9, 2004.

[74] M. S. Abual-Rub and R. Abdullah. A survey of protein fold recognition algorithms. *Journal of Computer Science*, 4(9):768–776, 2008.

[75] L. Wei and Q. Zou. Recent progress in machine learning-based methods for protein fold recognition. *International journal of Molecular Sciences*, 17(12):2118, 2016.

[76] J. Chen, M. Guo, X. Wang, and B. Liu. A comprehensive review and comparison of different computational methods for protein remote homology detection. *Briefings in Bioinformatics*, 19(2):231–244, 2018.

[77] K. Stapor, I. Roterman-Konieczna, and P. Fabian. Machine learning methods for the protein fold recognition problem. In *Machine Learning Paradigms*, volume 149, pages 101–127. Springer, 2019.

[78] J. Söding. Protein homology detection by HMM–HMM comparison. *Bioinformatics*, 21(7):951–960, 2005.

[79] J. Ma, S. Wang, Z. Wang, and J. Xu. MRFalign: Protein homology detection through alignment of Markov random fields. *PLoS Computational Biology*, 10(3):e1003500, 2014.

[80] J. Xu, M. Li, D. Kim, and Y. Xu. RAPTOR: optimal protein threading by linear programming. *Journal of Bioinformatics and Computational Biology*, 1(1):95–117, 2003.

[81] J. Peng and J. Xu. Boosting protein threading accuracy. In *Annual International Conference on Research in Computational Molecular Biology*, pages 31–45, 2009.

[82] Y. Yang, E. Faraggi, H. Zhao, and Y. Zhou. Improving protein fold recognition and template-based modeling by employing probabilistic-based matching between predicted one-dimensional structural properties of query and corresponding native properties of templates. *Bioinformatics*, 27(15):2076–2082, 2011.

[83] J. Ma, J. Peng, S. Wang, and J. Xu. A conditional neural fields model for protein threading. *Bioinformatics*, 28(12):i59–i66, 2012.

[84] J. A. Morales-Cordovilla, V. Sanchez, and M. Ratajczak. Protein alignment based on higher order conditional random fields for template-based modeling. *PLoS ONE*, 13(6):e0197912, 2018.

[85] D. W. A. Buchan and D. T. Jones. EigenTHREADER: analogous protein fold recognition by efficient contact map threading. *Bioinformatics*, 33(17):2684–2690, 2017.

[86] W. Zheng, Q. Wuyun, Y. Li, et al. Detecting distant-homology protein structures by aligning deep neural-network based contact maps. *PLoS Computational Biology*, 15(10):1–27, 2019.

[87] C. H. Q. Ding and I. Dubchak. Multi-class protein fold recognition using support vector machines and neural networks. *Bioinformatics*, 17(4):349–358, 2001.

[88] H.-B. Shen and K.-C. Chou. Ensemble classifier for protein fold pattern recognition. *Bioinformatics*, 22(14):1717–1722, 2006.

[89] J.-Y. Yang and X. Chen. Improving taxonomy-based protein fold recognition by using global and local features. *Proteins: Structure, Function, and Bioinformatics*, 79(7):2053–2064, 2011.

[90] J. Lyons, A. Dehzangi, R. Heffernan, et al. Advancing the accuracy of protein fold recognition by utilizing profiles from hidden Markov models. *IEEE Transactions on Nanobioscience*, 14(7):761–772, 2015.

[91] D. Chen, X. Tian, B. Zhou, and J. Gao. ProFold: Protein fold classification with additional structural features and a novel ensemble classifier. *BioMed Research International*, 2016:1–10, 2016.

[92] J. Xia, Z. Peng, D. Qi, H. Mu, and J. Yang. An ensemble approach to protein fold classification by integration of template-based assignment and support vector machine classifier. *Bioinformatics*, 33(6):863–870, 2016.

[93] W. Ibrahim and M. S. Abadeh. Protein fold recognition using deep kernelized extreme learning machine and linear discriminant analysis. *Neural Computing and Applications*, 31(8):4201–4214, 2019.

[94] S. Bankapur and N. Patil. An enhanced protein fold recognition for low similarity datasets using convolutional and skip-gram features with deep neural network. *IEEE Transactions on NanoBioscience*, 20(1):42–49, 2020.

[95] W. Elhefnawy, M. Li, J. Wang, and Y. Li. DeepFrag-k: a fragment-based deep learning approach for protein fold recognition. *BMC Bioinformatics*, 21(6):1–12, 2020.

[96] J. Cheng and P. Baldi. A machine learning information retrieval approach to protein fold recognition. *Bioinformatics*, 22(12):1456–1463, 2006.

[97] T. Jo and J. Cheng. Improving protein fold recognition by random forest. *BMC Bioinformatics*, 15(11):S14, 2014.

[98] T. Jo, J. Hou, J. Eickholt, and J. Cheng. Improving protein fold recognition by deep learning networks. *Scientific Reports*, 5:17573, 2015.

[99] K. Yan, X. Fang, Y. Xu, and B. Liu. Protein fold recognition based on multi-view modeling. *Bioinformatics*, 35(17):2982–2990, 2019.

[100] K. Yan, J. W. an Yong Xu, and B. Liu. Protein fold recognition based on auto-weighted multi-view graph embedding learning model. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2020.

[101] K. Yan, J. Wen, Y. Xu, and B. Liu. MLDH-Fold: Protein fold recognition based on multi-view low-rank modeling. *Neurocomputing*, 421:127–139, 2021.

[102] J. Hou, B. Adhikari, and J. Cheng. DeepSF: deep convolutional neural network for mapping protein sequences to folds. *Bioinformatics*, 34(8):1295–1303, 2018.

[103] J. Zhu, H. Zhang, S. C. Li, et al. Improving protein fold recognition by extracting fold-specific features from predicted residue–residue contaacts. *Bioinformatics*, 33(23):3749–3757, 2017.

[104] B. Liu, C.-C. Li, and K. Yan. DeepSVM-fold: protein fold recognition by combining support vector machines and pairwise sequence similarity scores generated by deep learning networks. *Briefings in Bioinformatics*, 2019.

[105] C.-C. Li and B. Liu. MotifCNN-fold: protein fold recognition based on fold-specific features extracted by motif-based convolutional neural networks. *Briefings in Bioinformatics*, 21(6):2133–2141, 2020.

[106] Y. Pang and B. Liu. SelfAT-Fold: protein fold recognition based on residue-based and motif-based self-attention networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2020.

[107] Y. Liu, Y.-H. Zhu, X. Song, J. Song, and D.-J. Yu. Why can deep convolutional neural networks improve protein fold recognition? a visual explanation by interpretation. *Briefings in Bioinformatics*, 2021.

[108] Y. Liu, K. Han, Y.-H. Zhu, et al. Improving protein fold recognition using triplet network and ensemble deep learning. *Briefings in Bioinformatics*, 22(6):bbab248, 2021.

[109] The UniProt Consortium. UniProt: the universal protein knowledgebase. *Nucleic Acids Research*, 45(D1):D158–D169, 2017.

[110] A. Villegas-Morcillo, S. Makrodimitris, R. C. H. J. van Ham, et al. Unsupervised protein embeddings outperform hand-crafted sequence and structure features at predicting molecular function. *Bioinformatics*, 37(2):162–170, 2021.

II

Results

2

Improved protein residue–residue contact prediction using image denoising methods

Amelia Villegas-Morcillo, Juan A. Morales-Cordovilla,
Angel M. Gomez, and Victoria Sanchez

This chapter has been published as follows: A. Villegas-Morcillo, J.A. Morales-Cordovilla, A.M. Gomez, and V. Sanchez. Improved protein residue--residue contact prediction using image denoising methods. In *Proceedings of 26th European Signal Processing Conference (EUSIPCO)*, pp. 1167-1171, September 3-7, Rome (Italy), 2018. DOI: 10.23919/EUSIPCO.2018.8553519

Abstract

A protein contact map is a simplified matrix representation of the protein structure, where the spatial proximity of two amino acid residues is reflected. Although the accurate prediction of protein inter-residue contacts from the amino acid sequence is an open problem, considerable progress has been made in recent years. This progress has been driven by the development of contact predictors that identify the co-evolutionary events occurring in a protein multiple sequence alignment (MSA). However, it has been shown that these methods introduce Gaussian noise in the estimated contact map, making its reduction necessary. In this paper, we propose the use of two different Gaussian denoising approximations in order to enhance the protein contact estimation. These approaches are based on (i) sparse representations over learned dictionaries, and (ii) deep residual convolutional neural networks. The results highlight that the residual learning strategy allows a better reconstruction of the contact map, thus improving contact predictions.

Index terms*: Protein Contact Map, Evolutionary Coupling, Image Denoising, Sparse Representations, Dictionary Learning, Deep Convolutional Neural Networks, Residual Learning*

2.1 Introduction

Proteins are life essential macromolecules composed of long chains that contain 20 different types of amino acid residues. Protein folding is one of the most challenging problems found in bioinformatics, which still remains unsolved. It refers to the spatial arrangement of the amino acid sequence in three-dimensional (3D) space, which is closely related to the biological function performed by the protein.

Several experimental techniques, such as X-ray crystallography, nuclear magnetic resonance spectroscopy and electron microscopy, are widely employed to determine protein structure. However, these methods are not cost-efficient compared to fast deoxyribonucleic acid (DNA) sequencing processes, which are continuously generating a huge quantity of amino acid sequences with unsolved structure [1]. This fact has led to a great development of computational methods that attempt to predict the protein structure from its amino acid sequence. These computational methods can be divided into template-based and template-free modeling. Template-free modeling methods, also known as *de novo* or *ab initio* prediction methods, are suitable for proteins without structural homologs [2]. The complexity of this approach is usually reduced by using a two-dimensional (2D) representation known as protein contact map. Two amino acid residues within a protein are said to form a contact when they share a spatial proximity that is sufficient for a molecular interaction to occur. Thus, the true contact map for a protein with L residues consists of an $L \times L$ symmetrical matrix, where each element specifies whether two inter-residues are in contact under a certain Euclidean distance threshold. By contrast, the matrix elements within an estimated contact map indicate the contact probability of two residues. Recent research has shown that even a low proportion of correctly-predicted contacts can be enough for accurately modeling the protein structure [3].

Current state-of-the-art contact predictors can be classified into two main categories, based on evolutionary coupling analysis (EC) and supervised machine learning (ML), respectively [4]. In the first group we can find methods such as PSICOV [5] and CCMpred [6], which aim to identify co-evolved residues from the protein multiple sequence alignment (MSA). Co-evolution is related to correlated mutations occurring in proteins. This means that if one of the two residues in contact has mutated during evolution, its counterpart has to change as well in order for the 3D structure to remain stable [7]. On the other hand, supervised machine learning-based predictors use a set of input features derived from the MSA to estimate the contact map, including position-specific scoring matrices (PSSM), secondary structure predictions or solvent accessibility information [8]. ML-based methods learn from these features using several algorithms, such as support vector machines, neural networks and random forests.

Generally, the performance of evolutionary coupling methods relies on having a sufficient number of effective homologous sequences to construct the protein MSA. In order to improve the precision of estimated contacts, methods such as MetaPSICOV [9] wisely combine the EC-based approach with machine learning. Additionally, the EC-based methods were designed to reduce the number of misleading indirect coupling pairs, i.e. those that show a high degree of correlated mutation without being close in the 3D space. This is caused by the transitive relationship between residue pairs, resulting in a transitive noise within the contact map [4].

Besides this transitive noise, it has been recently shown that Gaussian noise also exists in contact maps derived from evolutionary couplings. To reduce its effect, the R_2C system [10] included a post-processing noise filter, yielding higher contact precision. This denoising step was implemented using the non-local means with optimal weights algorithm specified in [11]. In this work, we aim to improve the estimated protein contact map by further suppressing the inherent Gaussian noise. To accomplish that, we have followed two approaches related to image processing: sparse representations over learned dictionaries, and deep residual convolutional neural networks.

First, we consider the K-SVD algorithm [12], widely used to design overcomplete dictionaries for sparse representations of signals. This method has proven to be successful for image super-resolution, compression and denoising [13]. Sparse representations exploit image redundancies, trying to find the minimum number of dictionary entries needed to properly reconstruct the image. Although finding the sparsest solution is an NP-hard problem, approximate solutions can be efficiently found by using pursuit algorithms.

Then, we explore a deep neural network approach, as there is a growing interest in deep learning for proteomics (see [14, 15]). In particular, we analyze a deep convolutional neural network (DCNN) specially designed for image denoising [16]. Convolutional neural networks have the ability to extract structural motifs, which makes them suitable for image processing. Moreover, it is possible to reach a high level of abstraction with very deep architectures and residual learning.

The rest of the paper is structured as follows. In Section 2.2, we explain the contact map estimation process. Section 2.3 describes the proposed methods for contact map denoising. In Section 2.4, we present the main results obtained from the experiments. Finally, Section 2.5 summarizes the conclusions derived from this research and its possible extensions.

2.2 EC-based contact map estimation

The overall approach proposed in this work is outlined in Figure 2.1. Our objective is to improve the precision of estimated contact maps. True contacts are defined in terms of spatial molecular interaction. The residue spatial proximity is measured as the Euclidean distance between their beta carbon (C_β) atoms (C_α in Glycine). This results in a distance map, which is converted to a binary contact map (a symmetrical matrix) by considering a threshold (in this work we used a distance less than 8 Å). Thus, the extraction of true contacts requires knowledge of the protein's 3D experimental structure.

In this work, we have used one of the most popular EC-based methods, CCMpred [6], to estimate the contact map. CCMpred implements the approach based on Markov

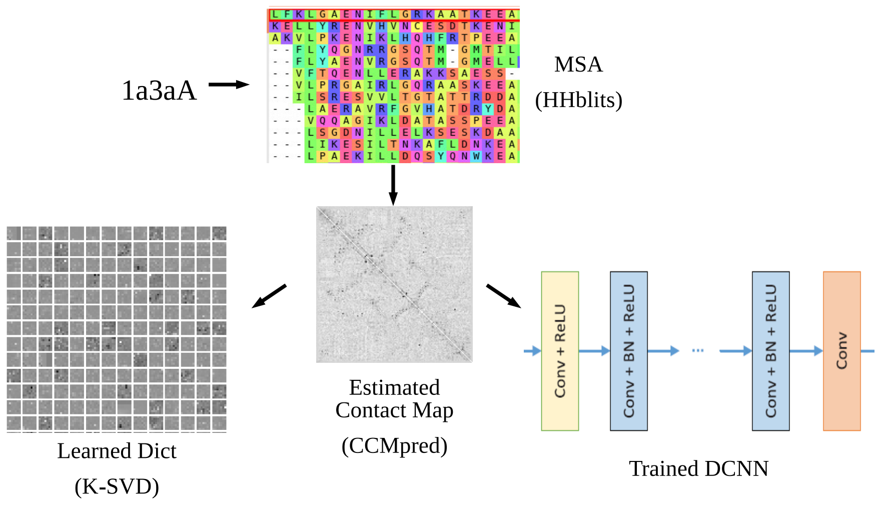


Figure 2.1: Description of the overall protein contact map denoising procedures.

random field pseudo-likelihood maximization. Its election was motivated by the high contact precision reached in comparison to other EC-based methods [4]. In addition, it can be run on a GPU card. The first step of this method is to build a multiple sequence alignment (MSA) for each query protein, using the HHblits software [17] to search for homologous sequences in the `uniprot20_2016_02` database [18]. Then, the MSA is used as input to CCMpred, which estimates correlated mutations between each pair of residues, generating a symmetrical contact likelihood matrix as a result. All parameters in both tools were set to default values, except for the number of HHblits iterations that was increased to five.

2.3 Proposed methods for contact map denoising

The estimated contact map derived from CCMpred can be viewed as an image **X**, which is a noisy version of the true contact map Y corrupted by an additive Gaussian noise **Z**:

$$\mathbf{X} = \mathrm{Y} + \mathbf{Z}. \tag{2.1}$$

As indicated in the introduction, in order to address the contact map denoising problem, we propose the use of two noise removal techniques: sparse representations over learned dictionaries and deep residual convolutional neural networks.

2.3.1 Dictionary learning for sparse representations

The overall K-SVD method for image denoising [13] can be seen as a three-stage iterative process: (i) sparse coding, (ii) dictionary updating, and (iii) final averaging.

In the sparse coding stage, a fixed dictionary **D** with K entries (also known as atoms) and the noisy image **X** divided into N overlapping patches of size $\sqrt{n} \times \sqrt{n}$ are considered. Thus, the sparse coding vector $\boldsymbol{\alpha}_i$ associated to the i-th patch ($i = 1, 2, ..., N$) can be computed by solving the following optimization problem:

$$\hat{\boldsymbol{\alpha}}_i = \arg\min_{\boldsymbol{\alpha}_i} \|\boldsymbol{\alpha}_i\|_0 \ \text{ subject to } \ \|\mathbf{x}_i - \mathbf{D}\boldsymbol{\alpha}_i\|_2^2 \leq \epsilon^2 \tag{2.2}$$

where $\mathbf{x}_i$ is the i-th flattened patch (column vector) of size n, **D** is the dictionary of size $n \times K$, ϵ is the error tolerance that depends on the noisy image SNR, and $\|\cdot\|_p$ represents the l^p-norm. This problem can be addressed with the greedy orthogonal matching pursuit (OMP) algorithm [19], which takes the closest atom of the dictionary at a time to update the sparse coding vector.

Once the N sparse coding vectors are computed, a new version of the patches $\hat{\mathbf{x}}_i = \mathbf{D}\hat{\boldsymbol{\alpha}}_i$ is obtained to update the dictionary (the initial dictionary $\mathbf{D}_0$ is usually built from the overcomplete discrete cosine transform (DCT) or with randomly taken patches from the image). For each atom k in the dictionary, the K-SVD algorithm [12] locates the set of patches that use this atom ($\hat{\boldsymbol{\alpha}}_i(k) \neq 0$), to perform a singular value decomposition (SVD) operation on the representation errors $\mathbf{e}_i^k$, which are calculated as

$$\mathbf{e}_i^k = \mathbf{x}_i - \sum_{l \neq k} \mathbf{d}_l \hat{\boldsymbol{\alpha}}_i(l) \tag{2.3}$$

where $\mathbf{d}_l$ are the flattened dictionary atoms. These two stages are repeated a fixed number of iterations, resulting in a learned dictionary $\mathbf{D}$ and sparse coding vectors $\hat{\boldsymbol{\alpha}}_i$ associated to each patch. Finally, the reconstructed image $\hat{\mathbf{X}}$ is computed as an average of the N overlapping denoised patches $\hat{\mathbf{x}}_i$ with the original noisy image $\mathbf{X}$:

$$\hat{\mathbf{X}} = \frac{\lambda \mathbf{X} + \sum_i \mathbf{R}_i^T \mathbf{D} \hat{\boldsymbol{\alpha}}_i}{\lambda \mathbf{I} + \sum_i \mathbf{R}_i^T \mathbf{R}_i} \tag{2.4}$$

where $\mathbf{R}_i$ is a mask matrix that extracts the i-th patch from the image ($\mathbf{x}_i = \mathbf{R}_i \mathbf{X}$) and λ is an hyper-parameter dependent on the noisy image SNR.

In our study, we evaluated the K-SVD method to denoise our contact map images. To do so, we adopted the efficient implementation available in [20], which is based on approximated K-SVD and batch-OMP. In order to exploit the symmetry in contact maps, we trained the dictionary only with overlapping patches from the upper triangular part of the image. We only considered this reconstructed part to evaluate the denoised contacts.

2.3.2 DCNN Training with residual learning

The DCNN model implements a very deep architecture with 2D convolutional layers. This architecture consists of d layers that apply 64 convolution filters of size 3×3. As our contact map images are in gray-scale, we have one channel at the input layer and one filter at the output layer. The output image size is kept by applying zero-padding before convolutions. At each layer (except for the output one), rectified linear units (ReLU) non-linearities are used as activation function. In addition, if we want that all the pixels within a patch (of size $\sqrt{n} \times \sqrt{n}$) contribute to each output after the cascade of convolutional layers (with $\sqrt{n_f} \times \sqrt{n_f}$ filter size), network depth d must be set to a minimum value given as:

$$d = \left(\sqrt{n} - 1\right) / \left(\sqrt{n_f} - 1\right). \tag{2.5}$$

On the other hand, the residual learning strategy aims to extract the differences between the network inputs and outputs. In other words, the DCNN trains a residual mapping of the input image regarding the noise $\mathcal{R}(\mathbf{X}) \approx \mathbf{Z}$ and then calculates the reconstructed image as $\hat{\mathbf{X}} = \mathbf{X} - \mathcal{R}(\mathbf{X})$. Thus, the loss function $l(\Theta)$ can be formulated as:

$$l(\Theta) = \frac{1}{2B} \sum_{i=1}^{B} \|\mathcal{R}(\mathbf{x}_i; \Theta) - (\mathbf{x}_i - \mathbf{y}_i)\|_F^2 \tag{2.6}$$

where B is the number of noisy/clean training patches in a batch, and $\mathbf{\Theta}$ are the trainable parameters. This is equivalent to minimizing the mean squared error between each clean patch $\mathbf{y}_i$ and the reconstructed patch $\hat{\mathbf{x}}_i$. Besides residual learning, an optimization strategy based on mini-batches with Adam algorithm for gradient-descend and batch-normalization was adopted as in [16].

Unlike the K-SVD method, this approximation is able to train the model without knowing the noisy image SNR. We adopted the TensorFlow GPU implementation of [16] to train a noise level-independent model with our contact map images. As the output matrix is not assured to be symmetrical, we computed the mean value of the two triangular parts (upper and lower) to evaluate the contacts.

2.4 Experimental framework and results

In this section, we first describe the protein datasets used in the experiments. Next, we specify the criteria followed for contact evaluation. Finally, we present the results obtained from the experiments carried out and we discuss them.

2.4.1 Training and test datasets

During the evaluation stage, we considered three test datasets. The first one comprises 150 Pfam proteins from PSICOV [5], which have been widely used for contact map evaluations. The other two sets were obtained from two CASP (Critical Assessment of Structure Prediction) competitions. In particular, 116 protein domains from CASP10 (2012) and 103 from CASP11 (2014) were used as in [10]. Moreover, none of the test sequences contained less than 50 residues or more than 519 residues.

For DCNN training purposes, a set with 3760 protein domains [15] was taken. We further modified this set in order to guarantee its independence from the testing sets. To do so, we run the CD-HIT [21] tool to exclude those proteins that shared more than 40% of sequence identity with any protein in the test set. A total number of 3427 training proteins were kept, with sequence lengths varying from 28 to 597 residues. From these, we randomly selected 300 proteins to validate the DCNN and we used the remaining ones in the training stage.

2.4.2 Evaluation criteria

Contact evaluation was accomplished by comparing the estimated contact map with the true contact map extracted from the protein data bank (PDB) [1]. The widely used evaluation strategy [4] is based on dividing contacts into three groups. These groups are dependent on the amino acid sequence positions, i.e. short-range (residue separation between 6 and 11 positions), medium-range (from 12 to 23) or long-range (greater than 23). For each group, prediction accuracy is obtained by computing the precision of the L/k contacts with the highest probability, where L is the sequence length and $k = \{10, 5, 2, 1\}$ controls the ratio of contacts to be evaluated.

2.4.3 K-SVD Parameter setting

In order to evaluate the K-SVD method, we followed a dictionary learning strategy based on training one dictionary $\mathbf{D}$ for each noisy contact map. Thus, we first divided the upper triangular part of the zero-padded image in overlapping patches. As the contact map has size $L \times L$ and protein lengths are variable, the number of training patches also varies in each case. To initialize each dictionary, we used the overcomplete DCT with K atoms. Then, we executed 10 iterations of the K-SVD algorithm, using OMP for sparse coding with maximum error $\epsilon = 1.15\sqrt{n}\sigma$, where the noise standard deviation σ was estimated following the fast approach cited in [10]. Finally, the upper triangular matrix was reconstructed using (2.4), with $\lambda = 0.5/\sigma$. We chose low values for λ in order to give less importance to the original noisy contact map than to the denoised one.

For K-SVD parameter selection purposes, we conducted initial experiments on the 150 Pfam proteins, testing two patch sizes (5×5 and 7×7) and two different number of atoms K (625 and 900). The results were compared with the baseline, i.e. contact precision values from CCMpred estimations. All combinations of patch/dictionary sizes yielded similar contact results. However, slightly better results were achieved when considering 5×5 patches and $K = 900$ atoms, so we have selected this configuration for a complete evaluation on the rest of datasets.

2.4.4 DCNN Parameter Setting

The denoising task with DCNN was carried out by adopting the architecture described in Section 2.3.2. In order to apply residual learning, we extracted the true and estimated contact maps from the 3127 training proteins mentioned in Section 2.4.1. All training contact maps were divided into patches of size 35×35 (with a stride of 10 pixels), which provided us with more than a million noisy/clean samples to train the DCNN. Unlike the previous method, we extracted patches from the entire image and not only from the upper triangle (the denoising procedure was also applied on the entire image). According to (2.5), the depth of the network was set to $d = 17$. Training contact maps that are smaller than the patch size (35×35) were filled with zeros at the input and multiplied by a binary mask before computing the loss function in (2.6).

Thereafter, we trained the DCNN for 50 epochs in a mini-batch mode. At each epoch, we randomly scrambled all patches and iteratively fed the network with subsets (batches) of 256 noisy/clean samples. When an epoch was finished, we used the current model to denoise and evaluate the 300 validation contact maps. Figure 2.2 shows the precision curves for long-range contact predictions on the validation set. As can be observed, the

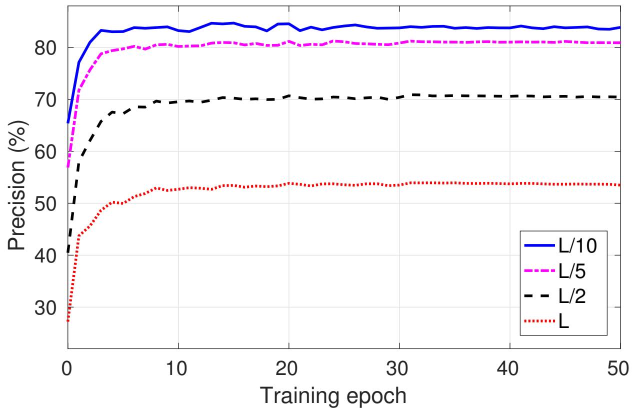


Figure 2.2: Long-range contact prediction performance on 300 validation protein domains along DCNN training epochs.

evaluation values in epoch 0 (before the first gradient-descend step) are similar to those from the baseline (CCMpred), which can be explained by the residual learning strategy. Although the precision curves show a considerable performance increase in the very first epochs, saturation appears rapidly. In this study, we followed an early stopping strategy to prevent DCNN overfitting. Thus, we selected the model at epoch 31 to denoise the three test datasets, as it yields the best evaluation results on the validation dataset.

2.4.5 Comparison of methods

Finally, we present the contact precision results for the three test datasets (i.e. 150 Pfam proteins, 116 CASP10 protein domains and 103 CASP11 protein domains), achieved by the baseline (CCMpred) and the proposed post-processing denoising methods K-SVD and DCNN. These results have been also compared to the R_2C noise filter [10]. In this case, we used the implementation in [11], setting the main parameters to 9×9 for neighboring size and 13×13 for patch size, as in [10].

Table 2.1 summarizes all the prediction results for short-, medium- and long-range contacts. For all contact ranges in each test dataset, we marked in red those results that are worse than the baseline and, in boldface, the best results. As we can see, both the R_2C filter and the K-SVD method do not perform very well in some cases (specially for short-range contacts). The proposed K-SVD method performs slightly better than the R_2C filter, whereas the highest overall prediction enhancement is achieved with the residual DCNN approach.

This is due to the fact that, both the R_2C filter and the K-SVD method only consider the estimated contact map itself, while the residual DCNN has access to true contact maps during the training phase. Therefore, the first two approaches have the ability to reduce

Table 2.1: Contact precision values for short-, medium- and long-range for the evaluated methods on the test datasets.

Test set	Method	Short-range				Medium-range				Long-range			
		L/10	*L/5*	*L/2*	*L*	*L/10*	*L/5*	*L/2*	*L*	*L/10*	*L/5*	*L/2*	*L*
150 Pfam proteins	Baseline	56.1	40.2	23.0	15.5	64.2	49.8	29.0	18.4	78.1	71.0	50.5	33.7
	R_2C filter	51.3	36.8	22.0	15.2	64.5	49.4	29.9	19.2	78.9	70.1	51.2	35.8
	K-SVD OMP	55.7	39.7	24.1	16.3	67.2	53.0	32.2	20.8	79.8	73.0	54.0	38.4
	DCNN	**77.6**	**65.2**	**40.9**	**25.0**	**80.5**	**71.0**	**48.3**	**30.3**	**89.6**	**85.3**	**72.1**	**54.5**
116 CASP10 proteins	Baseline	41.5	31.2	19.4	13.5	53.1	41.9	26.3	18.1	53.7	47.8	34.4	23.1
	R_2C filter	41.3	30.6	19.8	14.3	54.0	42.5	27.8	19.1	57.1	51.1	37.3	26.4
	K-SVD OMP	43.1	32.2	20.6	14.7	55.6	43.8	29.8	20.4	56.3	49.7	38.2	27.1
	DCNN	**58.4**	**48.5**	**31.9**	**20.7**	**65.8**	**58.1**	**43.0**	**29.8**	**67.3**	**63.2**	**50.6**	**37.6**
103 CASP11 proteins	Baseline	32.9	23.9	15.3	11.3	38.0	28.5	17.8	12.5	47.4	40.2	28.9	20.1
	R_2C filter	31.4	22.8	14.6	11.5	39.7	29.6	19.2	13.8	50.0	42.5	30.7	22.3
	K-SVD OMP	32.8	23.4	15.4	12.1	40.4	31.7	20.6	14.3	48.5	42.4	31.4	22.7
	DCNN	**48.1**	**38.8**	**26.1**	**17.6**	**53.7**	**46.6**	**31.9**	**21.2**	**56.5**	**53.2**	**43.0**	**32.6**

Gaussian noise but at the cost of losing some contacts due to a smoothing effect. On the contrary, the residual DCNN shows noise reduction potential along with the ability to recover some missing contacts not present in CCMpred estimations. Moreover, one of the main DCNN advantages is that we can train a blind model without knowing the noise standard deviation, so we can avoid using poor estimations of it as in the R_2C filter or the K-SVD method. As an example, Figure 2.3 shows the resulting contact maps for protein domain T0682-D1 (included in the CASP10 dataset), produced by the three denoising methods in comparison to the true and baseline contact maps. Once again, we can see that the DCNN provides the best denoised contact map.

2.5 Conclusions

In this paper, we have evaluated two alternative Gaussian denoising methods for the protein contact map prediction problem. The first one is based on sparse representations over learned dictionaries for image denoising, using K-SVD with the OMP algorithm. In the second one, we train a deep convolutional neural network (DCNN) with residual learning for image noise removal. The experimental results show that better contact precision values can be obtained by these noise reduction techniques. We particularly found that the residual DCNN strategy performs the best in the denoising task, allowing the acquisition of more true contacts. Future work will explore other residual DCNN architectures that can exploit the sparse nature of contact maps, along with the study of how the improved contacts enhance the prediction of the 3D protein structure.

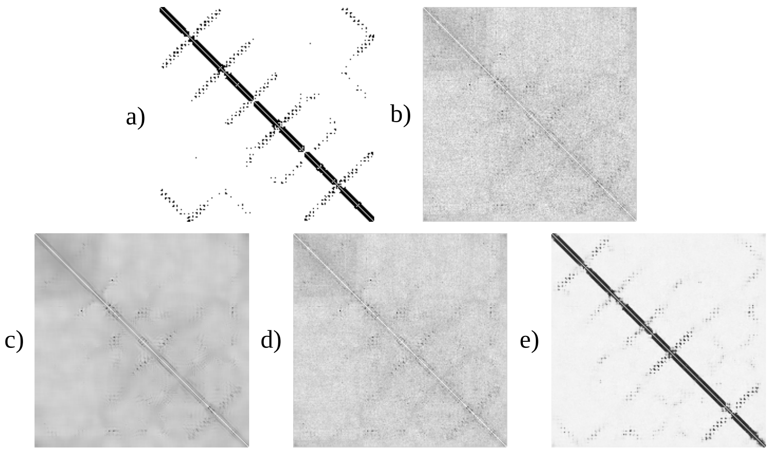


Figure 2.3: Contact maps obtained for protein domain T0682-D1 in the CASP10 dataset. **a)** True from PDB. **b)** Estimated from CCMpred. **c)** Denoised using R_2C. **d)** Denoised using K-SVD OMP. **e)** Denoised using DCNN.

Acknowledgments

This work has been supported by the Spanish MINECO/FEDER Project TEC2016-80141-P and the associated FPI grant BES-2017-079792. We also acknowledge the support of NVIDIA Corporation with the donation of a Titan X GPU.

References

[1] H. M. Berman, J. Westbrook, Z. Feng, et al. The protein data bank. *Nucleic Acids Research*, 28(1):235–242, 2000.

[2] S. H. P. de Oliveira, J. Shi, and C. M. Deane. Comparing co-evolution methods and their application to template-free protein structure prediction. *Bioinformatics*, 33(3):373–381, 2017.

[3] D. E. Kim, F. DiMaio, R. Y.-R. Wang, Y. Song, and D. Baker. One contact for every twelve residues allows robust and accurate topology-level protein structure modeling. *Proteins*, 82(0 2):208–218, 2014.

[4] Q. Wuyun, W. Zheng, Z. Peng, and J. Yang. A large-scale comparative assessment of methods for residue–residue contact prediction. *Briefings in Bioinformatics*, pages 1–12, 2016.

[5] D. T. Jones, D. W. A. Buchan, D. Cozzetto, and M. Pontil. PSICOV: precise structural contact prediction using sparse inverse covariance estimation on large multiple sequence alignments. *Bioinformatics*, 28(2):184–190, 2012.

[6] S. Seemayer, M. Gruber, and J. Söding. CCMpred—fast and precise prediction of protein residue–residue contacts from correlated mutations. *Bioinformatics*, 30(21):3128–3130, 2014.

[7] U. Göbel, C. Sander, R. Schneider, and A. Valencia. Correlated mutations and residue contacts

in proteins. *Proteins: Structure, Function, and Bioinformatics*, 18(4):309–317, 1994.

[8] J. Eickholt and J. Cheng. Predicting protein residue–residue contacts using deep networks and boosting. *Bioinformatics*, 28(23):3066–3072, 2012.

[9] D. T. Jones, T. Singh, T. Kosciolek, and S. Tetchner. MetaPSICOV: combining coevolution methods for accurate prediction of contacts and long range hydrogen bonding in proteins. *Bioinformatics*, 31(7):999–1006, 2015.

[10] J. Yang, Q.-Y. Jin, B. Zhang, and H.-B. Shen. R_2C: improving *ab initio* residue contact map prediction using dynamic fusion strategy and Gaussian noise filter. *Bioinformatics*, 32(16):2435–2443, 2016.

[11] Q. Jin, I. Grama, C. Kervrann, and Q. Liu. Non-local means and optimal weights for noise removal. *SIAM Journal on Imaging Sciences*, 10(4):1878–1920, 2017.

[12] M. Aharon, M. Elad, and A. Bruckstein. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, 2006.

[13] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image Processing*, 15(12):3736–3745, 2006.

[14] S. Wang, S. Sun, Z. Li, R. Zhang, and J. Xu. Accurate de novo prediction of protein contact map by ultra-deep learning model. *PLoS Computational Biology*, 13(1):e1005324, 2017.

[15] J. Zhu, H. Zhang, S. C. Li, et al. Improving protein fold recognition by extracting fold-specific features from predicted residue–residue contaacts. *Bioinformatics*, 33(23):3749–3757, 2017.

[16] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017.

[17] M. Remmert, A. Biegert, A. Hauser, and J. Söding. HHblits: lightning-fast iterative protein sequence searching by HMM-HMM alignment. *Nature Methods*, 9(2):173–175, 2012.

[18] The UniProt Consortium. UniProt: the universal protein knowledgebase. *Nucleic Acids Research*, 45(D1):D158–D169, 2017.

[19] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad. Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. *27-th Asilomar Conference on Signals, Systems and Computers*, 1:40–44, 1993.

[20] R. Rubinstein, M. Zibulevsky, and M. Elad. Efficient implementation of the K-SVD algorithm using Batch Orthogonal Matching Pursuit. *Cs Technion*, 40(8):1–15, 2008.

[21] W. Li and A. Godzik. Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*, 22(13):1658–1659, 2006.

3

On the impact of using estimated, enhanced, or native contact and categorical maps on protein fold recognition performance

3.1 Introduction

In this chapter we take a first step towards the design of protein fold recognition methods that advance the state of the art. In particular, we consider image representations of the protein in the form of contact maps. The approach we follow here is the one proposed for the deep learning-based DeepFR method [1]. As input source, DeepFR takes estimated contact maps from the evolutionary coupling analysis-based method CCMpred [2]. These image representations are fed into a deep convolutional neural network (DCNN) that is trained to classify them into the whole set of fold classes defined in the SCOP database (more than 1000) [3, 4]. Instead of directly evaluating the predicted folds, the DeepFR model is used to extract a fold-specific vector for each protein domain (*embedding*). Then, fold recognition is performed for each pair of query-template domains using these vectors and a comparison metric based on cosine similarity. When evaluating the fold recognition performance, three levels of difficulty are usually considered following the hierarchical classification from SCOP: *fold, superfamily*, and *family*. The hardest one is the *fold* level, where protein domains significantly differ in their amino acid sequences, and therefore the fold class cannot be easily inferred by comparison with other sequences (homology modeling).

Our objective here is to analyze to what extent contact maps alone are useful for protein fold recognition, following the DeepFR approach. To do so, we study the impact of using different input image representations on the fold-specific vectors learned by the model, which we refer to as *Fold-DCNN* here. We compare the performance of estimated contact maps (using CCMpred) to native contact maps obtained directly from the PDB structure. We also propose to use categorical maps as proxies of the full distance maps and include them in the comparison. As a middle ground between estimated and native maps, we obtain enhanced versions of the contact and categorical maps. To this end, we train and evaluate an adapted version of the fully-convolutional network (FCN) model from our previous work [5], called *CMap-FCN* here, which also uses estimated contact maps from CCMpred as inputs.

3.2 Materials and methods

3.2.1 Image representations of the protein

Native maps

To obtain a native (ground-truth) image representations of the protein (Figure 3.1a) we first pulled their experimental 3D structures from the protein data bank (PDB). Using the 3D coordinates, we then computed the *distance map* using beta-carbon atoms [6, 7]. From this, we obtained two different representations: the *contact map* [8, 9], a binary matrix

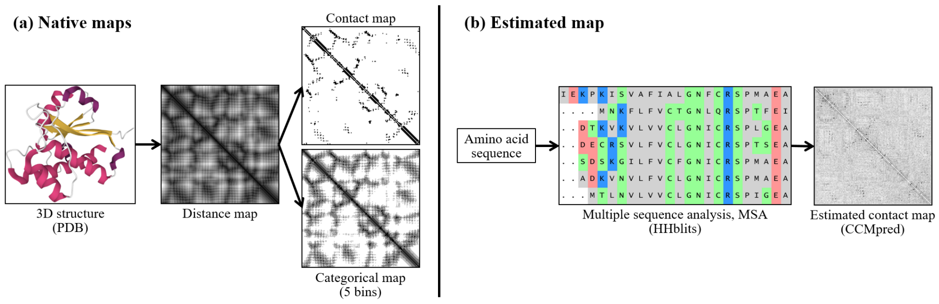

(a) Native maps
3D structure (PDB)
Distance map
Contact map
Categorical map (5 bins)
(b) Estimated map
Amino acid sequence
I E K P K I S V A F I A L G N F C R S P M A E A
. . . . M N K F L F V C T G N L Q R S P T F E I
. . D T K V K V L V V C L G N I C R S P L G E A
. . D E C R S V L F V C L G N I C R S P T S E A
. . S D S K G I L F V C F G N I C R S P M A E A
. . A D K V N V L F V C L G N I C R S P M A E A
. . . M T L N V L V V C L G N I C R S P I G E A
Multiple sequence analysis, MSA (HHblits)
Estimated contact map (CCMpred)

Figure 3.1: Image representations of the protein with PDB id *1d1q* (chain *A*). **(a)** Maps obtained from the 3D structure of the protein: real-valued distance map, binary contact map (threshold 8 Å), and categorical map using 5 bins. **(b)** Estimated contact map, obtained from applying multiple sequence analysis (MSA) on the amino acid sequence of the protein. Darker gray level indicates closer 3D distance for the native maps, and higher probability of contact for the estimated map.

which results from applying a distance threshold of 8 Å over the distance map; and the *categorical map* [10, 11], a coarse approximation of the distance map which is generated by quantizing the distance values in several bins. To obtain the optimal intervals, we computed the Euclidean distances between beta-carbon atoms in a set of 3127 protein domains from SCOPe version 2.06 (training set from [5]). The histogram of the distance values is plotted in Figure 3.2. As can be seen, the histogram shows three local maximum peaks at 5.53 Å, 7.48 Å, and 10.73 Å, while the global maximum peak is at 20.48 Å, after which the frequency decreases exponentially with the distance. Based on this information,

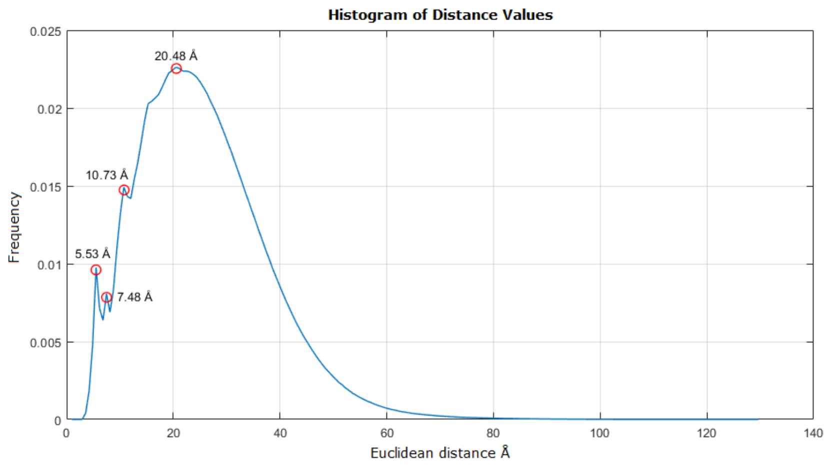


Figure 3.2: Histogram of the Euclidean distance values (measured in Å) in a set of 3127 protein domains from SCOPe (version 2.06).

we decided to use the following 5 (quantization) intervals in our categorical maps: $[0,5)$, $[5,8)$, $[8,15)$, $[15,20)$, and $[20,+\infty)$ Å.

Estimated and enhanced maps

For the predicted image representations of the protein we first considered the *estimated contact map* (Figure 3.1b) using the evolutionary coupling-based method CCMpred [2]. This method works on evolutionary information extracted from the multiple sequence alignment (MSA) for each protein. The MSA was obtained running HHblits [12] to search for homologous sequences in the `uniprot20_2016_02` database [13]. The CCMpred method then uses this information to estimate correlated mutations between each pair of amino acid residues, generating a symmetrical contact likelihood matrix as a result.

In addition, in this chapter we use both CCMpred estimated contact maps and native maps to obtain enhanced versions of the protein contact and categorical maps. To do so, we train a fully-convolutional network (FCN) on estimated contact maps, called *CMap-FCN*, which we describe in detail in the following section.

3.2.2 *CMap-FCN* for protein contact map enhancement

Model architecture and training

We followed-up on our previous work [5] for contact map improvement using denoising methods. More specifically, we adapted the fully-convolutional network (FCN) with residual learning to work on full images instead on patches. The original model architecture (*DenDCNN* [5, 14]) was proposed for image denoising, and consisted of an initial 2D convolutional layer followed by 17 intermediate layers, all applying 64 convolution filters of size 3×3. The final convolutional layer combined the 64 filters of the last intermediate layer by applying 1 filter of size 1×1 to obtain the output gray-scale image. At each layer (except for the last one), the rectified linear unit (ReLU) was used as activation function, and batch normalization [15] was applied in all the intermediate layers. In addition, the original model was trained to learn differences between the network input and output image (residual learning), and then reconstruct the image using the input and the residual map. Thereby, the loss function computed the mean squared error (MSE) between the native contact map image and the reconstructed one.

Here, we made several changes to the aforementioned model that improved performance. In the following, we refer to the input, enhanced, and native maps as $\mathbf{x}$, $\hat{\mathbf{x}}$, and $\mathbf{y}$, respectively. First, we simplified the model by removing the residual connection and so trained the model to directly estimate the enhanced contact/categorical map at the output. We also applied dropout [16] with drop probability 0.2 to every third intermediate layer. The training was performed on full images instead of patches, using a batch size of 1. Since the native contact/categorical map is a symmetric matrix, we applied a symmetrization

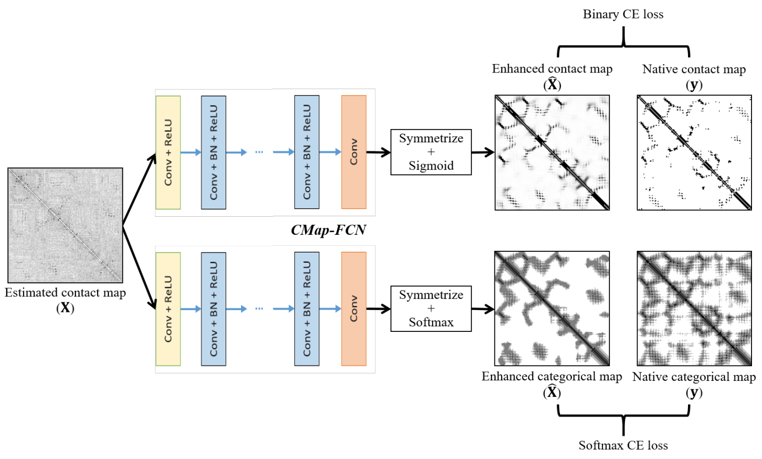


Figure 3.3: Overview of the *CMap-FCN* model for protein contact and categorical map enhancement. The model architecture has been taken from [14]. Here we used the protein with PDB id *1d1q* (chain *A*) as an example.

operation to the output image as $\left(\hat{\mathbf{x}} + \hat{\mathbf{x}}^T\right)/2$. Moreover, we changed the MSE to a negative log loss function, as it is better suited for classification tasks in which the aim is to predict the likelihood that the input belongs to a specific class. We optimized the model for 50 epochs using Adam [17] with learning rate 10^{-3}.

We refer to this adapted model as *CMap-FCN* (Figure 3.3). In fact, we trained two different models with the same architecture but different training targets: one to predict binary contact maps and another one for categorical map prediction using 5 bins (see Section 3.2.1). Therefore, the last convolutional layer applied either 1 or 5 filters, for contact and categorical map prediction, respectively. Two different loss functions have been used here: binary and softmax cross-entropy. To account for the imbalance between classes, we added weights for each class to the loss.

- Weighted binary cross-entropy loss for contact map prediction, computed as

$$L_{bce}(\Theta) = -\frac{1}{N}\sum_{n=1}^{N}\sum_{i\in y_n} \mathbf{w}_{ni}\left[\mathbf{y}_{ni}\log(\sigma(\hat{\mathbf{x}}_{ni})) + (1-\mathbf{y}_{ni})\log(1-\sigma(\hat{\mathbf{x}}_{ni}))\right], \tag{3.1}$$

  where N is the number of samples in a batch, n is each element in the flattened matrices, i refers to the class, $\sigma(z) = 1/(1+\mathrm{e}^{-z})$ is the sigmoid function, and $\mathbf{w}_n$ is a weight vector of the same size as $\mathbf{y}_n$ containing the following values:

$$\mathbf{w}_{ni} = \begin{cases} 3 & \text{if } \mathbf{y}_{ni} = 1 \quad \text{(positive contact)} \\ 1 & \text{if } \mathbf{y}_{ni} = 0 \quad \text{(negative contact)} \end{cases} \tag{3.2}$$

- Weighted softmax cross-entropy loss for categorical map prediction, computed as

$$L_{ce}(\Theta) = -\frac{1}{N}\sum_{n=1}^{N}\sum_{i\in y_n}\mathbf{w}_{ni}\log(\mathbf{p}_{ni}) = -\frac{1}{N}\sum_{n=1}^{N}\sum_{i\in y_n}\mathbf{w}_{ni}\log\frac{\exp\left(\hat{\mathbf{x}}_{ni,\mathbf{y}_{ni}}\right)}{\sum_{k=1}^{K}\exp\left(\hat{\mathbf{x}}_{ni,k}\right)}, \tag{3.3}$$

  where K is the number of categories ($K = 5$ in this case), $\mathbf{y}_{ni}$ is the true class (as a one-hot vector of K elements). Here, the weight vector $\mathbf{w}_n$ contains the following values:

$$\mathbf{w}_{ni} = \begin{cases} 4 & \text{if } \mathbf{y}_{ni} = \{0,1\} \quad \text{(distance bins } [0,5) \text{ and } [5,8) \text{ Å)} \\ 2 & \text{if } \mathbf{y}_{ni} = \{2,3\} \quad \text{(distance bins } [8,15) \text{ and } [15,20) \text{ Å)} \\ 1 & \text{if } \mathbf{y}_{ni} = 4 \qquad\quad \text{(distances} \geq 20 \text{ Å)} \end{cases} \tag{3.4}$$

Evaluation

We evaluated the contact prediction performance of the *CMap-FCN* model by comparing its output with the native contact map. Following the evaluation in [5] we first divided the contacts into three groups depending on the position of the amino acid residues in the sequence. These groups are: short-range (residue separation between 6 and 11 positions), medium-range (from 12 to 23) and long-range (greater than 23). For each group, we reported the precision accuracy of the L/k contacts with the highest probability, where L is the sequence length and $k = \{10; 5; 2; 1\}$ controls the number of contacts to be evaluated.

Training and test sets

To train and evaluate the *CMap-FCN* models, we used the datasets described in our previous work [5]. That is, a training set with 3427 protein domains, from which we extracted a validation set with 300 domains; and three test sets for evaluation: 150 Pfam proteins (Pfam150), 116 proteins from CASP10 (2012), and 103 from CASP11 (2014).

3.2.3 *Fold-DCNN* for protein fold recognition

Model architecture

We performed protein fold recognition following the pipeline proposed in the DeepFR method [1]. This method trained a deep convolutional neural network (DCNN) model to classify input protein domains into the fold classes defined by SCOPe [4] (version 2.06). The DCNN model proposed in [1] is depicted in Figure 3.4. The inputs to the original model were estimated contact maps using CCMpred. In addition to these, here we also compare the different protein image representations described in the previous section. Particularly, the DCNN model takes as inputs 227×227 crops of images with size 256×256 and one channel. The first 2D convolutional layer applies 96 filters of size 7×7 and stride 2 in both spatial dimensions. The outputs are passed through the ReLU function, max-pooled (in regions of size 3×3 and using a stride of 2), and batch normalized [15]. This

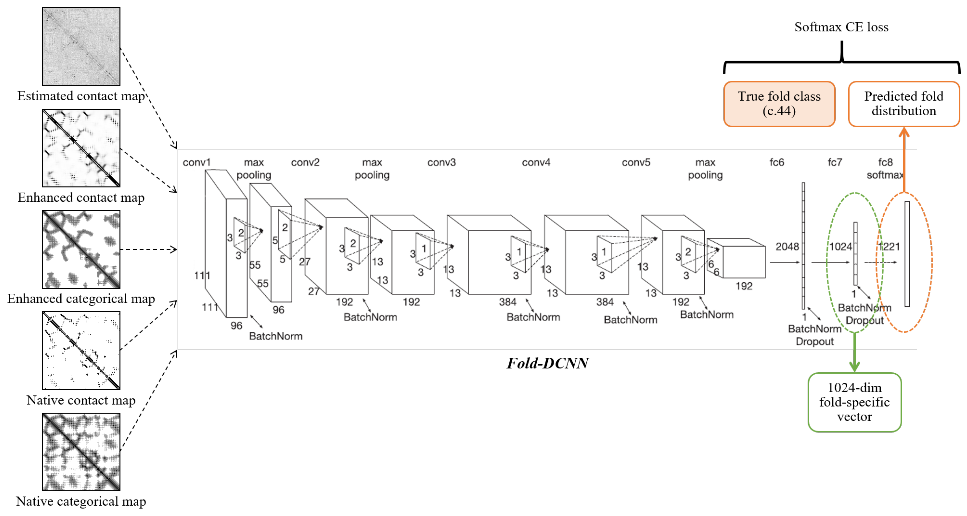


Figure 3.4: Overview of the *Fold-DCNN* model for protein fold recognition using one of the 5 image representations of the protein: estimated contact map (from CCMpred), enhanced contact and categorical maps (from *CMap-FCN*), and native contact and categorical maps (from the PDB). The model architecture has been taken from DeepFR [1]. Here we used as an example the protein domain with SCOPe id *d1d1qa_* and fold class *c.44*.

results in 96 feature maps of size 55×55. A similar process is repeated in the following convolutional layers (from 2 to 5), but max-pooling is skipped in the 3rd and 4th layers. The final 192 feature maps of size 6×6 are flattened in a 6912-dimensional vector. This vector is processed by two fully-connected layers with 2048 and 1024 units, followed by ReLU activation and batch normalization. The 1024-dimensional output of this layer is extracted in the evaluation phase as fold-specific features to compute similarity between protein domains. The final linear layer has 1221 units, equal to the number of fold types in SCOPe 2.06, and applies the softmax operation to output a predicted distribution over fold classes. We call this model *Fold-DCNN* (Figure 3.4) to differentiate it from the model described in the previous section for contact/categorical map enhancement *CMap-FCN* (Figure 3.3).

Training strategies

We trained a different *Fold-DCNN* model for each of the 5 image representations of the protein we considered here (Figure 3.4). To fix the size of the input images to 256×256, we followed the two strategies proposed in [1]. On the one hand, the first strategy (*s1*) uses a single input image for each protein domain. Each input map is resized by applying local averaging or nearest-neighbor interpolation, depending on whether the sequence length is greater or less than 256, respectively. On the other hand, the second strategy (*s2*) considers an ensemble of multiple matrices for each domain (number proportional to the sequence length). Instead of resizing, in this case we either sample crops from the original map or embed it in random positions into a 256×256 matrix initialized to zeros. This set of ensemble matrices is fed into the *Fold-DCNN* model and the outputs are averaged.

Common to both strategies, we trained the models by minimizing the cross-entropy loss between the predicted class distribution and the true fold class, using mini-batches with 16 input maps each. The optimization was performed using the stochastic gradient descent (SGD) algorithm [18] with momentum set to 0.9 and an initial learning rate of 10^{-3}. As in [1], the learning rate was reduced by a factor of 10 every 200k iterations, and the whole optimization process was completed after 500k iterations. To prevent overfitting, we applied L_2 penalty with a small weight decay of $5 \cdot 10^{-4}$, and dropout [16] with a drop probability of 0.5 in the fully-connected layers. We used the original implementation[1] in `caffe` [19], and run it on a single GPU card with 12GB of memory.

Evaluation

For each protein domain contained in the test sets, we extracted a 1024-dimensional fold-specific feature vector from the trained *Fold-DCNN* model (*fc7* layer outputs in Figure 3.4). Then, we compared the feature vectors of every two protein domains using the cosine similarity (DeepFR score in [1]) as comparison metric. For each individual domain in the test set (query), we ranked all other domains (templates) according to the cosine similarity score, and then assigned the fold type of the most similar template to the query. The evaluation was performed at three levels of SCOP: *family*, *superfamily*, and *fold*, as proposed in [20]. For each level, we reported the accuracy of finding the true fold class in the top 1 and top 5 ranked templates as performance metrics.

Training and test sets

We used the datasets described in [1] for training and evaluating the *Fold-DCNN* models. The training set consists of 16133 protein domains from SCOPe (version 2.06) covering 1154 folds. This set resulted from removing any redundant sequence with respect to the test datasets (maximum sequence identity of 40% and e-value $< 10^{-4}$) and within the training set itself (sequence identity $< 95\%$). We also used the validation set with 3760 domains.

The performance of the *Fold-DCNN* models was evaluated on two test sets: SCOP_TEST [21] and LINDAHL [20]. The SCOP_TEST set contains 124 domains from SCOP (version 1.75), which cover 14 fold classes, while LINDAHL has 976 domains from SCOP (version 1.37) covering a total of 330 folds. The maximum sequence identity between domains in each test set is 40%. The number of individual domains evaluated at the family, superfamily, and fold levels, respectively, are as follows: 106, 56, and 36 for the SCOP_TEST set; and 555, 434 and 321 for the LINDAHL set.

[1]`https://github.com/zhujianwei31415/deepfr`

3.3 Experiments and results

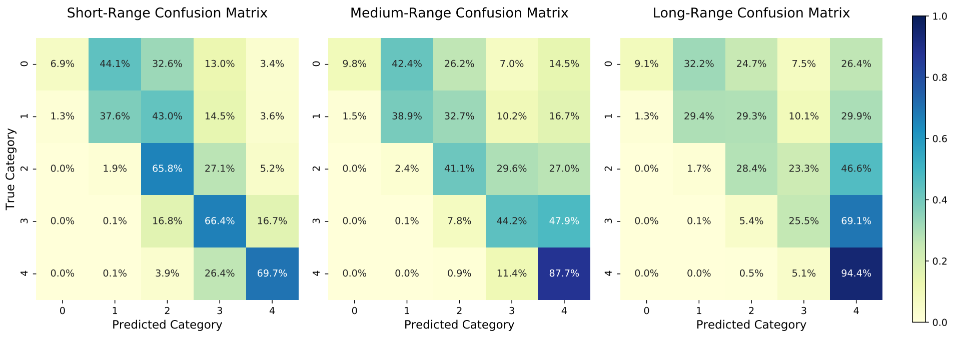
3.3.1 Performance of *CMap-FCN* on predicting contacts and categorical distances

We first evaluated the performance of the *CMap-FCN* models to predict protein contacts. We assessed both the enhanced contact and categorical maps provided by these models. For categorical maps, we summed the softmax scores of the first two bins, corresponding to $[0, 5)$ and $[5, 8)$ Å, to obtain the likelihood of the spatial distance being less than 8 Å. In Table 3.1 we report the contact precision results for short-, medium- and long-range, using as a reference the results in [5] for the CCMpred and denoising DCNN (*DenDCNN*) methods. As can be seen, the *CMap-FCN* models yield better results than the original *DenDCNN* model in most cases. In addition, we observe small differences in contact precision when comparing the enhanced contact maps to the categorical maps obtained by *CMap-FCN*. Here, the enhanced contact maps provide better results overall, particularly for the CASP11 test set, which could be explained by the model being trained to only discriminate between contacts and non-contacts.

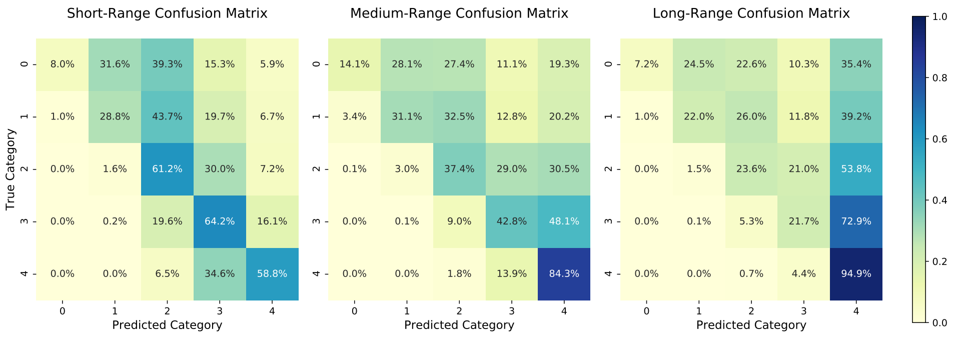
We then tested the ability of the *CMap-FCN* model to classify spatial distances into 5 bins (categorical maps). Figure 3.5 plots the average confusion matrices (color mapped) for each test set, differentiating between short-, medium- and long-range groups of residue pairs. Here, each row of the confusion matrix has been normalized to add up to 100%.

Table 3.1: Contact precision values for short-, medium- and long-range for the evaluated methods on the Pfam150, CASP10 and CASP11 test sets. Boldface indicates best performance per test set.

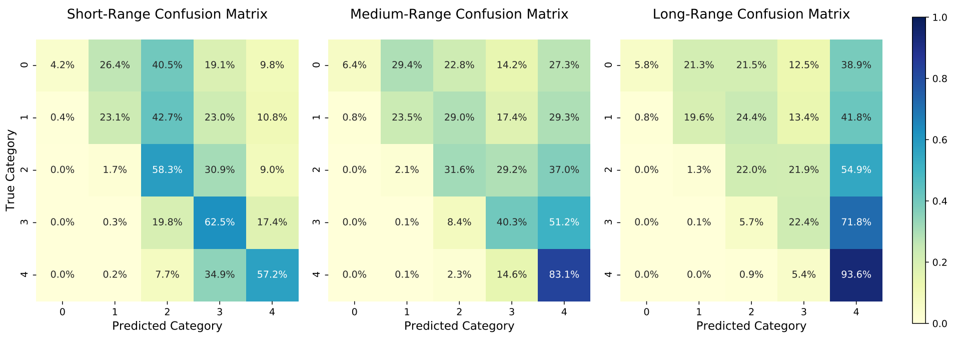
Test set	**Method**	**Short-range**				**Medium-range**				**Long-range**			
		L/10	***L/5***	***L/2***	***L***	***L/10***	***L/5***	***L/2***	***L***	***L/10***	***L/5***	***L/2***	***L***
Pfam150	CCMpred [5]	56.1	40.2	23.0	15.5	64.2	49.8	29.0	18.4	78.1	71.0	50.5	33.7
	DenDCNN [5]	77.6	65.2	40.9	25.0	80.5	**71.0**	**48.3**	30.3	89.6	85.3	**72.1**	**54.5**
	CMap-FCN												
	—Contact	78.9	66.0	**41.3**	**25.3**	**81.3**	70.6	48.0	**30.6**	**91.3**	85.9	71.0	54.4
	—Categorical	**79.2**	**66.1**	41.0	24.8	**81.3**	70.4	47.9	30.3	91.2	**86.6**	71.4	54.1
CASP10	CCMpred [5]	41.5	31.2	19.4	13.5	53.1	41.9	26.3	18.1	53.7	47.8	34.4	23.1
	DenDCNN [5]	58.4	48.5	31.9	20.7	65.8	58.1	43.0	29.8	67.3	63.2	50.6	37.6
	CMap-FCN												
	—Contact	**60.4**	**50.4**	**33.7**	**21.3**	66.8	**59.6**	**43.1**	**30.3**	68.9	63.4	**51.2**	**38.3**
	—Categorical	59.7	50.0	33.2	**21.3**	**67.2**	58.5	42.4	29.9	**70.1**	**64.1**	50.5	37.5
CASP11	CCMpred [5]	32.9	23.9	15.3	11.3	38.0	28.5	17.8	12.5	47.4	40.2	28.9	20.1
	DenDCNN [5]	48.1	38.8	26.1	17.6	53.7	46.6	31.9	21.2	56.5	53.2	43.0	32.6
	CMap-FCN												
	—Contact	**51.5**	**43.3**	**27.6**	**18.1**	**54.3**	**47.4**	**32.9**	**21.5**	**61.0**	**56.6**	**45.4**	**34.1**
	—Categorical	49.0	40.4	26.3	17.7	54.1	46.7	31.9	21.2	60.1	55.7	44.5	33.5



(a) Pfam150 Test Set



(b) CASP10 Test Set



(c) CASP11 Test Set

Figure 3.5: Average confusion matrices of the categorical maps (5-bin classification) for short-, medium- and long-range for the *CMap-FCN* method on the **(a)** Pfam150, **(b)** CASP10 and **(c)** CASP11 test sets.

This shows, for every true category, the percentage of predictions that lie in each of the 5 categories. The diagonals therefore include the accuracy results for each numbered category from 0 to 4, corresponding to the 5 distance bins specified in Section 3.2.1. We observe that the classification performance is very similar for all test sets. In particular, larger spatial distances (bin $4 := [20, +\infty)$ Å) are better predicted for medium- and long-range, while the smaller distances (bin $0 := [0, 5)$ Å) are harder to predict at the three ranges. We note that the model tends to classify long-range residues as very far in space (prediction category 4). This could be explained by the large number of distances that fall into this category in the training data, especially for long-range residues. Nonetheless, it must be taken into account that we tried to minimize the impact of this by adding weights to each class (see Eq. 3.4). Moreover, the categorical maps have been predicted using contact estimations from CCMpred as inputs, which is based on co-evolution information alone. Thereby, these results could be improved by adding other types of input representations. However, since the main objective of this study is to compare the fold recognition performance between enhanced and native contact/categorical maps, the individual classification results of the categorical maps predicted by *CMap-FCN* are not as crucial here.

3.3.2 Training the *Fold-DCNN* models: Validation accuracy

As a preliminary test, during training we tracked the performance of the *Fold-DCNN* models in classifying protein folds as a function of the image representation used as input. For this purpose, we evaluated the classification accuracy on an independent validation dataset at each training iteration. Figure 3.6 shows the validation accuracy for the two training strategies: single input image (*s1*) and multiple input images (*s2*). To provide better visual clues about the accuracy, we applied a Gaussian filter to smooth the validation curves. As we can see, the accuracy of all models improves and stabilizes after iteration

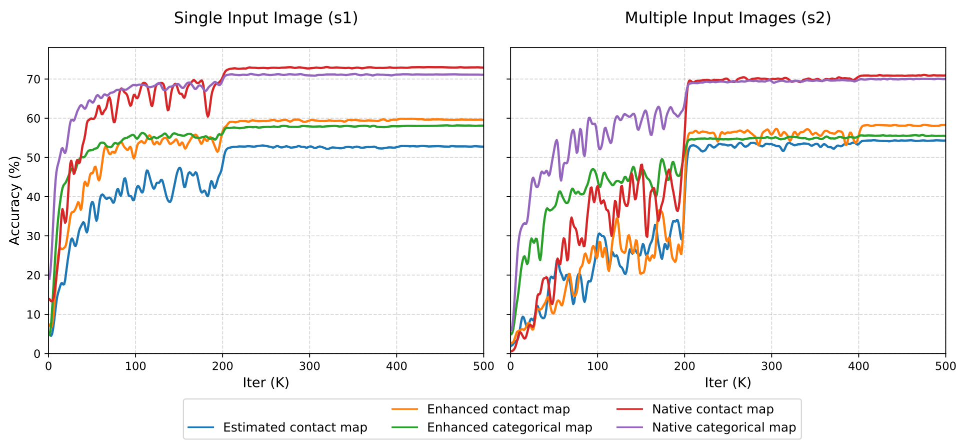


Figure 3.6: Validation accuracy (%) of the *Fold-DCNN* models for each iteration of training, using 5 distinct image representations of the protein: estimated contact map (from CCMpred), enhanced contact and categorical maps (from *CMap-FCN*), and native contact and categorical maps (from the PDB). Here we differentiate between two training strategies: single input image (*s1*) on the left and multiple input images (*s2*) on the right.

number 200k, when we first reduce the learning rate. For the *s1* strategy, we observe three distinct performance levels: one for estimated contact maps (~ 52%), one for enhanced maps (slightly below 60%), and one for native maps (above 70%). Here, the enhanced and native contact maps provide better results than the respective enhanced and native categorical maps. In contrast, for the *s2* strategy, the performance of the estimated and the two enhanced maps is more similar to one another (~ 55%); the same for the native contact and categorical maps (~ 70%). Altogether, the accuracy values of *s1* are slightly better than *s2*, particularly for the enhanced and native maps. However, the strategy *s2* can be seen as a data augmentation approach, in which we average the results of several images for the input protein domain. This could have a different impact on the final classification performance than on the quality of the fold-specific vector extracted from the *Fold-DCNN* model (see Figure 3.4), an issue we address in the next section.

3.3.3 Protein fold recognition performance of *Fold-DCNN*

The performance of the fold-specific vectors extracted from the *Fold-DCNN* models was evaluated on the SCOP_TEST and LINDAHL sets. For each test set, the pairwise fold recognition results at the family, superfamily and fold levels, considering the top 1 and top 5 predictions, can be found in Tables 3.2 and 3.3, respectively. We differentiate between the two training strategies: single input image (*s1*) and multiple input images (*s2*). We also compare the performance of the 5 distinct image representations to the state-of-the-art DeepFR method [1], which uses estimated contact maps from CCMpred as inputs.

In general, for both test sets, we observe that the models trained here using estimated contact maps as inputs perform worse than the original DeepFR model. This is particularly noticeable in the LINDAHL test set at the fold level (Table 3.3). Thereby, we were unable to reproduce the fold recognition results in [1], even when applying the same procedure

Table 3.2: Pairwise fold recognition accuracy (%) results at the family, superfamily and fold levels within the SCOP_TEST set. Underline indicates best performance per estimated/enhanced or native maps for each training strategy (single input image, *s1*, and multiple input images, *s2*). Boldface indicates best overall.

Input map		***Single input image (s1)***						***Multiple input images (s2)***					
		Family		**Superfamily**		**Fold**		**Family**		**Superfamily**		**Fold**	
		Top 1	***Top 5***	***Top 1***	***Top 5***	***Top 1***	***Top 5***	***Top 1***	***Top 5***	***Top 1***	***Top 5***	***Top 1***	***Top 5***
DeepFR [1]		74.5	91.5	76.8	87.5	77.8	80.6	75.5	93.4	78.6	83.9	86.1	88.9
Estimated	Contact	71.7	90.6	76.8	83.9	72.2	83.3	79.2	88.7	80.4	85.7	80.6	88.9
Enhanced	Contact	72.6	88.7	78.6	83.9	80.6	86.1	82.1	95.3	78.6	91.1	86.1	88.9
	Categorical	71.7	89.6	78.6	85.7	80.6	88.9	75.5	90.6	78.6	89.3	83.3	88.9
Native	Contact	75.5	92.5	76.8	87.5	88.9	88.9	81.1	**96.2**	**87.5**	**92.9**	**91.7**	**91.7**
	Categorical	77.4	90.6	75.0	89.3	88.9	**91.7**	**84.0**	94.3	85.7	**92.9**	**91.7**	**91.7**

Table 3.3: Pairwise fold recognition accuracy (%) results at the family, superfamily and fold levels within the LINDAHL set. Underline indicates best performance per estimated/enhanced or native maps for each training strategy (single input image, *s1*, and multiple input images, *s2*). Boldface indicates best overall.

Input map		***Single input image (s1)***						***Multiple input images (s2)***					
		Family		**Superfamily**		**Fold**		**Family**		**Superfamily**		**Fold**	
		Top 1	***Top 5***	***Top 1***	***Top 5***	***Top 1***	***Top 5***	***Top 1***	***Top 5***	***Top 1***	***Top 5***	***Top 1***	***Top 5***
DeepFR [1]		67.4	80.9	47.0	63.4	44.5	62.9	65.4	83.4	51.4	67.1	56.1	70.1
Estimated	Contact	68.8	81.3	42.9	58.5	35.2	57.9	70.3	84.0	50.9	70.0	45.2	65.4
Enhanced	Contact	67.4	80.5	46.3	65.7	45.5	66.0	70.1	85.9	61.5	76.0	56.7	74.5
	Categorical	65.4	78.9	42.9	63.1	45.5	66.7	69.7	83.6	59.7	74.2	53.3	71.3
Native	Contact	**74.2**	90.1	56.9	77.9	69.2	86.8	**74.2**	89.7	67.7	**86.2**	**79.4**	**94.4**
	Categorical	73.5	87.9	60.8	78.3	68.8	88.5	74.1	**91.0**	**68.0**	84.1	76.6	93.8

to generate the estimated contact maps and train the DCNN model. However, we do see that the performance of the strategy *s2* is considerably better than *s1* for all image representations at the three levels, in accordance with the original DeepFR results. When comparing the performance of the individual input images, we see that the native maps perform the best at the task, followed by the enhanced maps, and then both the estimated contact map and DeepFR. In addition, the enhanced and native contact maps tend to perform better than their respective categorical maps, which is consistent with the validation results in Figure 3.6.

In particular, if we focus on the results of strategy *s2*, for the SCOP_TEST set (Table 3.2) we see that the enhanced contact maps achieve top 1 accuracy values of 82.1% (87/106 hits), 78.6% (44/56 hits), and 86.1% (31/36 hits) at the family, superfamily, and fold levels, respectively. These results are outperformed by the native contact maps at the superfamily and fold levels, obtaining 87.5% (49/56 hits) and 91.7% (33/36 hits) top 1 predictions, which are very similar to the results obtained by native categorical maps. For the LINDAHL set (Table 3.3), the enhanced contact maps yield top 1 accuracy values of 70.1% (389/555 hits), 61.5% (267/434 hits), and 56.7% (182/321 hits) at the three levels, respectively. In this case, the native contact maps achieve significantly better results than the enhanced maps, especially at the fold level: 79.4% (255/321 hits) top 1 and 94.4% (303/321 hits) top 5 accuracy values. Here the differences in performance between contact and categorical maps are slightly larger at the fold level: 56.7% compared to 53.3% for the enhanced maps, and 79.4% compared to 76.6% for the native maps. Note that few domains are evaluated in the SCOP_TEST set, and therefore small errors in the predictions can lead to large changes in the accuracy values (especially at the fold level).

3.4 Discussion and conclusion

In this chapter we have analyzed the performance of different input image representations for the protein fold recognition task following the approach from DeepFR [1]. To do so, we trained deep convolutional neural network-based models (*Fold-DCNN*) to map input domains into fold classes, and then extracted fold-specific vectors to compare pairs of protein domains. As input images, we used either estimated contact maps (from CCMpred), enhanced contact and categorical maps, or native contact and categorical maps (from the PDB). The enhanced maps were obtained from fully-convolutional network-based models (*CMap-FCN*), trained to classify contacts and categorical distances from input estimated contact maps. Although the contact precision of the resulting enhanced maps is not excessively high, they perform better than estimated contact maps when used for recognizing protein folds. For this task, we have found that native contact maps are the most suitable input representation for the protein domains. This is an expected conclusion, since native maps are directly obtained from the protein 3D structure. However, while native categorical maps contain more information about the Euclidean distance between amino acid residues, they do not perform as well as the native contact maps for the task. This suggests that the *Fold-DCNN* model itself handles binary images better than gray-scale ones. In addition, it should be noted that in the *s1* training strategy, the model takes a degraded version of the input map, resulting from the resizing operation to 256×256, after which the residue–residue distance information is blurred in the matrix. This is partially mitigated by the *s2* training strategy, where fixed-size crops are randomly taken along the main diagonal of the matrix. However, this may cause a loss of information as well, since pairs of amino acid residues separated by more than 256 positions in the sequence are not seen by the model.

As for the input information about the protein domains, we could obtain improved fold recognition results by using better contact or distance estimation methods, such as the best performer in the CASP12 challenge [22]: the RaptorX-Contact [23] method. However, our results on the LINDAHL test set imply that, even when native contact maps are available, there is still room for improvement at all three levels (top 1 accuracy values are below 80%). Furthermore, in most cases we are interested in inferring the fold class from protein sequential information alone, as there is a general lack of solved protein 3D structures. Therefore, there exists a need to investigate models that better handle other types of sequence information, such as that extracted from the protein MSA (among others), which may perform better at recognizing protein folds than the *Fold-DCNN* model trained on estimated, enhanced or even native contact maps.

References

[1] J. Zhu, H. Zhang, S. C. Li, et al. Improving protein fold recognition by extracting fold-specific features from predicted residue–residue contaacts. *Bioinformatics*, 33(23):3749–3757, 2017.

[2] S. Seemayer, M. Gruber, and J. Söding. CCMpred—fast and precise prediction of protein residue–residue contacts from correlated mutations. *Bioinformatics*, 30(21):3128–3130, 2014.

[3] A. G. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia. SCOP: A structural classification of proteins database for the investigation of sequences and structures. *journal of Molecular Biology*, 247(4):536–540, 1995.

[4] N. K. Fox, S. E. Brenner, and J.-M. Chandonia. SCOPe: Structural classification of proteins–extended, integrating SCOP and ASTRAL data and classification of new structures. *Nucleic Acids Research*, 42(D1):D304–D309, 2014.

[5] A. Villegas-Morcillo, J. A. Morales-Cordovilla, A. M. Gomez, and V. Sanchez. Improved protein residue-residue contact prediction using image denoising methods. In *26th European Signal Processing Conference (EUSIPCO)*, pages 1167–1171, 2018.

[6] L. Holm and C. Sander. Protein structure comparison by alignment of distance matrices. *Journal of Molecular Biology*, 233(1):123–138, 1993.

[7] J. E. Chen, C. C. Huang, and T. E. Ferrin. RRDistMaps: a UCSF Chimera tool for viewing and comparing protein distance maps. *Bioinformatics*, 31(9):1484–1486, 2015.

[8] I. A. Emerson and A. Amala. Protein contact maps: A binary depiction of protein 3d structures. *Physica A: Statistical Mechanics and its Applications*, 465:782–791, 2017.

[9] S. H. P. de Oliveira, J. Shi, and C. M. Deane. Comparing co-evolution methods and their application to template-free protein structure prediction. *Bioinformatics*, 33(3):373–381, 2017.

[10] I. Walsh, D. Baù, A. J. M. Martin, et al. Ab initio and template-based prediction of multi-class distance maps by two-dimensional recursive neural networks. *BMC Structural Biology*, 9(1):1–20, 2009.

[11] P. Kukic, C. Mirabello, G. Tradigo, et al. Toward an accurate prediction of inter-residue distances in proteins using 2d recursive neural networks. *BMC Bioinformatics*, 15(1):1–15, 2014.

[12] M. Remmert, A. Biegert, A. Hauser, and J. Söding. HHblits: lightning-fast iterative protein sequence searching by HMM-HMM alignment. *Nature Methods*, 9(2):173–175, 2012.

[13] The UniProt Consortium. UniProt: the universal protein knowledgebase. *Nucleic Acids Research*, 45(D1):D158–D169, 2017.

[14] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017.

[15] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *International Conference on Machine Learning*, pages 448–456, 2015.

[16] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[17] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[18] L. Bottou. Stochastic gradient descent tricks. In G. Montavon, G. B. Orr, and K.-R. Müller, editors, *Neural Networks: Tricks of the Trade*, pages 421–436. Springer Berlin Heidelberg, 2012.

[19] Y. Jia, E. Shelhamer, J. Donahue, et al. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM International Conference on Multimedia*, pages 675–678, 2014.

[20] E. Lindahl and A. Elofsson. Identification of related proteins on family, superfamily and fold level. *Journal of Molecular Biology*, 295(3):613–625, 2000.

[21] T. Jo, J. Hou, J. Eickholt, and J. Cheng. Improving protein fold recognition by deep learning networks. *Scientific Reports*, 5:17573, 2015.

[22] J. Schaarschmidt, B. Monastyrskyy, A. Kryshtafovych, and A. M. J. J. Bonvin. Assessment of contact predictions in CASP12: co-evolution and deep learning coming of age. *Proteins: Structure, Function, and Bioinformatics*, 86:51–66, 2018.

[23] S. Wang, S. Sun, Z. Li, R. Zhang, and J. Xu. Accurate de novo prediction of protein contact map by ultra-deep learning model. *PLoS Computational Biology*, 13(1):e1005324, 2017.

4

Protein fold recognition from sequences using convolutional and recurrent neural networks

Amelia Villegas-Morcillo, Angel M. Gomez,
Juan A. Morales-Cordovilla, and Victoria Sanchez

This chapter has been published as follows: A. Villegas-Morcillo, A.M. Gomez, J.A. Morales-Cordovilla, and V. Sanchez. Protein fold recognition from sequences using convolutional and recurrent neural networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 18(6):2848-2854, 2021. DOI: 10.1109/TCBB.2020.3012732.

- Status: Received in February 2020. Accepted and Published Online in July 2020. Issued in December 2021.
- Impact Factor (JCR 2021): 3.702. Mathematics, Interdisciplinary Applications. Rank 16/108 (Q1).

Abstract

The identification of a protein fold type from its amino acid sequence provides important insights about the protein 3D structure. In this paper, we propose a deep learning architecture that can process protein residue-level features to address the protein fold recognition task. Our neural network model combines 1D-convolutional layers with gated recurrent unit (GRU) layers. The GRU cells, as recurrent layers, cope with the processing issues associated to the highly variable protein sequence lengths and so extract a fold-related embedding of fixed size for each protein domain. These embeddings are then used to perform the pairwise fold recognition task, which is based on transferring the fold type of the most similar template structure. We compare our model with several template-based and deep learning-based methods from the state-of-the-art. The evaluation results over the well-known LINDAHL and SCOP_TEST sets, along with a proposed LINDAHL test set updated to SCOP 1.75, show that our embeddings perform significantly better than these methods, specially at the fold level. Supplementary material, source code and trained models are available at `http://sigmat.ugr.es/~amelia/CNN-GRU-RF+/`.

Index terms: *Protein Fold Recognition, Deep Learning, Convolutional Neural Networks, Recurrent Neural Networks, Embedding Learning, Random Forests*

4.1 Introduction

The determination of the protein's tri-dimensional structure from its amino acid sequence is one of the main challenges in structural biology. Its importance is given by the close relationship between the protein's 3D structure and its biological function. Although structure prediction at atomic level is a hard problem, knowing the fold type [1] [2] the protein belongs to may disclose relevant information about its structure and function [3]. Protein fold prediction can be accomplished by comparison with related proteins from the Protein Data Bank (PDB) database [4] that are already classified in different levels according to sequential and structural similarities [5].

The Structural Classification of Proteins (SCOP) database [6] and its latest extended version SCOPe [7] provide a hierarchical classification of protein domains in structural classes, folds, superfamilies, families, proteins and species. According to SCOP, structural class and fold levels comprise domains that share structural features and similar topology, and do not imply sequence homology as those in the same superfamily or family. Moreover, it has been estimated that the number of possible folds is limited in nature [8].

State-of-the-art computational methods aim to identify protein folds following two main approaches, which are known as protein fold classification and protein fold recognition. The taxonomy-based fold classification approach [9] can be viewed as a typical multi-class classification problem, in which the protein sequences are directly mapped into fold classes. The machine learning-based methods, such as [10], FP-Pred [11], ACCFold [12], TAXFOLD [13], HMMFold [14] and ProFold [15], were designed to successfully classify into a predefined group of SCOP fold classes. However, the selected folds comprise a small set including only those folds with a higher amount of protein domains (27 folds in [10] [12] or 30 folds in [16]), in contrast to the more than one thousand existing fold classes in the SCOP database.

On the other hand, the protein fold recognition approach is derived from the template-based structure prediction problem (homology modelling and threading), where the fold type is inferred by comparing with template proteins with known structure [17] [18]. In the fold recognition methods, the query protein is compared with a set of templates and the fold class of the most similar template is transferred to the query [19]. Homology modelling methods recognize template proteins which present a high sequence similarity with the query protein. This can be assessed by sequence-to-sequence alignment [20] or profile-to-profile alignment using hidden Markov models (HHpred [21]) and Markov random fields (MRFAlign [22]). On its part, threading methods are suitable for those proteins that have low similarity in sequence, and try to match the query sequence with template structures using structural properties. Threading methods include RAPTOR [23], BoostThreader [24], SPARKS-X [25], CNFpred [26] and [27], mostly based on conditional random fields, and CEthreader [28]. More recently, the fold recognition task has been addressed as a binary classification problem for each query-template protein pair by using machine learning tools such as support vector machines (FOLDpro [29]), random forests (RF-Fold [30]) and deep neural networks (DN-Fold [31]). Furthermore, other ensemble methods that combine multiple feature types (describing amino acid sequences, evolutionary changes or structure properties) and different prediction techniques have been proposed, such as the machine learning with template-based recognition TA-Fold [32], the multi-view model MT-fold [33], and the learning to rank Fold-LTR-TCP [34] methods.

In the past few years, deep learning [35] approaches have been introduced to improve the existing protein fold recognition methods. These methods rely on training a deep neural network to classify the whole set of SCOP fold classes. Then, the outputs of the last hidden layer are extracted as a fold-related embedding vector for each protein domain, which is used to perform pairwise protein fold recognition. Among the state-of-the-art techniques we can find the deep learning methods DeepFR [36], DeepSF [37], DeepSVM-fold [38] and MotifCNN-fold [39]. The proposed architectures are mainly based on applying convolution operations over image-form features (predicted contact maps) and protein residue-level features (evolutionary and secondary structure predictions). However, convolutional networks only exploit local relations within the protein sequence. Another hurdle is that protein sequences cover a wide range of possible lengths (from 10 to 10000 amino acids), but feed-forward neural network architectures require a fixed-size input. In order to tackle these issues, we propose here a different architecture which introduces a recurrent layer after the CNN to obtain a fixed-size representation from the variable-length input sequence.

Recurrent neural networks (RNN) with long short-term memory (LSTM) units [40] have been employed in sequential problems such as speech recognition [41] and machine translation [42], as well as in the field of proteomics, addressing the protein homology detection [43] and protein secondary/tertiary structure prediction [44] [45] tasks. Moreover, architectures that combine deep CNN with LSTM have been proposed in proteomics to improve the predictions of subcellular localizations of proteins [46], residue-residue contact maps [47], protein secondary structure [48] and local backbone angles [49]. Thereby, the promising results achieved in these studies suggest that neural networks including recurrent layers can also be suitable for addressing the protein fold recognition task, as shown by the recent DeepSVM-fold [38] method. However, the neural network architecture introduced in [38] imposes limitations on the length of the protein sequences to be processed. In this paper we go one step further and propose a more straightforward architecture that can handle protein sequences of any length.

Thus, the contributions of this work are several. First of all, we propose a neural network architecture that combines convolutional and recurrent layers, which is able to process protein residue-level features of arbitrary length, and classify them into all the SCOP fold classes. Secondly, we leverage the trained neural network to extract a fold-related embedding suitable for performing pairwise fold recognition. Finally, we provide evaluation results by combining our similarity scores with other pairwise measures in a random forest classifier, which outperforms all state-of-the-art existing methods. A complete overview of our fold recognition approach is depicted in Figure 4.1.

4.2 Materials and methods

4.2.1 Feature extraction

In this work, we represent the protein domain with variable-length L as a sequence of L vectors, each one of size 45 representing each amino acid residue in the domain sequence (Figure 4.1a), as in [37]. These 45 residue-level features include information about the amino acid, evolutionary profile, secondary structure and solvent accessibility. Thus, the first 20 elements contain a one-hot vector representation of the amino acid. The profile information is given by the position-specific scoring matrix (PSSM) that contains 20 elements for each amino acid position, and is computed by PSI-BLAST [50] using the non-redundant database ‘nr90’ for sequence homology searching. Secondary structure and solvent accessibility information are predicted, respectively, by the SSpro and ACCpro methods from the SCRATCH suite [51]. This secondary structure is then encoded by a one-hot vector of three elements which correspond to helix, strand and loop structural

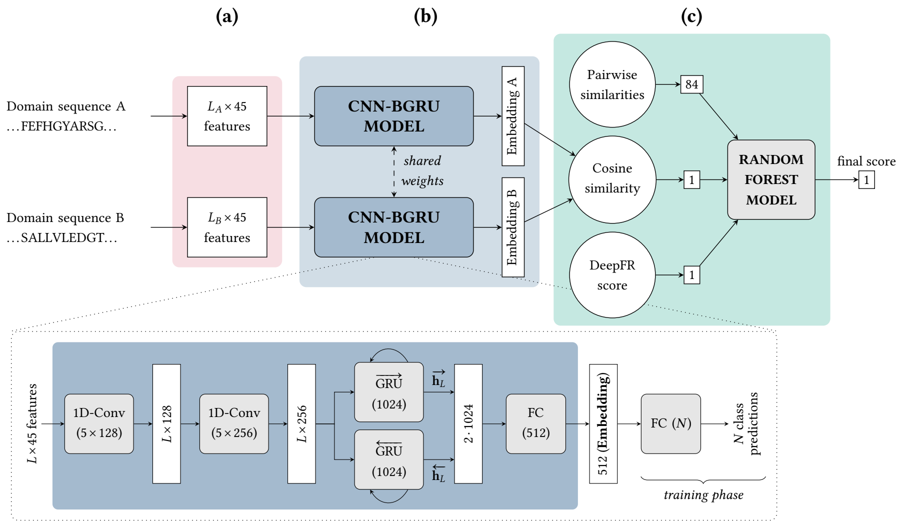


Figure 4.1: The proposed fold recognition approach to obtain a fold similarity score for two protein domains. **(a)** First, 45 input features are extracted for each amino acid in each of the two domain sequences. **(b)** The resulting $L \times 45$ residue-level features are passed through the CNN-BGRU model, previously trained to map protein sequences into fold classes. The model architecture (bottom part) consists of two one-dimensional convolutional layers, a bidirectional gated recurrent unit (GRU) layer and two fully-connected (FC) layers. The CNN-GRU model is very similar but with a unidirectional GRU instead and hence an output of size 1024. The two input protein domains are processed independently by the same network model (i.e. with identical trained weights). This model is then used to extract a fold-related embedding vector for each one (from the 512-dimensional output of the first FC layer). **(c)** The cosine similarity distance is computed between the two embeddings, which is concatenated to other similarity measures in a vector to obtain the final fold similarity score using a random forest model.

classes, whereas the solvent accessibility is encoded by a one-hot vector of two elements corresponding to exposed and buried states. The resulting $L \times 45$ features for each protein domain are then used as input to the neural network model, which is trained to map it into one of the SCOP fold classes.

4.2.2 Convolutional-recurrent neural network model

Our neural network model architecture (Figure 4.1b) is formed by a concatenation of three different types of layers: convolutional (CNN), recurrent (RNN) and fully-connected (FC). The details and purpose of each part of the network are described in the following subsections. The number of layers, filters, units per layer, and the rest of hyperparameters which configure the network, were chosen after conducting a cross-validation study (described in Section 4.2.4).

Convolutional neural network

The purpose of the convolutional neural network (CNN) part is to capture the local context of each amino acid in the protein domain and discover patterns in the sequence. To this end, several 1D-CNN layers are used to convolve the inputs across the sequence dimension.

In contrast to conventional 2D-CNN networks, frequently used in image processing, 1D-CNN are intended for sequence processing. In these networks, a kernel of learned weights, or convolutional filter, slides across only one dimension (instead of two). In our model, the sliding is done through the sequence dimension (from amino acid $l = 1$ to $l = L$). Thus, a filter of length k (and deep 45) is applied over the residue-level features of each amino acid l in the sequence along with those in its local context (that is, from $l - \frac{k-1}{2}$ to $l + \frac{k-1}{2}$). As each filter yields a value per amino acid and several filters are learned and applied to the same layer, the output of the 1D-CNN layer is of size $L \times K$, where K is the number of filters.

In our model, two 1D-CNN layers, using $K_1 = 128$ and $K_2 = 256$ filters of length $k_1 = k_2 = 5$, respectively, are applied over the $L \times 45$ residue-level input features (Figure 4.1b). Note that the use of 1D convolutions allows the model to be insensitive to the residue-level feature size. Thus, an arbitrary number of features per amino acid could be used instead of 45 as this only implies a different deep (inner size) of the first-layer's convolutional filters. After each 1D-convolutional layer, ReLU activation and Batch-Normalization [52] are applied.

Recurrent neural network

Unlike feed-forward neural networks, where the information only flows from the input to the output, recurrent neural networks (RNNs) introduce iteration loops that allow information to travel from the output to the input of the network as well. Thus, a typical RNN uses its own output at a previous iteration, along with the input at the current one, to compute the current output. For this reason, RNNs are frequently used for sequential data processing, exploiting long-distance relations and summarizing the sequence.

In this case, RNNs are used to iteratively process the amino acid sequence. In each iteration (i.e. for each amino acid l in the sequence), the current output is computed using the RNN output at the previous iteration (from the previous amino acid $l - 1$) which, in turn, has been computed considering the output from the previous amino acid ($l - 2$) and so on. In our model, only the RNN output at the last iteration (from amino acid $l = L$) is retained as a summary of the whole domain sequence, while RNN outputs for intermediate amino acids ($l = 1, \dots, L - 1$) are discarded.

As vanilla RNN layers suffer from the vanishing/exploding gradient problem when the sequence length L increases, gated recurrent unit (GRU) based layers, as those proposed in [53], are used in this work instead. GRU-based layers solve the gradient problem by defining and updating a state vector at each network iteration l which summarizes relevant information about the previous amino acids in the sequence $(1,\ldots,l-1)$. More information about how this state vector is updated can be found in the Supplementary Material (Section 4.S). Thus, in each iteration, the output of a GRU layer is a vector of size H, where H is the number of state units in the GRU cell. Other authors propose retaining and post-processing the full sequence of state vectors along iterations (i.e. a matrix of $L \times H$ outputs) [38], which makes the model architecture dependent on the sequence length dimension. By contrast, as we mentioned above, in this work we only consider the last state vector ($\mathbf{h}_L$), after processing the entire sequence of length L. This fixed-size vector represents the whole variable-length protein domain sequence. In this way, the proposed model can process domain sequences of any arbitrary length.

Therefore, in our neural network architecture, the $L \times 256$ output from the previous 1D-CNN is fed into a GRU-based recurrent layer with $H = 1024$ state units, yielding a fixed-size vector of size 1024 (Figure 4.1b). We refer to this approach as CNN-GRU model. As we do not want to make assumptions about directionality relevance in protein sequence processing, the same sequence but in reverse order (i.e. from $l = L$ to $l = 1$), is also fed into another same-size GRU-based recurrent layer [54]. The outputs from both forward ($\overrightarrow{\mathbf{h}_L}$) and backward ($\overleftarrow{\mathbf{h}_L}$) GRU layers are then concatenated into a vector of 2048 elements. We refer to the model resulting from this bidirectional GRU as CNN-BGRU model.

Fully-connected layer

Finally, a fully-connected (FC) layer is used to learn a non-linear combination of the GRU output vector ($\overrightarrow{\mathbf{h}_L}$ of size 1024 in the CNN-GRU model or $[\overrightarrow{\mathbf{h}_L}, \overleftarrow{\mathbf{h}_L}]$ of size 2048 in the CNN-BGRU one) and create a fold-related embedding representing the protein domain. This FC layer consists of a dense layer with 512 units, after which the sigmoid activation function is applied.

While fold-related embeddings are the actual outputs of our model and the ones used for the fold recognition task during the test phase (Figure 4.1c), model training requires some additional steps to be successfully accomplished. Thus, to train the network for learning suitable fold-related embeddings, a direct fold classification task is employed. To this end, a second FC layer is added to the model (Figure 4.1b, bottom right). This dense layer contains N output units and performs a simple linear combination of the 512-dimensional embedding vector. Here, N is the number of fold classes in which the input proteins are going to be classified during training. The cross-entropy computed from model predictions and one-hot vectors encoding the true fold class is used as the loss function which guides the optimization process.

Table 4.1: Three levels SCOP_TEST, LINDAHL and LINDAHL_1.75 test sets for pairwise fold recognition.

Test set name	SCOP version	Full test set *Proteins*	Full test set *Folds*	Family level *Pairs*	Family level *Proteins*	Family level *Folds*	Superfamily level *Pairs*	Superfamily level *Proteins*	Superfamily level *Folds*	Fold level *Pairs*	Fold level *Proteins*	Fold level *Folds*
SCOP_TEST [31]	1.75	124	14	614	106	13	336	56	6	300	36	4
LINDAHL [19]	1.37	976	330	1646	555	121	2130	434	79	3662	321	38
LINDAHL_1.75	1.75	976	323	1532	547	120	1988	431	75	4188	356	43

4.2.3 Datasets

In our study, we use the same set of protein domains selected from the SCOPe version 2.06 database as in [36] to train the models. This training set includes 16133 domains with less than 95% pairwise sequence identity covering $N = 1154$ folds. In order to test the method, we considered the well-known SCOP_TEST [31] and LINDAHL [19] sets, which have been widely used in previous studies for assessing the pairwise fold recognition task. The SCOP_TEST set contains 124 protein domains from SCOP version 1.75, which cover 14 fold classes, while the LINDAHL set has 976 domains from SCOP version 1.37 covering a total of 330 folds. The maximum sequence identity inside both test sets and also regarding the training set is 40%. In order to test the performance of the methods on the pairwise fold recognition problem, the protein domains within each test set are paired and evaluated at family, superfamily and fold levels, as detailed in Section 4.2.5. The number of positive pairs at each level and the resulting individual domains to be evaluated are reported in Table 4.1. As can be seen, in the SCOP_TEST set, 106, 56 and 36 domains are evaluated at family, superfamily and fold levels respectively, whereas in the LINDAHL set, we evaluate 555, 434 and 321 domains at each level respectively.

In addition, it should be mentioned that the original LINDAHL dataset is quite old (1999) and the SCOP definition of folds has changed since then. Therefore, in this paper we also propose an update of the original LINDAHL from SCOP version 1.37 to version 1.75, where the family, superfamily and fold classes are consistent with the most recent versions of SCOPe. To do so, we searched for the original LINDAHL sequences in the SCOP v1.75 database and then we updated the modified domain identifiers and class labels. In Table 4.1, we show the total number of fold classes included in the new LINDAHL_1.75 set. As can be observed, the number of pairs and protein domains at each level have changed, so the evaluation at fold level is now carried out over more domains covering more fold classes.

4.2.4 Model training and validation

In order to decide the best hyperparameters for our CNN-GRU and CNN-BGRU models, we performed a 5-stage cross-validation over the training set. To ensure independence between cross-validation subsets, we split the family classes contained within each structural class into 5 groups. Next, we used 4 subsets to train the model and the remaining one to validate it. Tested hyperparameter values included model architecture configuration, learning rate and number of epochs. We selected those that achieved the highest average results to carry out a final training procedure comprising the entire training dataset.

During the training phase, the variable-length sequence domains were grouped into mini-batches of 50 samples applying zero-padding to the maximum length within each mini-batch. In the GRU layer, we kept the last state vector of each domain sample in the mini-batch (i.e. before zero-padding). We trained the network by minimizing the cross-entropy loss function for a fixed number of epochs. We used the Adam optimizer [55] with an initial learning rate of 10^{-3}, which we reduced to 10^{-4} in epoch number 40 as cross-validation suggested. The whole optimization process was completed in 80 epochs. In order to prevent overfitting to the most populated fold classes, we applied L2 penalty with a small weight decay of $5 \cdot 10^{-4}$, and dropout [56] with a drop probability of 0.2 in the CNN and the first FC layers.

Our neural network models were implemented using Pytorch [57] (version 1.0) and trained on a GPU card (NVIDIA GTX Titan X, 12GB). The GPU memory required for training our CNN-GRU and CNN-BGRU models was 4.0 and 6.6 GB, respectively. The network complexity was estimated in terms of MAC (multiply-accumulator unit) counts, yielding 2.4 MMACs per processed amino acid for the CNN-GRU model and 3.5 MMACs for the CNN-BGRU model. Moreover, the CNN-GRU and CNN-BGRU models needed 75 and 109 seconds per epoch, completing the 80 training epochs in approximately 2 and 3 hours, respectively.

4.2.5 Evaluation and similarity measures

Evaluation is conducted in terms of pairwise fold recognition accuracy, using the fold-related embedding vectors. In this task, all the test domains are paired and evaluated following the SCOP hierarchy at three levels with increasing difficulty, as proposed by [19]: (i) *family level*, populated with pairs of test domains that share the same family class, (ii) *superfamily level*, with pairs of test domains that share the same superfamily class, but not the same family, and (iii) *fold level*, with pairs of test domains that share the same fold class, but neither share the same family nor superfamily.

At each level, the individual protein domains are considered as queries which are compared to a pool of templates. We addressed this task in the same way as the DeepFR method [36]. To compare each pair of query-template domains, we first use the extracted fold-related embeddings from each neural network model (CNN-GRU or CNN-BGRU). This comparison is carried out by computing the cosine similarity distance between the query and the template embedding vectors (Figure 4.1c). Then, the templates for each query are ranked by score value and, following the nearest neighbor condition, the fold class of the most similar template is assigned to the query domain. We refer to these methods as

CNN-GRU and CNN-BGRU.

This first fold similarity score was enhanced by training a random forest (RF) model using our cosine similarity score along with the 84 similarity measures used by the DeepFRpro method [36]. These pairwise measures contain sequence similarity information, sequence and profile alignment features, and structural relations [30, 31]. Thus, the RF input vectors are of size 85 and correspond to each pair of proteins. As before, we used the RF output pairwise scores to perform template ranking for each query domain. We named these methods CNN-GRU-RF and CNN-BGRU-RF, depending on the recurrent network used for extracting the fold-related embeddings (unidirectional or bidirectional).

Finally, with the aim of including structural information from predicted contact maps, we also concatenated the cosine similarity score from the DeepFR method to our vector of pairwise scores (total size of 86). This extended score vector was then used to train a second random forest model and obtain an improved fold similarity score (Figure 4.1c, CNN-GRU-RF+ and CNN-BGRU-RF+ methods).

The random forest classifiers were trained with 500 decision trees, using the Python Scikit-learn package [58] implementation. As in [36], for the SCOP_TEST set the RF models were trained on the whole LINDAHL set, while we performed a 10-stage cross-validation to evaluate the LINDAHL and LINDAHL_1.75 test sets.

The performance metric used in all approaches is the fold recognition accuracy, which computes the ratio of samples that have been correctly classified (top 1 prediction). We also provide the ratio of finding the true fold in one of the five classes with the highest score values (named as top 5 prediction).

4.3 Results and discussion

The experimental accuracy results for the SCOP_TEST and LINDAHL test sets can be found in Table 4.2 and Table 4.3, respectively. We provide pairwise fold recognition results at the family, superfamily and fold levels, considering both the top 1 and top 5 predictions, as described in Section 4.2.5. In both test sets, we compare our CNN-GRU (CNN-GRU-RF, CNN-GRU-RF+) and CNN-BGRU (CNN-BGRU-RF, CNN-BGRU-RF+) based methods to several template-based and deep learning-based methods from the state-of-the-art that also address the protein fold recognition problem.

For the SCOP_TEST set (Table 4.2), our CNN-BGRU method obtained the best results at fold level, with accuracy values of 91.7% (33/36 hits) and 94.4% (34/36 hits) at top 1 and top 5 predictions, respectively. These results were closely followed by our CNN-BGRU-RF and CNN-BGRU-RF+ methods, both achieving 88.9% accuracy at the fold level (91.7% at top 5). The mentioned methods also provided good results at the family and superfamily levels. Indeed, our CNN-BGRU-RF+ method yielded the best accuracy values at superfamily level

Table 4.2: Pairwise fold recognition accuracy results at the family, superfamily and fold levels within the SCOP_TEST set.

Method	Family		Superfamily		Fold	
	Top 1	*Top 5*	*Top 1*	*Top 5*	*Top 1*	*Top 5*
RF-Fold [31]	93.4	98.1	83.9	91.1	55.6	72.2
DN-Fold [31]	94.3	97.2	82.1	91.1	61.1	86.1
RFDN-Fold [31]	93.4	97.2	82.1	91.1	61.1	86.1
MRFalign [36]	91.5	98.1	85.7	94.6	63.9	72.2
CEthreader [28]	84.0	95.3	73.2	89.3	80.6	91.7
DeepFR (s1) [36]	74.5	91.5	76.8	87.5	77.8	80.6
DeepFR (s2) [36]	75.5	93.4	78.6	83.9	86.1	88.9
DeepFRpro (s1) [36]	**95.3**	**99.1**	89.3	92.9	75.0	91.7
DeepFRpro (s2) [36]	94.3	96.2	87.5	94.6	83.3	88.9
CNN-GRU	84.9	95.3	80.4	87.5	72.2	86.1
CNN-BGRU	84.9	96.2	87.5	91.1	**91.7**	**94.4**
CNN-GRU-RF	93.4	96.2	87.5	**96.4**	75.0	83.3
CNN-BGRU-RF	91.5	95.3	87.5	92.9	88.9	91.7
CNN-GRU-RF+	93.4	96.2	89.3	94.6	83.3	88.9
CNN-BGRU-RF+	92.5	95.3	**92.9**	**96.4**	88.9	91.7

(92.9% and 96.4% at top 1 and top 5 predictions). Furthermore, as can be seen in Table 4.3, our CNN-BGRU-RF+ method outperformed all state-of-the-art methods at the fold level considering the LINDAHL test set. It achieves accuracy values of 76.3% (245/321 hits) and 85.7% (275/321 hits) at top 1 and top 5 predictions, respectively. This method was closely followed by our CNN-GRU-RF+ method, which also yielded the best results at the superfamily level (78.3% and 88.0% at top 1 and top 5 predictions).

In order to assess the raw performance of the fold-related embeddings extracted from our CNN-GRU and CNN-BGRU models, we can compare our results with those obtained from the DeepFR method [36] (both strategies 1 and 2). In this case, the metric used for embedding comparison is the cosine similarity distance. A notable difference is that our models use protein residue-level features at input, while the DeepFR model processes more informative structural features in the form of predicted contact maps. The evaluation results showed that our models performed better than the best DeepFR method (strategy 2). Specifically, our CNN-BGRU model stands out for its top 1 and top 5 predictions in both test sets (Tables 4.2 and 4.3). These results imply that our fold-related embedding vectors are suitable for template ranking and fold matching. Our models succeed in processing the input protein residue-level features (which include evolutionary and secondary structure information). The performance is comparable or even better to that obtained with image-form contact map features by the DeepFR method. This can be attributed to the use of recurrent layers that properly cope with the variable-length protein sequence, providing a well-suited fold-related embedding.

Table 4.3: Pairwise fold recognition accuracy results at the family, superfamily and fold levels within the LINDAHL test set.

Method	**Family**		**Superfamily**		**Fold**	
	Top 1	***Top 5***	***Top 1***	***Top 5***	***Top 1***	***Top 5***
PSI-BLAST [19]	71.2	72.3	27.4	27.9	4.0	4.7
HHpred [24]	82.9	87.1	58.0	70.0	25.2	39.4
RAPTOR [24]	**86.6**	89.3	56.3	69.0	38.2	58.7
BoostThreader [24]	86.5	90.5	66.1	76.4	42.6	57.4
SPARKS-X [25]	84.1	90.3	59.0	76.3	45.2	67.0
FOLDpro [31]	85.0	89.9	55.0	70.0	26.5	48.3
RF-Fold [31]	84.5	91.5	63.4	79.3	40.8	58.3
DN-Fold [31]	84.5	91.2	61.5	76.5	33.6	60.7
RFDN-Fold [31]	84.7	91.5	65.7	78.8	37.7	61.7
TA-fold [32]	85.2	––	74.2	––	53.9	––
MRFalign [36]	85.2	90.8	72.4	80.9	38.6	56.7
CEthreader [28]	76.6	87.2	69.4	81.8	52.3	70.4
DeepFR (s1) [36]	67.4	80.9	47.0	63.4	44.5	62.9
DeepFR (s2) [36]	65.4	83.4	51.4	67.1	56.1	70.1
DeepFRpro (s1) [36]	85.6	91.9	66.6	82.0	57.6	73.8
DeepFRpro (s2) [36]	83.1	92.3	69.6	82.5	66.0	78.8
MT-fold [33]	––	––	––	––	54.1	––
DeepSVM-fold [38]	––	––	––	––	67.3	––
MotifCNN-fold [39]	––	––	––	––	72.6	––
Fold-LTR-TCP [34]	––	––	––	––	73.2	––
CNN-GRU	68.6	89.2	56.2	77.4	56.7	74.1
CNN-BGRU	71.0	87.7	60.1	77.2	58.3	78.8
CNN-GRU-RF	84.9	94.1	74.4	**88.5**	63.9	81.3
CNN-BGRU-RF	85.6	93.0	72.4	86.9	65.4	82.2
CNN-GRU-RF+	84.5	**95.0**	**78.3**	88.0	73.2	**86.3**
CNN-BGRU-RF+	85.4	93.5	73.3	87.8	**76.3**	85.7

To further explore whether the extracted embeddings from our CNN-BGRU model are related to the fold classes, we run bi-clustering over a subset of SCOP_TEST with 46 protein domains covering 6 folds. These fold classes are $a.3$, $b.36$, $c.1$, $c.67$, $d.41$, and $f.4$, which provided better classification results in the SCOP_TEST dataset using the CNN-BGRU model. Figure 4.2 shows the resulting dendroheatmap, where each row corresponds to the 512-dimensional embedding extracted for each sample. As can be seen, protein domains in the same fold class present similar blocks (in red and green colors) in their embedding vectors. Also, the hierarchical clustering differentiates 6 clusters, each one grouping protein domains from the same fold together. We notice that folds $c.1$ and $c.67$ (from structural class c) cluster together at a higher level, indicating that the clustering is consistent with the SCOPe hierarchy. These findings suggest that the extracted embedding vectors are fold related.

We then evaluated the pairwise score obtained from our random forest classifier (using

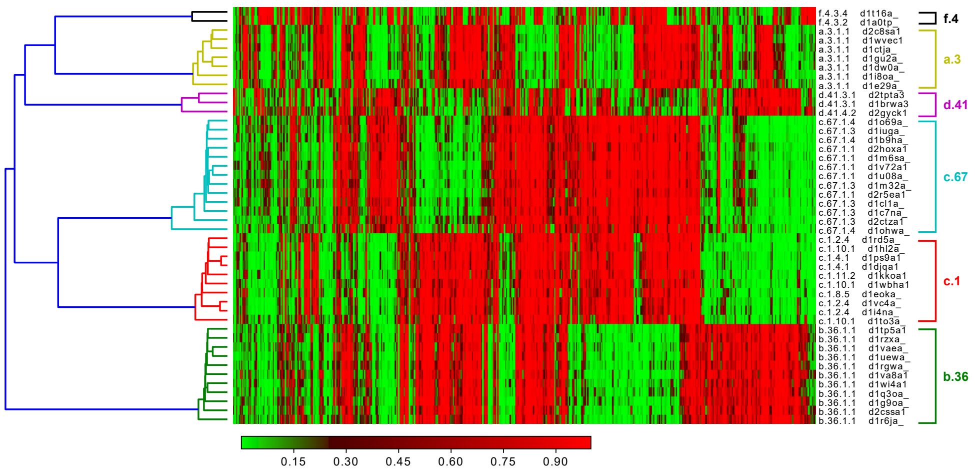


Figure 4.2: Dendroheatmap of the 512-dimensional fold-related embeddings extracted from the CNN-BGRU model. The analysis has been done by running bi-clustering over 46 protein domains (covering 6 folds) in the SCOP_TEST set. We computed the cosine distance between embedding vectors (rows) and embedding components (columns) separately. We then applied hierarchical clustering with single linkage to group similar vectors and components together. The embedding components are colored according to their values (lower values in green and higher values in red).

the 85-dimensional score vectors) in the fold recognition task. The results on the LINDAHL test set show an improvement of our advanced methods CNN-GRU-RF and CNN-BGRU-RF with respect to using our cosine similarity score alone. Accuracy results are competitive when compared to other previous template-based methods (as RAPTOR [23] and RF-Fold [30]) and ensemble methods (as TA-fold [32] and DeepFRpro [36]) at the three levels. Our methods also provide good results if we consider the top 5 predictions, outperforming the most similar method DeepFRpro (both s1 and s2) at the three levels. When evaluating the smaller SCOP_TEST set, we noticed a greater improvement from our CNN-GRU-RF method than the CNN-BGRU-RF version, likely due to its good results before the integration of different scores.

Our CNN-GRU-RF+ and CNN-BGRU-RF+ methods led to a further performance in both test sets, specially at the superfamily and fold levels. In this case, we integrate pairwise scores derived from both evolutionary information and predicted contact maps, so that our LINDAHL results can be fairly compared with those from the recent deep learning-based methods DeepSVM-fold [38] and MotifCNN-fold [39]. We can see that our CNN-GRU-RF+ and CNN-BGRU-RF+ methods performed better than these methods at the fold level. Moreover, our CNN-BGRU-RF+ method outperformed the best method from the state-of-the-art, Fold-LTR-TCP [34]. These results suggest that our fold-related embeddings contain complementary information to those of the DeepFR method (derived

Table 4.4: Pairwise fold recognition accuracy results at the family, superfamily and fold levels within the LINDAHL_1.75 test set.

Method	**Family**		**Superfamily**		**Fold**	
	Top 1	***Top 5***	***Top 1***	***Top 5***	***Top 1***	***Top 5***
DeepFR (s1)	69.7	82.3	43.4	60.6	40.2	59.0
DeepFR (s2)	72.2	85.6	46.6	64.3	50.8	67.1
DeepFRpro (s1)	88.3	**94.3**	71.9	82.8	56.5	74.2
DeepFRpro (s2)	87.0	93.8	71.9	82.6	63.5	77.2
CNN-GRU	70.2	88.7	55.7	77.0	57.9	77.0
CNN-BGRU	73.1	88.3	60.1	74.9	60.1	78.7
CNN-GRU-RF	87.9	93.6	74.9	87.7	69.1	80.6
CNN-BGRU-RF	87.9	93.1	71.7	87.0	66.3	83.4
CNN-GRU-RF+	**89.0**	**94.3**	**77.3**	**89.1**	**73.6**	84.0
CNN-BGRU-RF+	88.5	**94.3**	74.0	86.3	71.1	**86.8**

from predicted contact maps), confirming the importance of combining both sequential and structural information for the protein fold recognition task.

The pairwise fold recognition was also assessed using the proposed LINDAHL_1.75, whose results are shown in Table 4.4. In this case, the DeepFR and DeepFRpro methods (both strategies 1 and 2) were trained and evaluated using the code and guidelines provided by the authors in [36]. We also extracted the 84 pairwise similarity measures for the new LINDAHL_1.75 and trained the random forest models. As can be seen in Table 4.4, the performance of the deep learning methods on the updated version of LINDAHL is similar to the original one at the family, superfamily and fold levels. Our CNN-GRU-RF+ provided the best top 1 results at the three levels, with accuracy values of 89.0% (487/547 hits), 77.3% (333/431 hits) and 73.6% (262/356 hits), respectively. In addition to this, our CNN-BGRU-RF+ method performed the best at the fold level considering the top 5 predictions, yielding an accuracy of 86.8% (309/356 hits). As can be seen, the updated LINDAHL_1.75 test set proposed in this work could be considered as a valid replacement of the original LINDAHL in future protein fold recognition studies.

4.4 Conclusion

In this study, we have proposed a deep learning method to address the protein fold recognition problem. Our neural network architecture combines convolutional and recurrent (unidirectional and bidirectional) layers in order to seamlessly process arbitrary-length protein sequences. In addition to properly processing the input residue-level features of proteins of any length, our method is able to learn fixed-size fold-related embeddings to perform pairwise fold recognition through similarity distance between embedding vectors. The inclusion of recurrent layers yielded very competitive results in comparison with the state-of-the-art methods, such as DeepFR, DeepSVM-fold and MotifCNN-fold, which make use of more informative structural features through predicted contact maps. In general, the learned embeddings from our CNN-BGRU model provide better results than those of our CNN-GRU model. Performance variability could be explained by the high number of parameters to be trained in the bidirectional GRU layer, which almost doubles those of the unidirectional one. Our study confirmed that the integration of both sequential and structural features is beneficial for protein fold recognition. When combining our cosine similarity score with other pairwise similarity measures, including the DeepFR score, we noticed a significant performance improvement with respect to other state-of-the-art ensemble methods. Our CNN-BGRU-RF+ method yielded the best fold level result in the LINDAHL test set, outperforming the best method Fold-LTR-TCP. We should also highlight our remarkable top 5 results in all cases, meaning that the proposed method is suitable for fold matching and template ranking. Furthermore, we provided a curated version of the well-known LINDAHL dataset updated to the SCOP version 1.75 database, which can be used as a benchmark test set in future studies.

As future work, the fold-related embeddings could be used to efficiently find templates in the template-based modeling of protein structures. Also, the proposed neural network architectures could be extended to integrate and process other protein-level features along with the residue-level features used here. These models could be further tested in the direct fold classification task, training and evaluating them in the complete set of SCOPe folds.

Acknowledgments

This work was supported by the Spanish MINECO/FEDER Project TEC2016-80141-P and the associated FPI grant BES-2017-079792. The authors also acknowledge the support of NVIDIA Corporation with the donation of a Titan X GPU.

References

[1] C. Chothia and A. V. Finkelstein. The classification and origins of protein folding patterns. *Annual Review of Biochemistry*, 59(1):1007–1035, 1990.

[2] D. T. Jones, W. R. Taylor, and J. M. Thornton. A new approach to protein fold recognition. *Nature*, 358(6381):86, 1992.

[3] R. Kolodny, L. Pereyaslavets, A. O. Samson, and M. Levitt. On the universe of protein folds. *Annual Review of Biophysics*, 42:559–582, 2013.

[4] H. M. Berman, J. Westbrook, Z. Feng, et al. The protein data bank. *Nucleic Acids Research*, 28(1):235–242, 2000.

[5] C. Hadley and D. T. Jones. A systematic comparison of protein structure classifications: SCOP, CATH and FSSP. *Structure*, 7(9):1099–1112, 1999.

[6] A. G. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia. SCOP: A structural classification of proteins database for the investigation of sequences and structures. *journal of Molecular Biology*, 247(4):536–540, 1995.

[7] N. K. Fox, S. E. Brenner, and J.-M. Chandonia. SCOPe: Structural classification of proteins–extended, integrating SCOP and ASTRAL data and classification of new structures. *Nucleic Acids Research*, 42(D1):D304–D309, 2014.

[8] R. D. Schaeffer and V. Daggett. Protein folds and protein folding. *Protein Engineering, Design & Selection*, 24(1-2):11–19, 2010.

[9] L. Wei and Q. Zou. Recent progress in machine learning-based methods for protein fold recognition. *International journal of Molecular Sciences*, 17(12):2118, 2016.

[10] C. H. Q. Ding and I. Dubchak. Multi-class protein fold recognition using support vector machines and neural networks. *Bioinformatics*, 17(4):349–358, 2001.

[11] H.-B. Shen and K.-C. Chou. Ensemble classifier for protein fold pattern recognition. *Bioinformatics*, 22(14):1717–1722, 2006.

[12] Q. Dong, S. Zhou, and J. Guan. A new taxonomy-based protein fold recognition approach based on autocross-covariance transformation. *Bioinformatics*, 25(20):2655–2662, 2009.

[13] J.-Y. Yang and X. Chen. Improving taxonomy-based protein fold recognition by using global and local features. *Proteins: Structure, Function, and Bioinformatics*, 79(7):2053–2064, 2011.

[14] J. Lyons, A. Dehzangi, R. Heffernan, et al. Advancing the accuracy of protein fold recognition by utilizing profiles from hidden Markov models. *IEEE Transactions on Nanobioscience*, 14(7):761–772, 2015.

[15] D. Chen, X. Tian, B. Zhou, and J. Gao. ProFold: Protein fold classification with additional structural features and a novel ensemble classifier. *BioMed Research International*, 2016:1–10, 2016.

[16] Y. H. Taguchi and M. M. Gromiha. Application of amino acid occurrence for discriminating different folding types of globular proteins. *BMC Bioinformatics*, 8(1):404, 2007.

[17] J. Chen, M. Guo, X. Wang, and B. Liu. A comprehensive review and comparison of different computational methods for protein remote homology detection. *Briefings in Bioinformatics*, 19(2):231–244, 2018.

[18] M. S. Abual-Rub and R. Abdullah. A survey of protein fold recognition algorithms. *Journal of Computer Science*, 4(9):768–776, 2008.

[19] E. Lindahl and A. Elofsson. Identification of related proteins on family, superfamily and fold level. *Journal of Molecular Biology*, 295(3):613–625, 2000.

[20] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 1990.

[21] J. Söding. Protein homology detection by HMM–HMM comparison. *Bioinformatics*, 21(7):951–

960, 2005.

[22] J. Ma, S. Wang, Z. Wang, and J. Xu. MRFalign: Protein homology detection through alignment of Markov random fields. *PLoS Computational Biology*, 10(3):e1003500, 2014.

[23] J. Xu, M. Li, D. Kim, and Y. Xu. RAPTOR: optimal protein threading by linear programming. *Journal of Bioinformatics and Computational Biology*, 1(1):95–117, 2003.

[24] J. Peng and J. Xu. Boosting protein threading accuracy. In *Annual International Conference on Research in Computational Molecular Biology*, pages 31–45, 2009.

[25] Y. Yang, E. Faraggi, H. Zhao, and Y. Zhou. Improving protein fold recognition and template-based modeling by employing probabilistic-based matching between predicted one-dimensional structural properties of query and corresponding native properties of templates. *Bioinformatics*, 27(15):2076–2082, 2011.

[26] J. Ma, J. Peng, S. Wang, and J. Xu. A conditional neural fields model for protein threading. *Bioinformatics*, 28(12):i59–i66, 2012.

[27] J. A. Morales-Cordovilla, V. Sanchez, and M. Ratajczak. Protein alignment based on higher order conditional random fields for template-based modeling. *PLoS ONE*, 13(6):e0197912, 2018.

[28] W. Zheng, Q. Wuyun, Y. Li, et al. Detecting distant-homology protein structures by aligning deep neural-network based contact maps. *PLoS Computational Biology*, 15(10):1–27, 2019.

[29] J. Cheng and P. Baldi. A machine learning information retrieval approach to protein fold recognition. *Bioinformatics*, 22(12):1456–1463, 2006.

[30] T. Jo and J. Cheng. Improving protein fold recognition by random forest. *BMC Bioinformatics*, 15(11):S14, 2014.

[31] T. Jo, J. Hou, J. Eickholt, and J. Cheng. Improving protein fold recognition by deep learning networks. *Scientific Reports*, 5:17573, 2015.

[32] J. Xia, Z. Peng, D. Qi, H. Mu, and J. Yang. An ensemble approach to protein fold classification by integration of template-based assignment and support vector machine classifier. *Bioinformatics*, 33(6):863–870, 2016.

[33] K. Yan, X. Fang, Y. Xu, and B. Liu. Protein fold recognition based on multi-view modeling. *Bioinformatics*, 35(17):2982–2990, 2019.

[34] B. Liu, Y. Zhu, and K. Yan. Fold-LTR-TCP: protein fold recognition based on triadic closure principle. *Briefings in Bioinformatics*, 2019.

[35] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521:436–444, 2015.

[36] J. Zhu, H. Zhang, S. C. Li, et al. Improving protein fold recognition by extracting fold-specific features from predicted residue–residue contaacts. *Bioinformatics*, 33(23):3749–3757, 2017.

[37] J. Hou, B. Adhikari, and J. Cheng. DeepSF: deep convolutional neural network for mapping protein sequences to folds. *Bioinformatics*, 34(8):1295–1303, 2018.

[38] B. Liu, C.-C. Li, and K. Yan. DeepSVM-fold: protein fold recognition by combining support

vector machines and pairwise sequence similarity scores generated by deep learning networks. *Briefings in Bioinformatics*, 2019.

[39] C.-C. Li and B. Liu. MotifCNN-fold: protein fold recognition based on fold-specific features extracted by motif-based convolutional neural networks. *Briefings in Bioinformatics*, 21(6):2133–2141, 2020.

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

[41] A. Graves and N. Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *International Conference on Machine Learning*, pages 1764–1772, 2014.

[42] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112, 2014.

[43] S. Hochreiter, M. Heusel, and K. Obermayer. Fast model-based protein homology detection without alignment. *Bioinformatics*, 23(14):1728–1736, 2007.

[44] S. K. Sønderby and O. Winther. Protein secondary structure prediction with long short term memory networks. *arXiv preprint arXiv:1412.7828*, 2014.

[45] M. AlQuraishi. End-to-end differentiable learning of protein structure. *Cell Systems*, 8(4):292–301, 2019.

[46] S. K. Sønderby, C. K. Sønderby, H. Nielsen, and O. Winther. Convolutional LSTM networks for subcellular localization of proteins. In *International Conference on Algorithms for Computational Biology*, pages 68–80. Springer, 2015.

[47] J. Hanson, K. Paliwal, T. Litfin, Y. Yang, and Y. Zhou. Accurate prediction of protein contact maps by coupling residual two-dimensional bidirectional long short-term memory with convolutional neural networks. *Bioinformatics*, 34(23):4039–4045, 2018.

[48] B. Zhang, J. Li, and Q. Lü. Prediction of 8-state protein secondary structures by a novel deep learning architecture. *BMC Bioinformatics*, 19(1):293, 2018.

[49] J. Hanson, K. Paliwal, T. Litfin, Y. Yang, and Y. Zhou. Improving prediction of protein secondary structure, backbone angles, solvent accessibility and contact numbers by using predicted contact maps and an ensemble of recurrent and residual convolutional neural networks. *Bioinformatics*, 35(14):2403–2410, 2019.

[50] S. F. Altschul, T. L. Madden, A. A. Schäffer, et al. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402, 1997.

[51] C. N. Magnan and P. Baldi. SSpro/ACCpro 5: almost perfect prediction of protein secondary structure and relative solvent accessibility using profiles, machine learning and structural similarity. *Bioinformatics*, 30(18):2592–2597, 2014.

[52] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *International Conference on Machine Learning*, pages 448–456, 2015.

[53] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[54] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[55] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[56] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[57] A. Paszke, S. Gross, S. Chintala, et al. Automatic differentiation in PyTorch. In *Advances in Neural Information Processing Systems*, 2017.

[58] F. Pedregosa, G. Varoquaux, A. Gramfort, et al. Scikit-learn: machine learning in Python. *journal of Machine Learning Research*, 12:2825–2830, 2011.

4.S Supplementary material

4.S.1 Cross-validation sets and training results

Table 4.S1: Number of protein domains, families, superfamilies and folds in each structural class for the SCOPe 2.06 training dataset.

Structural class	Protein domains	Family classes	Superfamily classes	Fold classes
a	2684	943	480	273
b	3603	858	343	168
c	4739	900	233	144
d	3882	1193	528	368
e	319	100	65	65
f	296	137	98	56
g	610	213	115	80
Total	16133	4344	1862	1154

Table 4.S2: Cross-validation subsets for the SCOPe 2.06 training dataset (family separation).

Subset	Protein domains		Family classes		Superfamily classes		Fold classes	
	Train	Valid	Train	Valid	Train	Valid	Train	Valid
CV1	12859	3274	3474	870	1641	615	1015	442
CV2	12895	3238	3474	870	1642	606	1032	429
CV3	12794	3339	3474	870	1633	630	1030	436
CV4	13023	3110	3474	870	1646	611	1035	436
CV5	12961	3172	3480	864	1626	632	1035	436

Table 4.S3: SCOPe 2.06 cross-validation accuracy results in training and validation sets (epoch 80).

Model	CNN-GRU				CNN-BGRU			
Subset	CV-Train		CV-Val		CV-Train		CV-Val	
	Top 1	Top 5	Top 1	Top 5	Top 1	Top 5	Top 1	Top 5
CV1	98.99	99.94	68.23	79.23	99.70	100.00	70.22	81.06
CV2	99.40	99.99	66.21	76.78	99.89	100.00	68.41	77.79
CV3	99.34	99.99	67.95	79.19	99.84	100.00	71.24	80.26
CV4	98.83	99.98	64.98	78.75	99.80	100.00	69.68	79.16
CV5	99.55	100.00	67.37	76.64	99.88	100.00	69.64	79.73

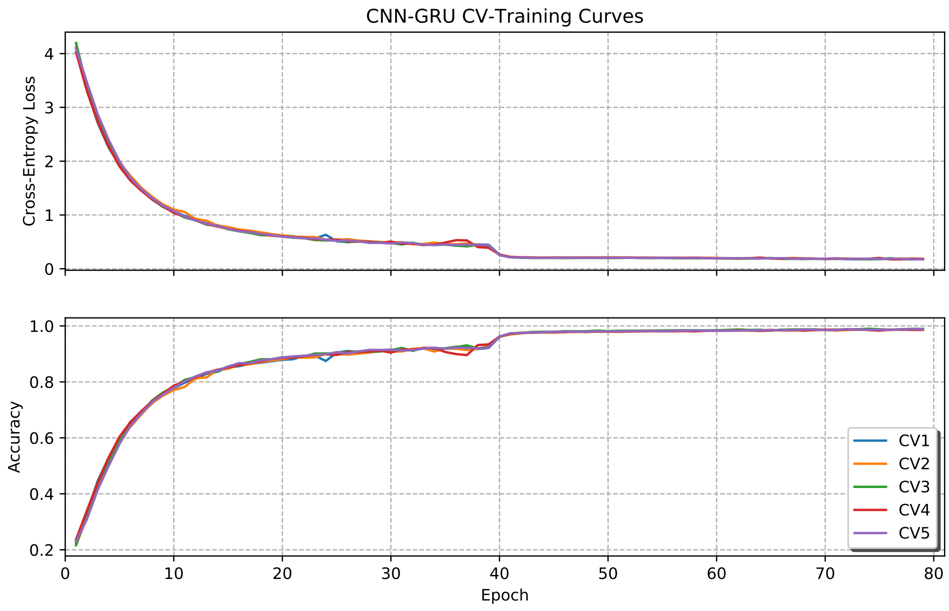


Figure 4.S1: Cross-validation: CNN-GRU model training loss and accuracy.

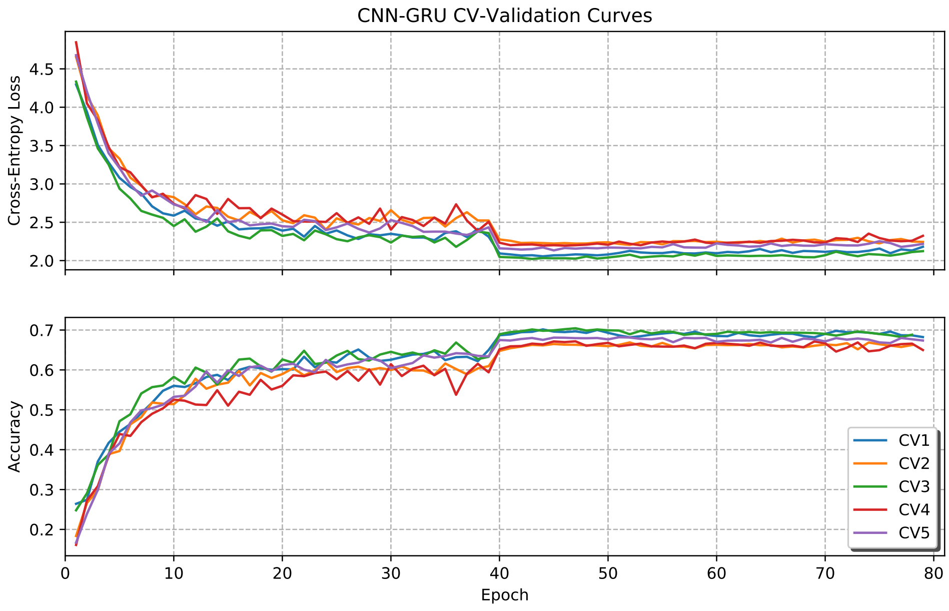


Figure 4.S2: Cross-validation: CNN-GRU model validation loss and accuracy.

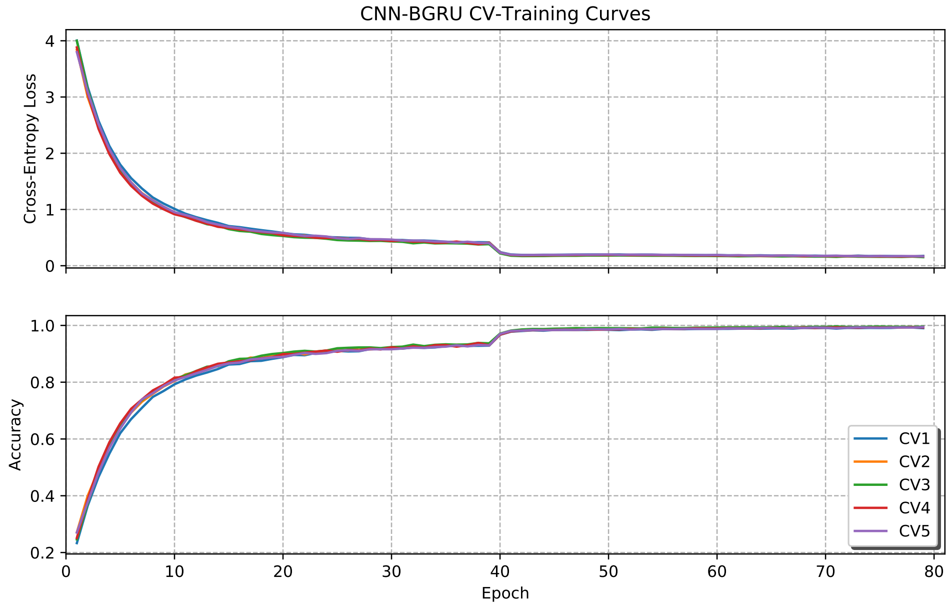


Figure 4.S3: Cross-validation: CNN-BGRU model training loss and accuracy.

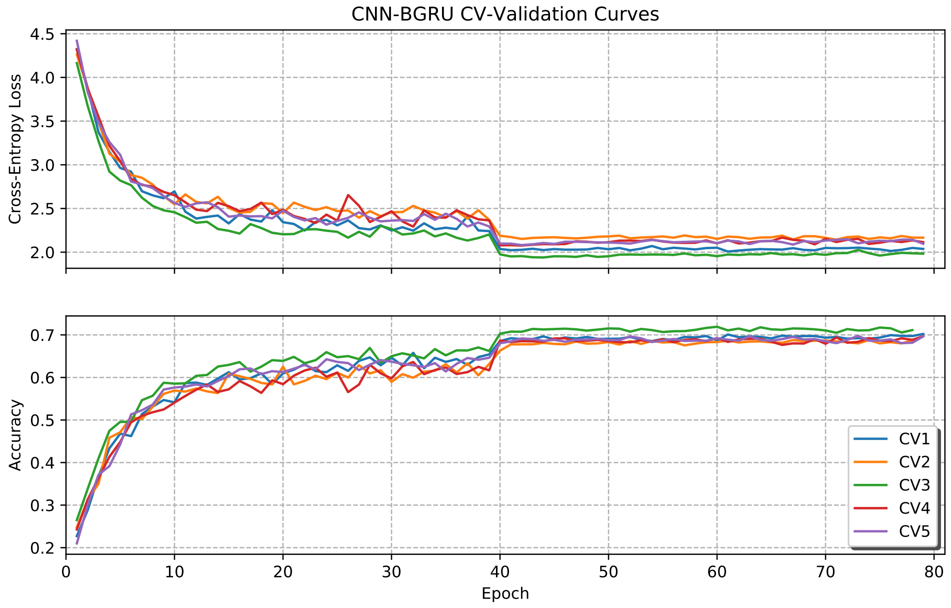


Figure 4.S4: Cross-validation: CNN-BGRU model validation loss and accuracy.

4.S.2 Lindahl dataset update (SCOP 1.37 to SCOP 1.75)

Table 4.S4: Original and new Lindahl identifiers and labels.

Original id	Original label	New id	New label
Changes in PDB names			
d1alo_1	1_47_1_1	d1vlba1	a.56.1.1
d1alo_3	4_23_1_1	d1vlba3	d.41.1.1
d1alo_4	4_77_1_1	d1vlba4	d.133.1.1
d1cxsa1	2_35_2_1	d1eu1a1	b.52.2.2
d1etd	1_4_3_9	d1k78b_	a.4.5.21
d1ezm_2	4_50_1_2	d1ezma_	d.92.1.2
d1gdoa	4_88_1_1	d1xffa_	d.153.1.1
d1kst	7_17_1_1	d1n4ya_	g.20.1.1
d1lefa	1_20_1_1	d2lefa_	a.21.1.1
d1lpt	1_42_1_1	d1gh1a_	a.52.1.1
d1tsg	4_97_1_3	d1o7bt_	d.169.1.4
d1tssa1	2_26_2_2	d2tssa1	b.40.2.2
d1tssa2	4_12_5_1	d2tssa2	d.15.6.1
d1wsyb	3_59_1_1	d1bksb_	c.79.1.1
d2blta	5_4_1_1	d1xx2a_	e.3.1.1
d2fxb	4_33_1_4	d1iqza_	d.58.1.4
d2stv	2_8_1_2	d2buka1	b.121.7.1
d2wrpr	1_78_1_1	d2oz9r1	a.4.12.1
d3b5c	4_66_1_1	d1cyoa_	d.120.1.1
Deleted domains			
d1alo_5	4_77_1_1	————	————
d1alo_6	4_77_1_1	————	————
d1alo_7	4_77_1_1	————	————
d1ezm_1	1_53_1_1	————	————
d1gal_2	4_13_1_4	————	————
Added domains			
————	————	d1v97a5	d.133.1.1
————	————	d1n62b2	d.133.1.1
————	————	d1t3qb2	d.133.1.1
————	————	d1ffya1	a.27.1.1
————	————	d1c0pa2	d.16.1.3
Decisions over split domains			
d1aky	3_25_1_1	d1akya1	c.37.1.1
d1div	4_82_1_1	d1diva1	d.99.1.1
d1dkza	5_17_1_1	d1dkza2	b.130.1.1
d1knya	5_10_1_2	d1knya2	d.218.1.1
d1rpl	5_10_1_1	d1rpla2	d.218.1.2
d1zyma	3_5_1_2	d1zyma2	c.8.1.2

4.S.3 De novo training of the DeepFR method

Image-form feature extraction command pipeline

For each protein domain in the dataset:

- Construct multiple sequence alignment (MSA) using the uniprot20_2016_02 HHM database (hhblits from HHsuite version 3.0.3):

```
hhblits -i ${id}.fasta -d uniprot20_2016_02 -n 5 \
-oa3m ${id}.a3m -o ${id}.hhr
```

- Convert a3m file to aln format:

```
awk ‘NR % 2 = = 0’ ${id}.a3m | sed ‘s/[a-z]//g’ > ${id}.aln
```

- Get predicted contact map matrix using CCMpred on GPU:

```
ccmpred -R ${id}.aln ${id}.ccm
```

- Create image file from the contact map matrix by using the following python script:

```
python convert_ccmpred_to_image.py ${id}.ccm ${id}.ccm.png
.......................................................
import sys
import matplotlib.pyplot as plt
import numpy as np

ccm_mat, image_name = sys.argv[1:]
matrix = np.loadtxt(ccm_mat)
plt.imsave(image_name, matrix, cmap=plt.cm.gray_r)
```

2D convolutional neural network architecture and training hyperparameters

- Common hyperparameters

– Input features size:	256x256x1
– Number of 2D convolutional layers:	5
– Max-pooling after layers:	1-2-5
– Embedding size:	1024
– Optimizer:	SGD with momentum 0.9
– Initial learning rate:	0.001
– Learning rate reduction step:	100k
– Learning rate reduction factor:	10
– Weight decay value:	0.0005

- Strategy 1 (reshaping) hyperparameters

– Mini-batch size:	32
– Number of training iterations:	500k

- Strategy 2 (sampling/padding) hyperparameters

– Mini-batch size:	128
– Number of training iterations:	250k

4.S.4 GRU-based recurrent layer

In the proposed neural network architecture, the outputs of the last convolutional layer are fed to the GRU-based recurrent layer (GRU stands for gated recurrent unit). The GRU cell computes the following function:

- Reset gate: $\mathbf{r}_t = \sigma(\mathbf{W}_{ir}\mathbf{x}_t + \mathbf{b}_{ir} + \mathbf{W}_{hr}\mathbf{h}_{(t-1)} + \mathbf{b}_{hr})$
- Update gate: $\mathbf{z}_t = \sigma(\mathbf{W}_{iz}\mathbf{x}_t + \mathbf{b}_{iz} + \mathbf{W}_{hz}\mathbf{h}_{(t-1)} + \mathbf{b}_{hz})$
- New gate: $\mathbf{n}_t = \tanh(\mathbf{W}_{in}\mathbf{x}_t + \mathbf{b}_{in} + \mathbf{r}_t * (\mathbf{W}_{hn}\mathbf{h}_{(t-1)} + \mathbf{b}_{hn}))$
- New hidden state: $\mathbf{h}_t = (1 - \mathbf{z}_t) * \mathbf{n}_t + \mathbf{z}_t * \mathbf{h}_{(t-1)}$

where $\mathbf{x}_t$ and $\mathbf{h}_t$ are the feature and hidden state vectors associated to the amino acid t in the protein sequence ($t = 1, \ldots, L$), $\mathbf{W}_{xy}$ and $\mathbf{b}_{xy}$ are the weight matrices and bias vectors learned in the GRU cell, $\sigma()$ and tanh() refer to the sigmoid and hyperbolic tangent activation functions, respectively. At each amino acid t, the reset gate ($\mathbf{r}_t$) decides which information to forget from the previous amino acid ($t-1$), while the update gate ($\mathbf{z}_t$) decides what information to throw away (previous amino acid) and what new information to add (current amino acid).

4.S.5 Random forest models

Python code for random forest classifier training and evaluation using the Scikit-learn package, where *X_train* and *X_test* are the pairwise score vectors (85 or 86-dimensional) and *y_train* contains the true training labels (1 if the pair of proteins belong to the same fold, -1 otherwise). The *y_prob* vector includes an improved pairwise score for each pair of proteins in the test set.

```
from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier(n_estimators=500, random_state=0)
clf.fit(X_train, y_train)
y_prob = clf.predict_proba(X_test)
```

4.S.6 Fold-related embeddings analysis using bi-clustering

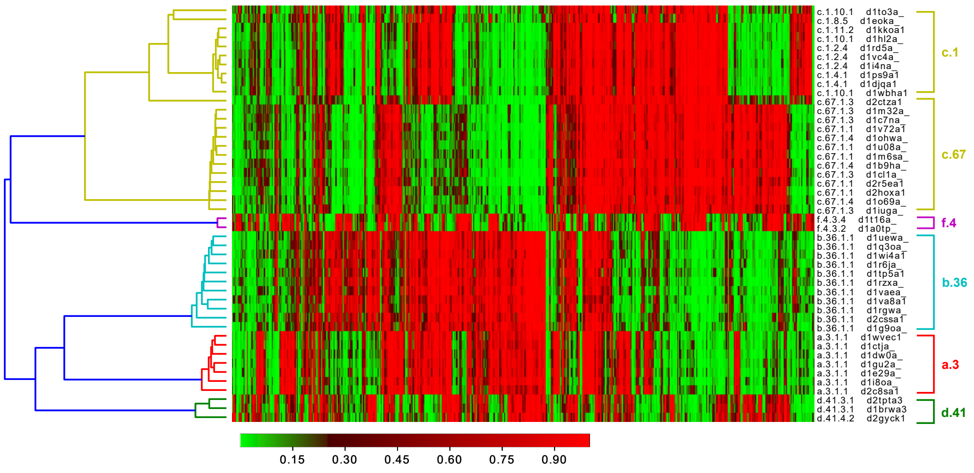


Figure 4.S5: Dendroheatmap of the 512-dimensional fold-related embeddings extracted from the CNN-GRU model. The analysis has been done by running bi-clustering over 46 protein domains (covering 6 folds) in the SCOP_TEST set.

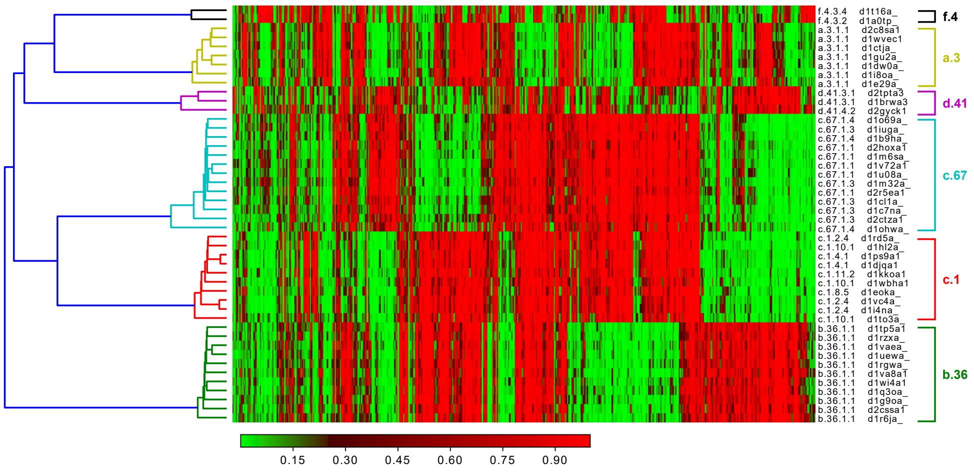


Figure 4.S6: Dendroheatmap of the 512-dimensional fold-related embeddings extracted from the CNN-BGRU model. The analysis has been done by running bi-clustering over 46 protein domains (covering 6 folds) in the SCOP_TEST set.

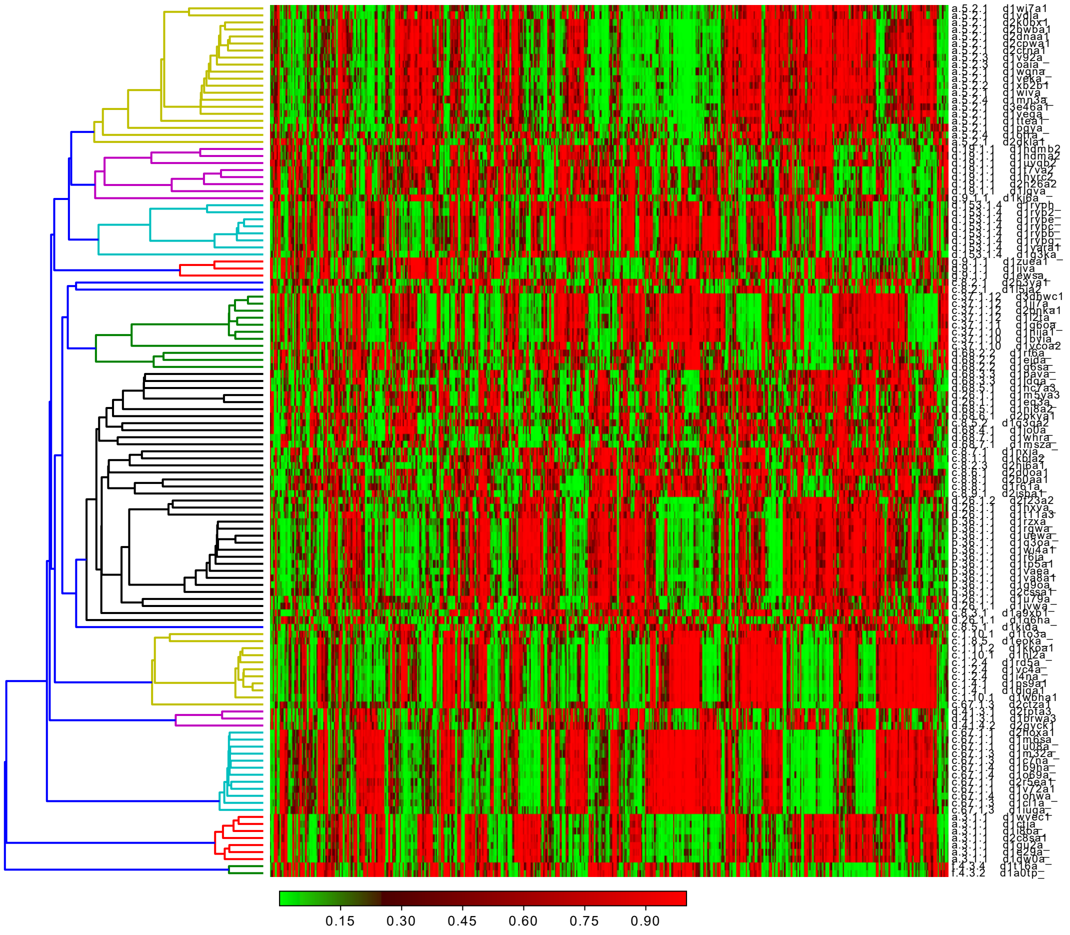


Figure 4.S7: Dendroheatmap of the 512-dimensional fold-related embeddings extracted from the CNN-GRU model. The analysis has been done by running bi-clustering over all 124 protein domains (covering 14 folds) in the SCOP_TEST set.

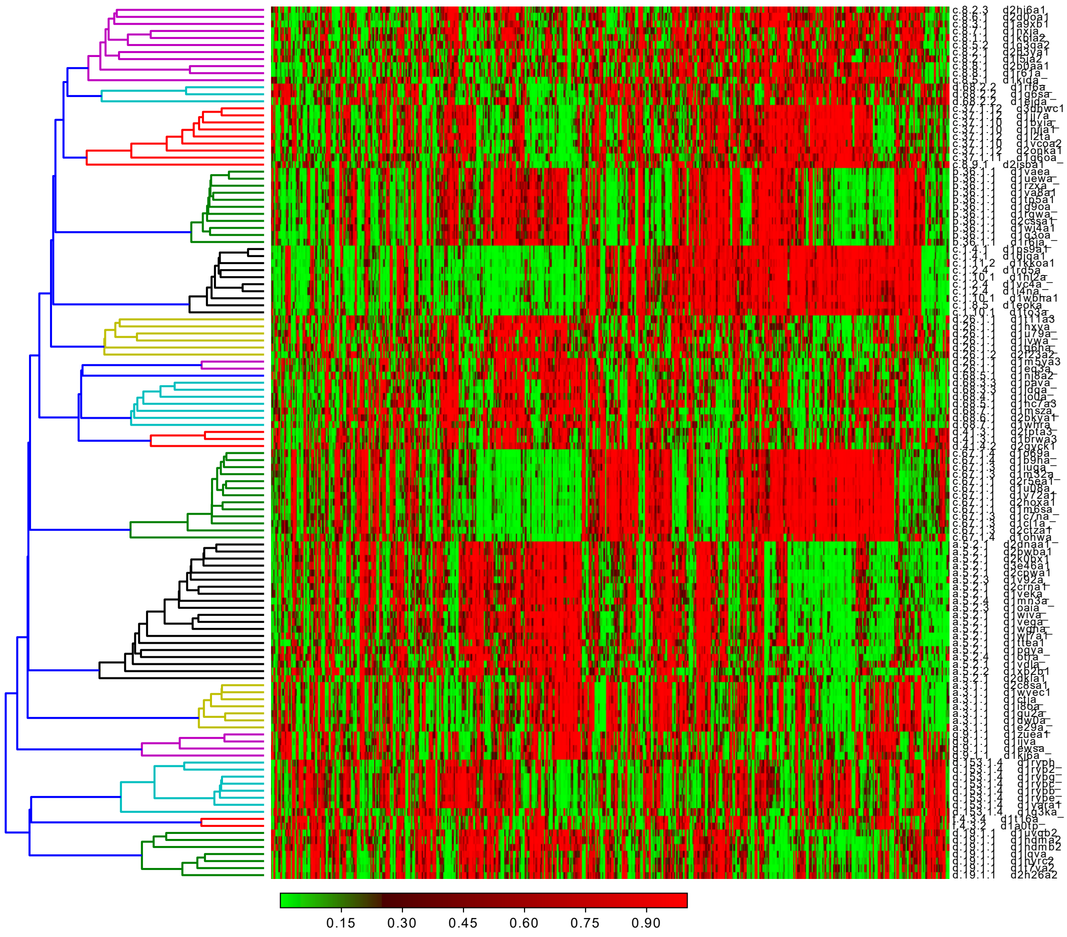


Figure 4.S8: Dendroheatmap of the 512-dimensional fold-related embeddings extracted from the CNN-BGRU model. The analysis has been done by running bi-clustering over all 124 protein domains (covering 14 folds) in the SCOP_TEST set.

5

FoldHSphere: deep hyperspherical embeddings for protein fold recognition

Amelia Villegas-Morcillo, Victoria Sanchez, and Angel M. Gomez

This chapter has been published as follows: A. Villegas-Morcillo, V. Sanchez, and A.M. Gomez. FoldHSphere: deep hyperspherical embeddings for protein fold recognition. *BMC Bioinformatics*, 22(1):1-21, 2021. DOI: 10.1186/s12859-021-04419-7.

- Status: Received in May 2021. Accepted in September 2021. Published in October 2021.
- Impact Factor (JCR 2021): 3.328. Mathematical & Computational Biology. Rank 20/57 (Q2).

Abstract

***Background**: Current state-of-the-art deep learning approaches for protein fold recognition learn protein embeddings that improve prediction performance at the fold level. However, there still exists a performance gap at the fold level and the (relatively easier) family level, suggesting that it might be possible to learn an embedding space that better represents the protein folds.*

***Results**: In this paper, we propose the FoldHSphere method to learn a better fold embedding space through a two-stage training procedure. We first obtain prototype vectors for each fold class that are maximally separated in hyperspherical space. We then train a neural network by minimizing the angular large margin cosine loss (LMCL) to learn protein embeddings clustered around the corresponding hyperspherical fold prototypes. Our network architectures, ResCNN-GRU and ResCNN-BGRU, process the input protein sequences by applying several residual-convolutional blocks followed by a gated recurrent unit-based recurrent layer. Evaluation results on the LINDAHL dataset indicate that the use of our hyperspherical embeddings effectively bridges the performance gap at the family and fold levels. Furthermore, our FoldHSpherePro ensemble method yields an accuracy of* 81.3% *at the fold level, outperforming all the state-of-the-art methods.*

***Conclusions**: Our methodology is efficient in learning discriminative and fold-representative embeddings for the protein domains. The proposed hyperspherical embeddings are effective at identifying the protein fold class by pairwise comparison, even when amino acid sequence similarities are low.*

***Keywords**: Protein Fold Recognition; Deep Neural Networks; Residual Convolutions; Embedding Learning; Hyperspherical Space; Thomson Problem*

5.1 Background

Protein structure prediction given the amino acid sequence is a challenging problem in structural bioinformatics. One of the key steps in the template-based modelling (TBM) of protein structures is the recognition of the protein fold [1–5]. The goal is to predict the fold type of a protein domain by comparison with template structures from the Protein Data Bank (PDB) [6]. Solved structure domains from the PDB are classified into several levels according to structural and sequence similarities in databases as SCOP [7, 8] and CATH [9]. The objective here is to identify proteins sharing the same *fold class*—with similar arrangement of structural elements but differing in the amino acid sequence.

Early computational approaches to recognizing proteins with similar structure and sequence (homology modelling) were based on sequence-to-sequence (BLAST [10]) or profile-to-profile (HHpred [11]) alignments, as well as Markov random fields (MRFAlign

[12]). In addition, threading methods aim to recognize distant-homologous proteins with low similarity in sequence by using structural properties instead. These methods include RAPTOR [13], BoostThreader [14], SPARKS-X [15], conditional random field-based CNF-pred [16] and [17], and more recently the EigenTHREADER [18] and CEthreader [19] methods, which use predicted contact map information.

In general, the protein fold recognition methods as the ones described above are derived from the template-based structure prediction problem. Unlike these, in the taxonomy-based fold classification approaches [20] the protein sequences are directly mapped into fold classes. To this end, machine learning approaches such as FP-Pred [21], ACCFold [22], TAXFOLD [23], [24, 25], HMMFold [26], ProFold [27], and DKELM-LDA [28], as well as the deep learning methods Conv-SXGbg-DeepFold [29] and DeepFrag-k [30], have been proposed to successfully classify into a pre-defined group of SCOP fold classes. However, the evaluated folds comprise a small set including only those folds with a higher amount of protein domains (27 or 30 folds), in contrast to the more than 1000 existing fold classes in the SCOP database.

Several machine learning algorithms have been also introduced for the protein fold recognition task [31]. First attempts treated the task as a binary classification problem to decide whether two protein domains belonged to the same fold. Different techniques were applied here, such as support vector machines (FOLDpro [32]), random forests (RF-Fold [33]) and neural networks (DN-Fold [34]). Moreover, ensemble methods enhance the recognition performance by combining multiple protein feature representations and prediction techniques. Examples are TA-Fold [35] and the multi-view learning ensemble frameworks MT-fold [36], EMfold [37] and MLDH-Fold [38]. On the other hand, the learning to rank methods, such as Fold-LTR-TCP [39], FoldRec-C2C [40], and ProtFold-DFG [41], treat the problem as an information retrieval task and try to learn the relationship among proteins in the datasets.

Furthermore, deep learning-based methods have been recently proposed to identify the protein fold, such as DeepSF [42], DeepFR [43], DeepSVM-Fold [44], MotifCNN-fold [45], SelfAT-Fold [46], VGGfold [47], and CNN-BGRU [48]. In these methods, a supervised neural network model is trained to classify the input protein domain into one of the possible fold classes. From the trained model, a fold-related embedding representation is extracted, which is then used to measure the similarity between each two protein domains. In this context, the learned embeddings constitute a d-dimensional space in which we can map high-dimensional protein representations such as evolutionary profiles [48] ($L \times 20$, where L is the protein sequence length) or contact maps [43] ($L \times L$). Moreover, these embeddings capture the fold information during training by placing inputs from the same fold close together in the embedding space. The model architecture for protein fold recognition usually contains a convolutional neural network (CNN) alone or in combination with recurrent layers—long-short term memory (LSTM) [49] or gated recurrent unit (GRU) [50] cells—or self-attention layers [51]. Hence, in preceding works, most of the effort has been put into improving the neural network architectures and making them suitable to process different protein representations, such as predicted contact maps, evolutionary profiles or predicted secondary structure elements. In this work, we propose two architectures formed by several blocks of residual-convolutions [52] and a recurrent layer, which we name ResCNN-GRU and ResCNN-BGRU. Here, the suffix 'BGRU' refers to bidirectional GRU, while 'GRU' indicates the use of a unidirectional GRU. These two architectures are derived from our previous CNN-GRU and CNN-BGRU models [48], and can also process arbitrary length protein sequences represented by residue-level features.

However, unlike previous deep learning approaches, our main interest here is to improve the fold-related embedding vectors by modifying the neural network optimization criterion. While the softmax cross-entropy loss is commonly used for multi-class classification problems, it lacks sufficient discriminative power for classification [53–55]. In this regard, modifications on the loss function have been introduced, leading to improved functions such as the center loss [53], the large margin softmax (L-Softmax) loss [54], and the angular softmax (A-Softmax) loss [55]. Thus, in this work, we propose to minimize an angular-based loss function, namely the *large margin cosine loss* (LMCL) [56]. LMCL removes any vector norm dependencies by normalizing the input embedding and class weight vectors in the classification layer and therefore distributes them angularly on a high-dimensional sphere—or *hypersphere*. The function also introduces a class boundary margin to enlarge the inter-class angular separation while reducing the intra-class separation for embeddings within the same fold class.

We further improve the training of our neural network model by minimizing the LMCL with a fixed weight matrix in the last classification layer. Such a matrix contains a pre-defined set fold class vectors—*hyperspherical prototypes*—that are maximally separated on the surface of a hypersphere. To ensure maximum angular separation between prototypes, we draw inspiration from the well-known Thomson problem [57]. Its goal is to determine the minimum energy configuration of K charged particles on the surface of a unit sphere. By minimizing a Thomson-based loss function, extended to a hypersphere of arbitrary number of dimensions, we optimize the angular distribution of our prototype vectors. Here we pre-train the prototype matrix separately and keep it fixed during the optimization of our neural network model. It must be noted that, unlike conventional transfer learning procedures in which the last layers of the network are fine-tuned, we pre-define the output embedding space given by a set of fold prototypes representing the cluster centroids for each fold class [58]. In this way, during training, the model is forced to learn protein embeddings clustered around the corresponding hyperspherical fold prototypes.

In summary, our main contribution is a training procedure that provides hyperspher-

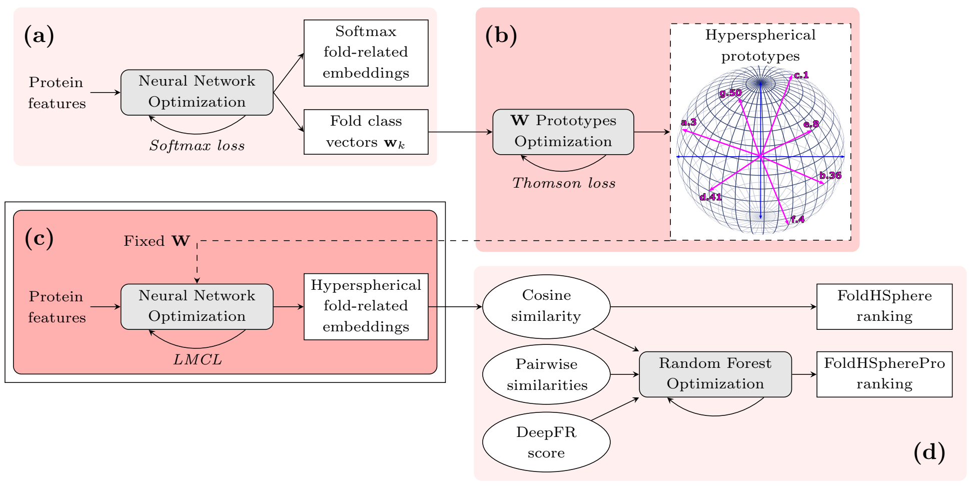

(a)
Protein features
Neural Network Optimization
Softmax loss
Softmax fold-related embeddings
Fold class vectors $\mathbf{w}_k$
(b)
W Prototypes Optimization
Thomson loss
Hyperspherical prototypes
c.1
g.50
a.3
e.8
b.36
d.41
f.4
(c)
Fixed W
Protein features
Neural Network Optimization
LMCL
Hyperspherical fold-related embeddings
Cosine similarity
Pairwise similarities
DeepFR score
Random Forest Optimization
FoldHSphere ranking
FoldHSpherePro ranking
(d)

Figure 5.1: Overview of the FoldHSphere approach for protein fold recognition. In the first stage **(a)** we train a neural network model to map the protein domains into K fold classes using the softmax cross-entropy as loss function. From this trained model, we extract fold class weight vectors $\mathbf{w}_k, k = 1, \ldots, K$ learned in the last classification layer. **(b)** We then optimize the position of the $\mathbf{w}_k$ vectors by our proposed Thomson-based loss, so that they are maximally separated in the angular space. **(c)** The resulting hyperspherical prototypes are used as a fixed non-trainable classification matrix $\mathbf{W}$ in the last layer of the neural network model, which is trained again, but now minimizing the LMCL. The final hyperspherical embeddings are extracted from the fully-connected part of this model. **(d)** Finally, the cosine similarity is computed between each two embeddings and a template ranking is performed for each query protein domain (FoldHSphere method). Moreover, template ranking is further improved by using enhanced scores provided by a random forest model trained with additional similarity measures as inputs (FoldHSpherePro method).

ical protein embeddings, learned by minimizing the angular LMCL around pre-defined prototypes for the fold classes in a hyperspherical space. We obtain these embeddings by training the ResCNN-GRU and ResCNN-BGRU architectures that are effective at processing arbitrary length protein sequences. An overview of our approach is depicted in Figure 5.1. Our proposed methods, named FoldHSphere and FoldHSpherePro, significantly advance the state-of-the-art performance on well-known benchmark datasets.

5.2 Materials and methods

5.2.1 Datasets

The datasets were obtained from the public protein databases SCOP [7] and the extended SCOPe [8]. These databases contain a hierarchical structural classification of protein domains with solved structure. From the top-down view, such hierarchical levels are *structural class, fold, superfamily* and *family*, which group protein domains with increasing sequence similarity at each level.

Training dataset

We trained our neural network models using the SCOPe 2.06 training set from [43]. Such a training set was obtained after filtering out protein domains having a significant sequence similarity to those in the test set. To do so, the following similarity reduction methods were executed: MMseqs2 [59] (sequence identity 25%, e-value 10^{-4}), CD-HIT-2D [60] (sequence identity 40%) and BLAST+ [61] (e-value 10^{-4}). The final dataset contains 16133 protein domains sharing at most 95% pairwise sequence identity, which are classified into $K = 1154$ folds. For hyperparameter tuning, we performed a 5-stage cross-validation over the entire training set. Hence, we split the 16133 protein domains into 5 groups, including domains from different families in each one (Supplementary Material 5.S.1). This prevents having proteins in the validation subsets with similar amino acid sequence to those in the corresponding training subset.

Benchmark datasets

We tested the effectiveness of our hyperspherical embeddings using both the well-known LINDAHL dataset [3] and the updated LINDAHL_1.75 dataset we recently proposed in [48]. The original LINDAHL dataset includes 976 domains from SCOP 1.37 covering 330 folds. Updated to SCOP 1.75, the LINDAHL_1.75 dataset contains the same number of proteins (976) but now classified into 323 folds. Protein domains within both test sets share a maximum sequence identity of 40%, as well as with respect to the training domains. Each dataset is paired and evaluated independently at three different levels—*family, superfamily* and *fold*. Thus, while the number of individual protein domains evaluated within the LINDAHL dataset are 555, 434 and 321 for the family, superfamily and fold levels, in LINDAHL_1.75 we evaluate 547, 431 and 356 domains, respectively.

5.2.2 Protein residue-level feature representation

In order to represent the protein amino acid sequence with variable length L, we considered 45 features for each amino acid as in previous works [42, 48]. These 45 residue-level features contain the following information:

- *Amino acid encoding:* one-hot vector of size 20 representing the amino acid type.
- *Position-specific scoring matrix (PSSM):* 20 elements which contain the evolutionary profile information obtained from the multiple sequence alignment (MSA). We computed the PSSM matrix using PSI-BLAST [10] and the non-redundant database 'nr90' for sequence homology searching.
- *Secondary structure:* one-hot vector of size 3 encoding the helix, strand and loop secondary structure elements. To predict the secondary structure we used the SSpro method from the SCRATCH suite [62].

- *Solvent accessibility:* one-hot vector of size 2 encoding the exposed and buried states. Similar to before, we used the ACCpro method from SCRATCH to predict the solvent accessibility states.

These $L \times 45$ features are used as input to our neural network models, which are trained to predict the fold class for each protein domain.

5.2.3 Residual-convolutional and recurrent neural network

In this study, we improve our previously proposed neural network models, CNN-GRU and CNN-BGRU [48], with blocks of residual convolutions [52]. As a result, the model architecture is formed by three main parts, as depicted in Figure 5.2: residual-convolutional (ResCNN), recurrent (RNN) and fully-connected (FC). We named these new models as ResCNN-GRU and ResCNN-BGRU, depending on the use of unidirectional or bidirectional layers of gated recurrent units (GRU) in the recurrent part.

Residual-convolutional part

The convolutional neural network (CNN) aims to capture the local context of each residue in the protein domain and discover short-term patterns within the amino acid sequence. At each CNN layer, we apply a 1D-convolution operation along the sequence dimension, with several convolutional filters of specific length to be learned. Considering an input of size $L \times 45$, the output of each 1D-convolutional layer is of size $L \times N_l$, where N_l is the number of learned filters in the l-th layer. In our model, the 1D-convolutional layers are grouped into residual blocks [52]. The output $\mathcal{R}(x_b, \mathcal{W}_b)$ of each residual block is combined with its input x_b as $x_{b+1} = x_b + \mathcal{R}(x_b, \mathcal{W}_b)$, where $\mathcal{W}_b$ are the weights and biases associated to the b-th residual block, and $\mathcal{R}(\cdot)$ is the mapping function performed by the block.

Figure 5.2a presents the ResCNN part of our model. We first apply an initial 1D-convolution to transform the $L \times 45$ input features into $L \times 256$ outputs by using 256 filters of length 1. These are then processed by two residual blocks, each one formed by two layers with 64 and 256 filters of length 5. After each convolution, ReLU activation and Batch-Normalization [63] are applied.

Recurrent part

The purpose of the recurrent neural network (RNN) is to exploit long-distance relations through all the amino acid sequence and generate a summary of the whole protein domain at its output. Here, the $L \times 256$ outputs from the ResCNN are fed into a gated recurrent unit (GRU) [50] based layer with 1024 state units.

As shown in Figure 5.2b, instead of saving all the $L \times 1024$ states of the GRU, we only consider the last state ($\overrightarrow{\mathbf{h}_L}$) as a summary vector of 1024 elements. In this way, our model architecture can process amino acid sequences of arbitrary length and extract a fixed-size

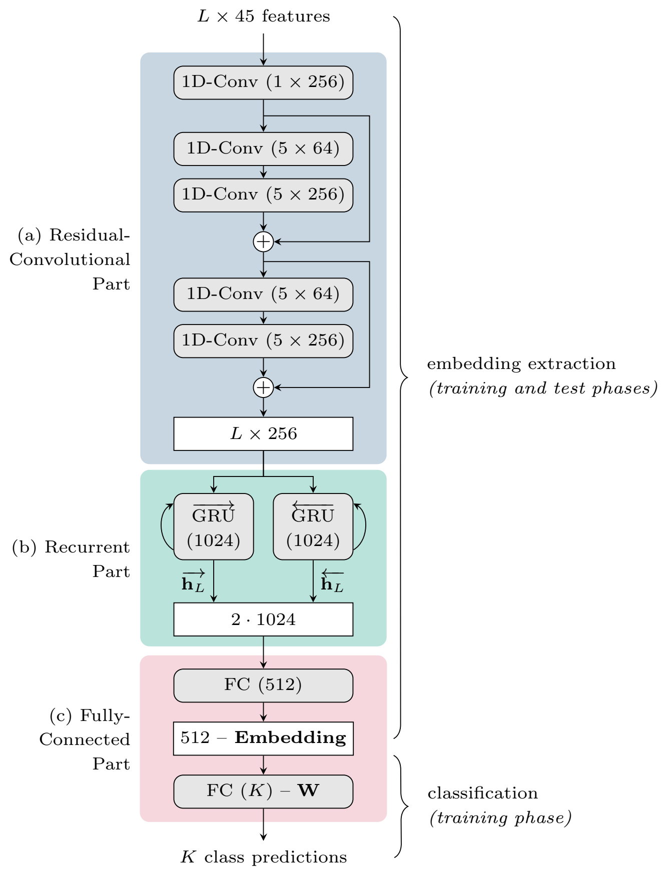


Figure 5.2: The proposed ResCNN-BGRU neural network model for fold-related embedding learning through protein fold classification. The model architecture contains three differentiated parts. The residual-convolutional network **(a)** processes the input $L \times 45$ residue-level features and consists of two residual blocks with two 1D-convolutional layers each. Its output is passed through a bidirectional layer of gated recurrent units **(b)** to obtain a fixed size representation of the input domain, which is further processed by two fully-connected layers **(c)**. The first FC layer learns a 512-dimensional embedding vector for each input, while the second one learns a class weight matrix $\mathbf{W}$ to perform the classification into K fold classes. The ResCNN-GRU model is identical but using a unidirectional GRU layer instead.

vector representing the whole protein domain. We refer to this model as ResCNN-GRU. An alternative architecture is that based on a bidirectional GRU [64] which also processes the sequence in reverse order. In such a case, last states from both forward ($\overrightarrow{\mathbf{h}_L}$) and backward ($\overleftarrow{\mathbf{h}_L}$) GRU layers are concatenated into a vector of 2048 elements. We denote this model as ResCNN-BGRU.

Fully-connected part

Finally, the fully-connected (FC) part combines the recurrent output to create a fold-related embedding for the whole protein domain, which is then used to perform a preliminary fold classification. The classification step guides the model during training to learn a meaningful embedding space, which is related to the protein folds. Then, these learned embeddings are used for pairwise fold recognition in the test phase.

In particular, the FC part (Figure 5.2c) consists of two dense layers. The first one, with 512 units, is used to learn a nonlinear combination of the GRU output vector (1024 or 2048 for the unidirectional and bidirectional architectures, respectively) which shapes the fold-related embedding. As nonlinearity, both the sigmoid and the hyperbolic tangent (tanh) activation functions have been tested in our experiments. The last layer performs a linear classification of the 512-dimensional embeddings using K output units. Here, K is the number of fold classes in which the input proteins are classified during training. In the following subsections we detail how this last classification layer can be modified to learn more discriminative embedding vectors by distributing the fold class vectors in hyperspherical space.

5.2.4 Neural network model optimization

We trained our neural network models with mini-batches of 64 protein domains. To process variable-length sequences, we applied zero-padding to the maximum length within each mini-batch. After the GRU layer, we kept the last state vector of each domain sample before the zero-padding, which corresponds to the last amino acid residue of each domain in the mini-batch. In the bidirectional GRU, the same GRU layers are used but the amino acid sequences were first reversed for the backward layer, so the last state (before zero-padding) corresponds to the first residue of each domain. The optimization process was performed in two different stages by comparing the model predictions with the true fold classes (ground truth). In the first one (Figure 5.1a), we optimized the models by minimizing the well-known softmax cross-entropy loss, while in the second stage (Figure 5.1c) we used the large margin cosine loss (LMCL) [56], which is a normalized and margin discriminative version of the softmax loss. In this case, we also used a fixed (i.e. non-trainable) weight matrix in the classification layer (**W** in Figure 5.2c) which maximally separates fold class vectors in hyperspherical space (Figure 5.1b). We used the Adam optimizer [65] with an initial learning rate of 10^{-3}, which we reduced by a factor of 10 at epoch number 40, whereas the whole optimization process was completed in 80 epochs. In order to prevent overfitting to the most populated fold classes, we applied L_2 penalty with a small weight decay of $5 \cdot 10^{-4}$ and dropout [66] with a drop probability of 0.2 in the convolutional and the first FC layers.

5.2.5 Large margin cosine loss

The softmax cross-entropy loss (softmax loss for simplicity) is one of the most common loss functions for multi-class classification problems. It is defined as:

$$L_{softmax} = -\frac{1}{N}\sum_{i=1}^{N}\log p_i = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{f_{y_i}}}{\sum_{k=1}^{K}e^{f_k}}, \tag{5.1}$$

where p_i is the posterior probability of the $\mathbf{x}_i$ embedding sample being classified into its ground-truth class y_i, N is the number of training samples in the mini-batch ($i = 1,\dots,N$), K is the number of classes ($k = 1,\dots,K$), and f_k is the output of the last linear classification layer with weight matrix $\mathbf{W} \in \mathbb{R}^{K\times d}$ (the bias is set to zero for simplicity). For each input $\mathbf{x}_i$, the output corresponding to class k is computed as:

$$f_k = \mathbf{w}_k^T\mathbf{x}_i = \|\mathbf{w}_k\|\,\|\mathbf{x}_i\|\cos(\theta_{k,i}), \tag{5.2}$$

with $\theta_{k,i}$ being the angle between the vectors $\mathbf{w}_k$ and $\mathbf{x}_i$. If we enforce that $\|\mathbf{w}_k\| = 1$ through L_2 normalization, and $\|\mathbf{x}_i\| = s$ by using a tunable scale hyperparameter, the posterior probability only depends on the cosine of the angle $\theta_{k,i}$. This results in the normalized softmax loss (NSL), defined as:

$$L_{ns} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s\cos(\theta_{y_i,i})}}{\sum_{k=1}^{K}e^{s\cos(\theta_{k,i})}}. \tag{5.3}$$

The feature embeddings learned by NSL are angularly distributed, but they are not necessarily more discriminative than the ones learned by softmax loss. In order to control the classification boundaries, two variants of the NSL, the angular softmax (A-Softmax) loss [55] and the large margin cosine loss (LMCL) [56], introduce a margin hyperparameter ($m \geq 0$). The decision margin in LMCL is defined in cosine space rather than in angle space, which proved to be more beneficial when learning the classification boundaries [56]. This is therefore the loss function we adopted to optimize our neural network models, and is formally defined as:

$$L_{lmc} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s(\cos(\theta_{y_i,i})-m)}}{e^{s(\cos(\theta_{y_i,i})-m)}+\sum_{k\neq y_i}e^{s\cos(\theta_{k,i})}}, \tag{5.4}$$

subject to $\cos(\theta_{k,i}) = \hat{\mathbf{w}}_k^T\hat{\mathbf{x}}_i$, where $\hat{\mathbf{w}}_k$ and $\hat{\mathbf{x}}_i$ are the L_2 normalized vectors ($\hat{\mathbf{w}}_k = \mathbf{w}_k/\|\mathbf{w}_k\|$ and $\hat{\mathbf{x}}_i = \mathbf{x}_i/\|\mathbf{x}_i\|$).

As stated in the original paper [56], by L_2-normalizing the embedding vectors $\mathbf{x}_i$, we enforce them to be distributed on the surface of a d-dimensional hypersphere. Thus, the scaling hyperparameter s controls the radius of such hypersphere and its value increases with the number of classes. The margin hyperparameter m relates to the capacity of learning more discriminative embeddings. Possible values are in the range $m \in [0, \frac{K}{K-1})$, although high values close to the upper-bound could cause failures in convergence. Having this in mind, we tuned the scale s and margin m hyperparameters for each neural network model through cross-validation.

5.2.6 Thomson-derived hyperspherical prototypes

We hypothesize that by providing a non-trainable matrix $\mathbf{W} \in \mathbb{R}^{K \times d}$ to the classification layer we can ease the training process. Such matrix contains K pre-defined prototype vectors representing each fold class, $\mathbf{W} = \{\mathbf{w}_1, \ldots, \mathbf{w}_K\}$. Thus, we can shape the embedding space to be representative of the protein folds, and so extract more meaningful fold-related embeddings for each protein during the training stage (Figure 5.1c). The use of such prototype networks was first proposed in [58].

Optimal distribution of prototypes

We argue that the optimal configuration of the K prototype vectors is that which provides maximal separation in the angular space. This can be achieved by placing the K points equidistant on the surface of a d-dimensional hypersphere, so $\mathbf{w}_k \in \mathbb{S}^{d-1}$, as shown in Figure 5.1b. The Thomson problem [57] addresses this by studying the distribution of K charged particles on the surface of a unit 3D-sphere. The minimum energy configuration can be optimized by measuring the Coulomb's law. When using simplified units for electron charges and Coulomb's constant, the formula for a pair of electrons reduces to $E_{ij} = 1/r_{ij}$, relying only on the distance (r_{ij}) between the two points.

This can be extended to points located on the surface of a hypersphere of d dimensions and computed for all possible pairs of points [67]. We could therefore optimize the distribution of our $\mathbf{w}_k$ prototype vectors by minimizing the generalized Thomson loss (THL), defined as:

$$L_{th} = \sum_{k=1}^{K} \sum_{j=1}^{k-1} \frac{1}{\left\| \mathbf{w}_k - \mathbf{w}_j \right\|_2^2} + \frac{\lambda}{2} \sum_{k=1}^{K} (\|\mathbf{w}_k\|^2 - 1)^2. \tag{5.5}$$

The hyperparameter λ controls the weight of the norm constraint. Note that the Thomson loss uses the Euclidean distance between points, which is affected by the norm of each vector, while the cosine similarity is more adequate to measure the angular separation (independent of the norm). In order to remove the norm constraint from the loss function, we propose to directly maximize the Euclidean distance of the projected (L_2-normalized) vectors. Thus, we can remove the hyperparameter λ from equation (5.5), obtaining the following Thomson loss (THL$-$*sum*):

$$L_{th\_sum} = \sum_{k=1}^{K} \sum_{j=1}^{k-1} \left\| \frac{\mathbf{w}_k}{\|\mathbf{w}_k\|} - \frac{\mathbf{w}_j}{\|\mathbf{w}_j\|} \right\|_2^{-2}. \tag{5.6}$$

Alternatively, we can instead minimize the maximum cosine similarity computed for each prototype vector [58], using the following loss function (THL−*maxcos*):

$$L_{th\_maxcos} = \frac{1}{K} \sum_{k=1}^{K} \max_{j \neq k} \left(\frac{\mathbf{w}_k \cdot \mathbf{w}_j}{\|\mathbf{w}_k\| \|\mathbf{w}_j\|} \right). \tag{5.7}$$

Maximally separated prototype vectors are obtained by means of gradient descent over the proposed loss function (either THL−*sum* or THL−*maxcos*), where it must be noted that all possible pairs of points are taken to perform a single iteration step.

Initial prototype vectors

As initial matrix of prototypes we can consider a set of K Gaussian random variables of dimension d, $\mathbf{W}^{random}$. However, we found that the learned classification matrix from a model previously trained with the softmax cross-entropy loss (Figure 5.1a), $\mathbf{W}^{softmax}$, provides better results. Unlike $\mathbf{W}^{random}$, the matrix $\mathbf{W}^{softmax}$ has been trained to classify protein domains into folds, somehow preserving the arrangement of the structural classes within the learned space. To show this, we measured the intra- and inter-structural class prototype separation, as well as the angular Fisher score (AFS) [55]. Further details can be found in Supplementary Material 5.S.2.

5.2.7 Pairwise similarity scores

Cosine similarity measures

The FoldHSphere method (Figure 5.1d) uses the hyperspherical embeddings extracted from our neural network model to compute a fold similarity measure between each pair of protein domains. Following previous works [43, 48], we used the cosine similarity between two embedding vectors $[\mathbf{x}_i, \mathbf{x}_j] \in \mathbb{R}^d$ as metric, computed as:

$$\cos(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|}, \tag{5.8}$$

which is a measure of angular separation (in the range $[-1, 1]$) and independent of the norm of each embedding vector.

Random forest enhanced scores

To obtain an improved fold similarity score (FoldHSpherePro in Figure 5.1d), we trained a random forest (RF) model using our cosine similarity score along with the 84 pairwise similarity measures from [33, 34] and the DeepFR cosine similarity [43]. Thus, each input vector is of size 86 and corresponds to a pair of protein domains. The RF model uses this information to determine whether the domains in such a pair share the same fold class (binary classification). We trained and evaluated the RF models in a 10-stage cross-validation setting for the LINDAHL and LINDAHL_1.75 test sets independently. The random forest models used 500 decision trees each as in [43, 48].

5.2.8 Evaluation

Three-level rank performance accuracy

As originally proposed in [3], we evaluated the test protein domains at three levels of increasing difficulty—*family, superfamily* and *fold.* At each level, we differentiated between positive and negative pairs of domains. A negative pair contains two protein domains from different fold classes, while in a positive pair both domains are from the same fold class. Each level includes all the negative pairs, while positive pairs are selected according to the SCOP hierarchy [7]. That is, the *family level* contains pairs of domains that share the same family class, and therefore the same superfamily and fold classes. At the *superfamily level*, the domains in each pair share the same superfamily class—and therefore the same fold—but not the same family. Finally, domains in positive pairs at the *fold level* only share the same fold class, but neither share the same family nor superfamily.

At each of these levels, for every individual protein domain (query) we ranked the rest of domains (templates) according to their similarity scores. These can be either cosine similarities or random forest output scores. Then, we assigned the fold class of the most similar template to the query and computed the ratio of hits—*top 1 accuracy.* We also obtained the ratio of finding the correct fold class within the 5 first-ranked templates—*top 5 accuracy.* It must be noted that, instead of using the training set as the search database, in this evaluation we aim to find template domains inside the test set itself (either LINDAHL or LINDAHL_1.75).

In order to measure the statistical significance of our top 1 and top 5 accuracy results, we also provide standard errors estimated as the standard deviation of 1000 bootstrap samples. To do so, we sampled with replacement from the set of individual protein domains that are tested at each level (555, 434 and 321 domains respectively in the LINDAHL dataset). Then, for each sampled set we selected all negative pairs and positive pairs corresponding to the specific level, and proceeded with the evaluation as before.

Fold-level LINDAHL cross-validation evaluation

In order to compare with some recent methods [35–41, 44–46] we also provide results on a fold-level 2-stage cross-validation setting on the LINDAHL test set [22]. Here, the 321 protein domains which form positive pairs at the fold level are separated into two subsets LE_a and LE_b, with 159 and 162 domains each. Note that the rest of domains within LINDAHL (up to 976) are not considered during this evaluation. When evaluating the protein domains in each subset (e.g. LE_a), the domains in the other subset (LE_b) act as templates for ranking. Thus, the random forest models are trained using pairs of protein domains from one subset, whereas the evaluation is performed on the other one. In this evaluation, we report the averaged performance accuracy over both cross-validation subsets.

5.3 Results

5.3.1 Learning fold-related embeddings with LMCL

We first assessed the performance of the different neural network models trained either with the softmax loss (5.1) or the LMCL (5.4) (see Figure 5.1a), by cross-validation over the SCOPe 2.06 training set. For the softmax loss, we used the sigmoid activation in the embedding layer (first FC layer in Figure 5.2c), so that we can compare with the CNN-GRU and CNN-BGRU models from [48]. Then, for each model trained with the LMCL function, we tuned the scale and margin hyperparameters through cross-validation. We considered two values for the scale $s = \{30, 50\}$ and margins in the range $m = [0.1, 0.9]$. Here we tested two activation functions at the embedding layer: sigmoid as well as hyperbolic tangent (tanh). We argue that having negative and positive values ranging from -1 to 1 in the embedding vector (tanh activation) would better exploit the hyperspherical space than having only positive values (sigmoid activation, range $[0, 1]$).

The cross-validation fold classification accuracy on the training set for the different models and loss functions is shown in Figure 5.3. When using softmax loss, we can observe that the models applying residual convolutions (ResCNN-GRU and ResCNN-BGRU) perform better at fold classification than their counterparts (CNN-GRU and CNN-BGRU). We also observe that the tanh activation function yields better results than the sigmoid activation for all tested margin values in the LMCL function. In this case, the scale value $s = 30$ outperforms $s = 50$ for both activation functions. As for the margin, larger values seem to further benefit models applying bidirectional GRU (CNN-BGRU and ResCNN-BGRU), suggesting that these models have a higher discriminative capacity. The optimal LMCL hyperparameters for each model are summarized in Table 5.1a.

In Table 5.2 we provide the fold recognition accuracy results on the LINDAHL test set (at the family, superfamily and fold levels), when using the cosine similarity (5.8) as ranking metric. Here, we used the optimal LMCL hyperparameters to train each model on the whole training set, from which we extracted the fold-related embeddings. Table 5.2a shows that the learned embeddings from the ResCNN-GRU and ResCNN-BGRU models using softmax loss yield slightly better fold recognition performance at the three levels than the CNN-GRU and CNN-BGRU models. On the other hand, in Table 5.2b we observe a large

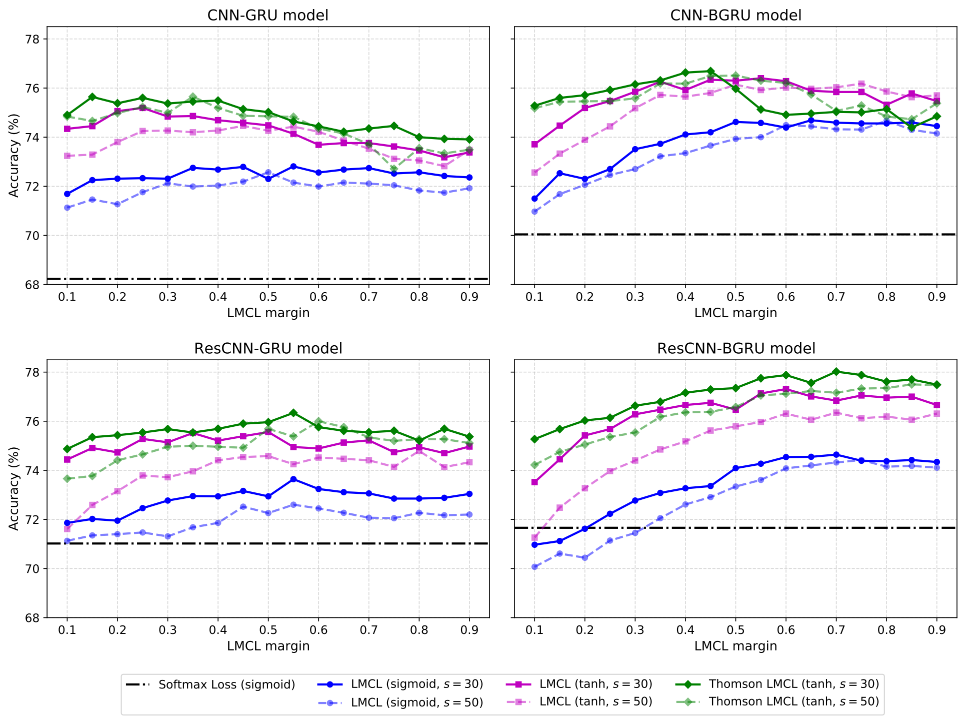


Figure 5.3: Cross-validation fold classification accuracy (%) results for different LMCL margins and scales $s = \{30, 50\}$, using the SCOPe 2.06 training set. The results are provided separately for each neural network model: CNN-GRU, CNN-BGRU, ResCNN-GRU and ResCNN-BGRU, trained using different combinations of activation function (in the embedding layer) and loss function. These are: softmax loss with sigmoid activation (dash-dotted horizontal line), LMCL with sigmoid activation (blue lines), LMCL with tanh activation (magenta lines) and Thomson LMCL with tanh activation (green lines). For the LMCL and Thomson LMCL results, solid lines and dashed lines correspond to scale values 30 and 50, respectively.

Table 5.1: Optimal set of hyperparameters for the LMCL function.

Model	**(a) LMCL**		**(b) Thomson LMCL**		
	scale	**margin**	**iter THL−*sum***	**scale**	**margin**
CNN-GRU	30	0.25	1130	30	0.25
CNN-BGRU	30	0.55	1172	30	0.45
ResCNN-GRU	30	0.50	1181	30	0.55
ResCNN-BGRU	30	0.60	1020	30	0.60

The scale and margin hyperparameters are provided for each neural network model and two approaches: **(a)** training the last classification layer end-to-end, **(b)** using the fixed prototype matrix by minimizing the Thomson loss THL−*sum*. We also include here the optimal iteration from the Thomson algorithm.

Table 5.2: Effect of model architecture and loss function choice on FoldHSphere performance using the LINDAHL dataset.

Model	**Family**		**Superfamily**		**Fold**	
	Top 1	***Top 5***	***Top 1***	***Top 5***	***Top 1***	***Top 5***
(a) Softmax Loss						
CNN-GRU [48]	68.6 (1.94)	89.2 (1.37)	56.2 (2.34)	77.4 (1.96)	56.7 (2.82)	74.1 (2.46)
CNN-BGRU [48]	71.0 (1.92)	87.7 (1.42)	60.1 (2.30)	77.2 (2.02)	58.3 (2.83)	**78.8** (2.27)
ResCNN-GRU	72.6 (1.87)	90.3 (1.24)	59.4 (2.32)	77.0 (2.00)	58.9 (2.88)	75.1 (2.44)
ResCNN-BGRU	**76.8** (1.78)	**91.2** (1.23)	**65.0** (2.29)	**82.0** (1.84)	**59.5** (2.79)	76.6 (2.35)
(b) LMCL						
CNN-GRU	**76.6** (1.80)	**90.8** (1.25)	64.7 (2.21)	80.2 (1.90)	65.7 (2.69)	79.8 (2.22)
CNN-BGRU	76.2 (1.79)	89.4 (1.31)	**70.5** (2.12)	83.2 (1.80)	72.0 (2.48)	81.0 (2.21)
ResCNN-GRU	75.7 (1.77)	89.7 (1.25)	66.4 (2.29)	81.1 (1.86)	67.6 (2.63)	80.1 (2.23)
ResCNN-BGRU	75.1 (1.84)	89.5 (1.30)	69.8 (2.25)	**85.3** (1.67)	**74.1** (2.42)	**82.2** (2.12)
(c) Thomson LMCL						
CNN-GRU	**80.0** (1.73)	90.6 (1.24)	66.8 (2.23)	80.2 (1.94)	66.0 (2.62)	80.1 (2.22)
CNN-BGRU	77.5 (1.75)	**91.7** (1.19)	69.8 (2.09)	85.3 (1.64)	72.6 (2.48)	82.2 (2.14)
ResCNN-GRU	76.9 (1.78)	89.5 (1.28)	69.1 (2.20)	82.9 (1.77)	69.5 (2.57)	79.4 (2.26)
ResCNN-BGRU	76.4 (1.77)	89.2 (1.30)	**72.8** (2.15)	**86.4** (1.63)	**75.1** (2.47)	**84.1** (2.12)

The fold recognition accuracy (%) results are provided at the family, superfamily and fold levels, considering both the top 1 and top 5 ranked templates. We compare the CNN-GRU, CNN-BGRU, ResCNN-GRU and ResCNN-BGRU neural network models, trained with different loss functions: **(a)** Softmax loss with sigmoid activation, **(b)** LMCL with tanh activation, and **(c)** Thomson LMCL with tanh activation. Optimal LMCL hyperparameters are in Table 5.1. For each accuracy result, we also provide in parentheses the standard error estimated using 1000 bootstraps.

performance boost at all levels when introducing the LMCL as loss function in comparison with softmax loss. At the fold level, we achieve performance gains of 9 or more percentage points for most of the models. More precisely, the CNN-BGRU and ResCNN-BGRU models stand out for their remarkable results at the fold level, with 72.0% and 74.1% top 1 accuracy values respectively.

5.3.2 Enhancing embedding discrimination power through Thomson-derived hyperspherical prototypes

We then tested the performance of our neural network models trained with a fixed matrix of prototypes $\mathbf{W} \in \mathbb{R}^{K \times d}$ in the classification layer (Figure 5.1c), being the number of fold classes $K = 1154$ and embedding dimension $d = 512$. The fold prototype vectors have been maximally separated in the angular space by minimizing the THL$-$*sum* (5.6), using the $\mathbf{W}^{softmax}$ from each model as initial matrix (Figure 5.1b). A detailed study comparing the performance of the two variants for the Thomson loss (THL$-$*sum* and THL$-$*maxcos*) and two options for the initial matrix ($\mathbf{W}^{softmax}$ and $\mathbf{W}^{random}$) can be found in Supplementary Material 5.S.2.

Given the optimized matrix of prototypes for each model, we tuned the LMCL scale and margin values by cross-validation over the SCOPe 2.06 training set, considering the tanh activation in the embedding layer. Results from this tuning are also shown in Figure 5.3. As can be observed, the Thomson LMCL achieves better fold classification results, specially for the models applying residual convolutions, and particularly in the case of ResCNN-BGRU.

Finally, we set the optimal LMCL hyperparameters for each model (Table 5.1b) and trained them to extract fold-related hyperspherical embeddings. The fold recognition LINDAHL results in Table 5.2c show that, at the fold level, all the models benefit from the Thomson LMCL. Our best model, the ResCNN-BGRU, achieves top 1 accuracy values of 76.4%, 72.8% and 75.1% at the family, superfamily and fold levels, and top 5 accuracy of 89.2%, 86.4% and 84.1% at each level, respectively.

5.3.3 Analysis of the hyperspherical embeddings

The fold recognition results of our FoldHSphere method using the ResCNN-BGRU model trained with hyperspherical prototypes reflect the effectiveness and discrimination capability of the learned hyperspherical embeddings. To further illustrate this, we analyzed the 512-dimensional embeddings extracted from the 976 protein domains in the LINDAHL dataset. Figure 5.4 compares the histogram of cosine similarities computed between each pair of embeddings for the softmax, LMCL and Thomson LMCL options. For each one, we plotted separately the histogram of negative pairs (different fold classes) and positive pairs (same fold class). It can be seen that the Thomson LMCL provides a better separation between positive and negative pairs, with a small overlap between the two groups. This directly contributes to a better performance in the pairwise fold recognition task. Additionally, we provide a two-dimensional visualization of the embedding space learned by the three loss functions in Supplementary Figure 5.S6, as well as a dendroheatmap of the hyperspherical embeddings obtained by the Thomson LMCL approach in Supplementary Figure 5.S7.

5.3.4 FoldHSphere and FoldHSpherePro pairwise fold recognition performance results

Finally, we compare the results of our FoldHSphere and FoldHSpherePro approaches with several methods from the state-of-the-art, considering both the LINDAHL and LINDAHL_1.75 test sets. The FoldHSphere results correspond to those from the ResCNN-BGRU model trained with hyperspherical prototypes (Table 5.2). The FoldHSpherePro results were obtained after conducting a 10-stage cross-validation on a random forest model using the FoldHSphere scores along with other pre-computed protein pairwise similarities as inputs.

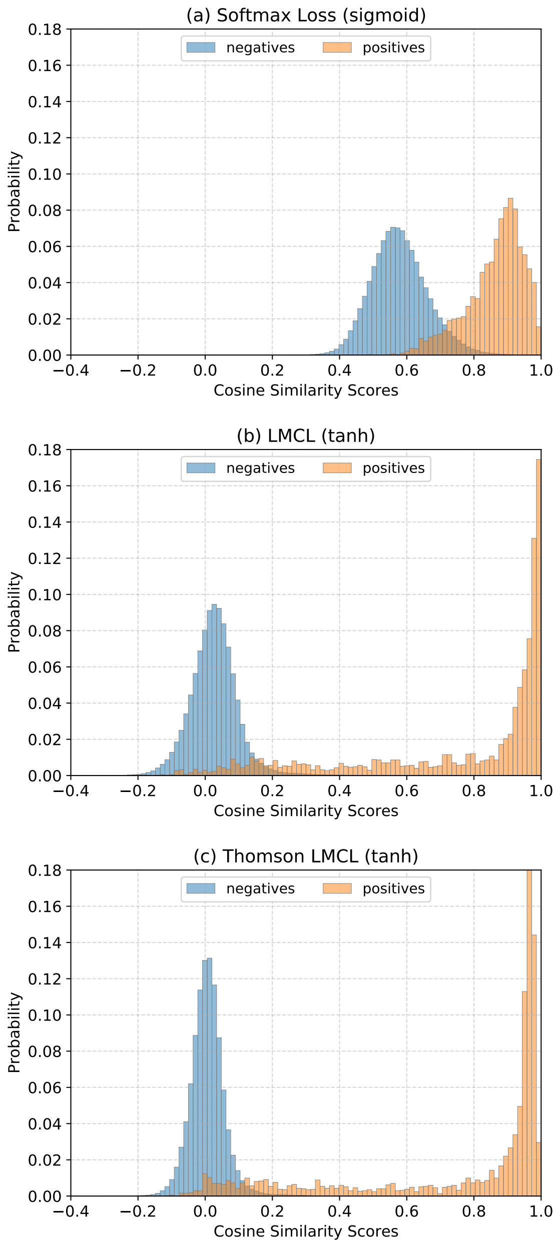


Figure 5.4: Cosine similarity probability histograms computed for all unique pairs within the LINDAHL test set (976 domains), grouping the negative pairs in blue color, and positive pairs in orange. To compute the cosine similarity scores, we used the embeddings extracted from the ResCNN-BGRU model trained with **(a)** softmax loss with sigmoid activation, **(b)** LMCL with tanh activation, or **(c)** Thomson LMCL with tanh activation.

The three-level LINDAHL fold recognition results are shown in Table 5.3. We can see that our FoldHSphere method yields better top 1 accuracy values, above 12 percentage points at the superfamily and fold levels compared to the state-of-the-art method CNN-BGRU [48]. At the family level, we outperform all the deep learning methods. However, the alignment methods and approaches relying on pairwise similarities provide better results at this level. We include such information in the FoldHSpherePro method, which can be compared to DeepFRpro [43] and CNN-BGRU-RF+ [48] as all of them apply the same random forest ensemble approach. Our method provides a significant performance boost, obtaining remarkable top 1 accuracy results, with values of 79.0% at the superfamily level and 81.3% at the fold level. In terms of top 5 accuracy values, FoldHSpherePro also achieves the best performance, providing 89.2% and 90.3% at superfamily and fold levels respectively. On the other hand, at the family level we obtain on par results with the CNN-BGRU-RF+ method, being only outperformed by alignment and threading methods. This suggests that the performance of deep learning approaches might be saturating at this level. Similar conclusions can be drawn when evaluating the LINDAHL_1.75 test set (Table 5.4). Here we only compare to the DeepFR and CNN-BGRU methods, as they have been previously tested on such a dataset. The results show that our FoldHSpherePro approach also performs the

Table 5.3: Three-level LINDAHL fold recognition results of FoldHSphere and FoldHSpherePro in comparison with the state-of-the-art.

Method	**Family**		**Superfamily**		**Fold**	
	Top 1	***Top 5***	***Top 1***	***Top 5***	***Top 1***	***Top 5***
PSI-BLAST [3]	71.2	72.3	27.4	27.9	4.0	4.7
HHpred [14]	82.9	87.1	58.0	70.0	25.2	39.4
RAPTOR [14]	**86.6**	89.3	56.3	69.0	38.2	58.7
BoostThreader [14]	86.5	90.5	66.1	76.4	42.6	57.4
SPARKS-X [15]	84.1	90.3	59.0	76.3	45.2	67.0
FOLDpro [34]	85.0	89.9	55.0	70.0	26.5	48.3
RF-Fold [34]	84.5	91.5	63.4	79.3	40.8	58.3
DN-Fold [34]	84.5	91.2	61.5	76.5	33.6	60.7
RFDN-Fold [34]	84.7	91.5	65.7	78.8	37.7	61.7
MRFalign [43]	85.2	90.8	72.4	80.9	38.6	56.7
CEthreader [48]	76.6	87.2	69.4	81.8	52.3	70.4
DeepFR (s2) [43]	65.4	83.4	51.4	67.1	56.1	70.1
DeepFRpro (s2) [43]	83.1	92.3	69.6	82.5	66.0	78.8
VGGfold [47]	67.9	84.3	53.2	68.4	58.3	73.5
CNN-BGRU [48]	71.0	87.7	60.1	77.2	58.3	78.8
CNN-BGRU-RF+ [48]	85.4	**93.5**	73.3	87.8	76.3	85.7
FoldHSphere	76.4	89.2	72.8	86.4	75.1	84.1
FoldHSpherePro	85.2	93.0	**79.0**	**89.2**	**81.3**	**90.3**

The accuracy (%) results are provided at the family, superfamily and fold levels, considering both the top 1 and top 5 ranked templates.

Table 5.4: Three-level LINDAHL_1.75 fold recognition results of FoldHSphere and FoldHSpherePro in comparison with the state-of-the-art.

Method	**Family**		**Superfamily**		**Fold**	
	Top 1	***Top 5***	***Top 1***	***Top 5***	***Top 1***	***Top 5***
DeepFR (s2) [48]	72.2	85.6	46.6	64.3	50.8	67.1
DeepFRpro (s2) [48]	87.0	93.8	71.9	82.6	63.5	77.2
CNN-BGRU [48]	73.1	88.3	60.1	74.9	60.1	78.7
CNN-BGRU-RF+ [48]	**88.5**	**94.3**	74.0	86.3	71.1	86.8
FoldHSphere	77.9	89.0	72.4	87.0	75.8	84.3
FoldHSpherePro	87.9	94.1	**81.2**	**90.0**	**80.9**	**88.5**

The accuracy (%) results are provided at the family, superfamily and fold levels, considering both the top 1 and top 5 ranked templates.

best in this dataset, yielding top 1 accuracy values of 87.9%, 81.2% and 80.9% at the three levels respectively.

Figure 5.5 includes the evaluation results of the fold-level 2-stage cross-validation setting on the LINDAHL dataset (over subsets LE_a and LE_b). In this case, we only compare to ensemble methods that have been assessed with such a methodology, namely TA-fold [35], the multi-view learning frameworks MT-fold [36], EMfold [37] and MLDH-Fold [38], the learning to rank approaches Fold-LTR-TCP [39], FoldRec-C2C [40] and ProtFold-DFG [41], and the deep learning methods DeepSVM-fold [44], MotifCNN-fold [45] and SelfAT-Fold [46] (residue and motif options). The results in Figure 5.5 show that our FoldHSpherePro method outperforms all of them yielding an accuracy of 85.6%.

5.4 Discussion

In order to learn an embedding space that is representative of the protein folds, we have proposed a two-stage learning procedure. Our intuition here is that pre-defining the structure of the embedding space through fixed fold prototypes would ease the learning process for a neural network that embeds individual proteins into this space. The result of our experiments indicate that this intuition is well founded.

There are two important considerations when pre-defining the embedding space in our methodology: the initialization of the fold prototypes and how to obtain a suitable spatial distribution of these prototypes in the embedding space. Here we find that rather than initializing with random vectors in the hyperspherical space, better results are obtained by using the weight matrix from the classification layer of a neural network previously trained to map proteins to fold classes. Then, by maximally separating the weight vectors in this matrix in hyperspherical space, we obtain an effective configuration of the fold prototypes. We believe one of the main reasons is that such a matrix preserves the arrangement of the

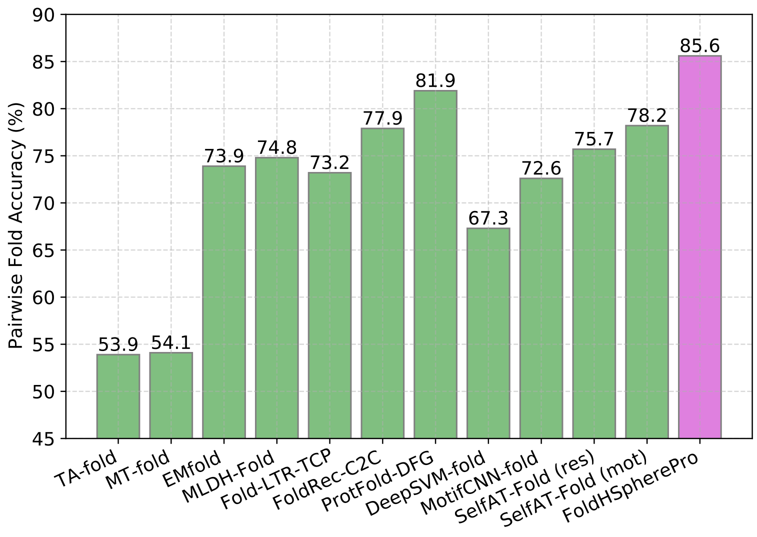


Figure 5.5: Fold-level LINDAHL fold recognition accuracy (%) results of our proposed FoldHSpherePro method in comparison with other ensemble methods from the state-of-the-art. The results are averaged over the two cross-validated subsets (LE_a and LE_b).

structural classes in the learned space, grouping related fold classes together and pushing them away from folds of unrelated structural classes (see Supplementary Material 5.S.1).

Our experimental results in Table 5.2 display a large performance boost at the superfamily and fold levels when comparing our methodology (using LMCL) to previous approaches that use the softmax loss. Our initial intuition for the lower performance of the state-of-the-art at these levels is that since evaluation is done for pairs of proteins, it is possible that two proteins from different folds lying near the fold classification boundary are closer to each other than they are to proteins from their respective folds. This informs our choice for using the LMCL as loss function, which introduces a margin between fold classes to avoid these cases.

A further performance gain is seen when combining the LMCL margin with the pretrained fold prototypes (Table 5.2c). Here we use the fold prototypes optimized in the previous stage as a fixed (non-trainable) classification matrix for each neural network. We believe that the additional performance improvement is due to the simplified learning process that results from having this pre-defined organization of the folds in the embedding space, which is especially useful with limited and unbalanced training data. Stated differently, our models can focus on projecting protein embeddings closest to the corresponding fold prototypes without simultaneously learning where these prototypes should be.

We also observe from Figure 5.3 and Table 5.2 that the models applying residual convolutions benefit more from the use of pre-trained prototypes compared to only optimizing with LMCL. This suggests the residual connections might extract more robust features for each amino acid, which seems to be helpful for the recurrent layer to obtain a better fixed-size representation for the whole protein domain. In particular, our ResCNN-BGRU architecture provides the best results, which can be attributed to its greater flexibility compared to the other tested architectures.

5.5 Conclusion

In this work we have proposed the FoldHSphere method to tackle the protein fold recognition problem. We described a neural network training procedure to learn fold-representative hyperspherical embeddings for the protein domains. The embeddings were extracted from a residual-convolutional and recurrent network architecture (ResCNN-BGRU), which is trained by minimizing the angular large margin cosine loss (LMCL) around pre-defined prototypes for the fold classes. We used a Thomson-based loss function to maximally separate the fold prototypes in hyperspherical space. This way, our embeddings proved to be more effective at identifying the fold class of each protein domain by pairwise comparison. When evaluating the LINDAHL dataset, FoldHSphere alone provided a remarkable performance boost at the superfamily and fold levels, being competitive even with previous ensemble methods. Furthermore, our FoldHSpherePro ensemble method significantly improved the state-of-the-art results, outperforming the best method CNN-BGRU-RF+ at these levels. Therefore, due to their discrimination capability, the hyperspherical embeddings could be used to find template proteins even when the amino acid sequence similarities are low and thus advance in the template-based modeling of protein structures.

As future work, we will explore the application of recently proposed embeddings from language models pre-trained using millions of unannotated protein sequences for the protein fold recognition task, as they have shown promising results in several downstream tasks, such as protein secondary structure prediction and subcellular localization prediction [68–70].

Data and code availability

Source code, data and trained models can be found at
`http://sigmat.ugr.es/~amelia/FoldHSphere/`.

Funding

This work has been supported by the Spanish Ministry of Science, Innovation and Universities Project No. PID2019-104206GB-I00 / SRA (State Research Agency) / 10.13039/501100011033, as well as the FPI grant BES-2017-079792.

Acknowledgements

Amelia would like to thank Chirag Raman for his valuable input and feedback provided throughout the project.

References

[1] C. Chothia and A. V. Finkelstein. The classification and origins of protein folding patterns. *Annual Review of Biochemistry*, 59(1):1007–1035, 1990.

[2] D. T. Jones, W. R. Taylor, and J. M. Thornton. A new approach to protein fold recognition. *Nature*, 358(6381):86, 1992.

[3] E. Lindahl and A. Elofsson. Identification of related proteins on family, superfamily and fold level. *Journal of Molecular Biology*, 295(3):613–625, 2000.

[4] R. D. Schaeffer and V. Daggett. Protein folds and protein folding. *Protein Engineering, Design & Selection*, 24(1-2):11–19, 2010.

[5] R. Kolodny, L. Pereyaslavets, A. O. Samson, and M. Levitt. On the universe of protein folds. *Annual Review of Biophysics*, 42:559–582, 2013.

[6] H. M. Berman, J. Westbrook, Z. Feng, et al. The protein data bank. *Nucleic Acids Research*, 28(1):235–242, 2000.

[7] A. G. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia. SCOP: A structural classification of proteins database for the investigation of sequences and structures. *journal of Molecular Biology*, 247(4):536–540, 1995.

[8] N. K. Fox, S. E. Brenner, and J.-M. Chandonia. SCOPe: Structural classification of proteins–extended, integrating SCOP and ASTRAL data and classification of new structures. *Nucleic Acids Research*, 42(D1):D304–D309, 2014.

[9] C. A. Orengo, A. D. Michie, S. Jones, et al. CATH — a hierarchic classification of protein domain structures. *Structure*, 5(8):1093–1109, 1997.

[10] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 1990.

[11] J. Söding. Protein homology detection by HMM–HMM comparison. *Bioinformatics*, 21(7):951–960, 2005.

[12] J. Ma, S. Wang, Z. Wang, and J. Xu. MRFalign: Protein homology detection through alignment of Markov random fields. *PLoS Computational Biology*, 10(3):e1003500, 2014.

[13] J. Xu, M. Li, D. Kim, and Y. Xu. RAPTOR: optimal protein threading by linear programming. *Journal of Bioinformatics and Computational Biology*, 1(1):95–117, 2003.

[14] J. Peng and J. Xu. Boosting protein threading accuracy. In *Annual International Conference on Research in Computational Molecular Biology*, pages 31–45, 2009.

[15] Y. Yang, E. Faraggi, H. Zhao, and Y. Zhou. Improving protein fold recognition and template-based modeling by employing probabilistic-based matching between predicted one-dimensional structural properties of query and corresponding native properties of templates. *Bioinformatics*, 27(15):2076–2082, 2011.

[16] J. Ma, J. Peng, S. Wang, and J. Xu. A conditional neural fields model for protein threading. *Bioinformatics*, 28(12):i59–i66, 2012.

[17] J. A. Morales-Cordovilla, V. Sanchez, and M. Ratajczak. Protein alignment based on higher order conditional random fields for template-based modeling. *PLoS ONE*, 13(6):e0197912, 2018.

[18] D. W. A. Buchan and D. T. Jones. EigenTHREADER: analogous protein fold recognition by efficient contact map threading. *Bioinformatics*, 33(17):2684–2690, 2017.

[19] W. Zheng, Q. Wuyun, Y. Li, et al. Detecting distant-homology protein structures by aligning deep neural-network based contact maps. *PLoS Computational Biology*, 15(10):1–27, 2019.

[20] L. Wei and Q. Zou. Recent progress in machine learning-based methods for protein fold recognition. *International journal of Molecular Sciences*, 17(12):2118, 2016.

[21] H.-B. Shen and K.-C. Chou. Ensemble classifier for protein fold pattern recognition. *Bioinformatics*, 22(14):1717–1722, 2006.

[22] Q. Dong, S. Zhou, and J. Guan. A new taxonomy-based protein fold recognition approach based on autocross-covariance transformation. *Bioinformatics*, 25(20):2655–2662, 2009.

[23] J.-Y. Yang and X. Chen. Improving taxonomy-based protein fold recognition by using global and local features. *Proteins: Structure, Function, and Bioinformatics*, 79(7):2053–2064, 2011.

[24] A. Dehzangi, K. K. Paliwal, J. Lyons, A. Sharma, and A. Sattar. A segmentation-based method to extract structural and evolutionary features for protein fold recognition. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 11(3):510–519, 2014.

[25] K. K. Paliwal, A. Sharma, J. Lyons, and A. Dehzangi. Improving protein fold recognition using the amalgamation of evolutionary-based and structural based information. *BMC Bioinformatics*, 15(16):1–9, 2014.

[26] J. Lyons, A. Dehzangi, R. Heffernan, et al. Advancing the accuracy of protein fold recognition by utilizing profiles from hidden Markov models. *IEEE Transactions on Nanobioscience*, 14(7):761–772, 2015.

[27] D. Chen, X. Tian, B. Zhou, and J. Gao. ProFold: Protein fold classification with additional structural features and a novel ensemble classifier. *BioMed Research International*, 2016:1–10, 2016.

[28] W. Ibrahim and M. S. Abadeh. Protein fold recognition using deep kernelized extreme learning machine and linear discriminant analysis. *Neural Computing and Applications*, 31(8):4201–4214, 2019.

[29] S. Bankapur and N. Patil. An enhanced protein fold recognition for low similarity datasets using convolutional and skip-gram features with deep neural network. *IEEE Transactions on*

NanoBioscience, 20(1):42–49, 2020.

[30] W. Elhefnawy, M. Li, J. Wang, and Y. Li. DeepFrag-k: a fragment-based deep learning approach for protein fold recognition. *BMC Bioinformatics*, 21(6):1–12, 2020.

[31] K. Stapor, I. Roterman-Konieczna, and P. Fabian. Machine learning methods for the protein fold recognition problem. In *Machine Learning Paradigms*, volume 149, pages 101–127. Springer, 2019.

[32] J. Cheng and P. Baldi. A machine learning information retrieval approach to protein fold recognition. *Bioinformatics*, 22(12):1456–1463, 2006.

[33] T. Jo and J. Cheng. Improving protein fold recognition by random forest. *BMC Bioinformatics*, 15(11):S14, 2014.

[34] T. Jo, J. Hou, J. Eickholt, and J. Cheng. Improving protein fold recognition by deep learning networks. *Scientific Reports*, 5:17573, 2015.

[35] J. Xia, Z. Peng, D. Qi, H. Mu, and J. Yang. An ensemble approach to protein fold classification by integration of template-based assignment and support vector machine classifier. *Bioinformatics*, 33(6):863–870, 2016.

[36] K. Yan, X. Fang, Y. Xu, and B. Liu. Protein fold recognition based on multi-view modeling. *Bioinformatics*, 35(17):2982–2990, 2019.

[37] K. Yan, J. W. an Yong Xu, and B. Liu. Protein fold recognition based on auto-weighted multi-view graph embedding learning model. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2020.

[38] K. Yan, J. Wen, Y. Xu, and B. Liu. MLDH-Fold: Protein fold recognition based on multi-view low-rank modeling. *Neurocomputing*, 421:127–139, 2021.

[39] B. Liu, Y. Zhu, and K. Yan. Fold-LTR-TCP: protein fold recognition based on triadic closure principle. *Briefings in Bioinformatics*, 2019.

[40] J. Shao, K. Yan, and B. Liu. FoldRec-C2C: protein fold recognition by combining cluster-to-cluster model and protein similarity network. *Briefings in Bioinformatics*, 2020.

[41] J. Shao and B. Liu. ProtFold-DFG: protein fold recognition by combining Directed Fusion Graph and PageRank algorithm. *Briefings in Bioinformatics*, 2020.

[42] J. Hou, B. Adhikari, and J. Cheng. DeepSF: deep convolutional neural network for mapping protein sequences to folds. *Bioinformatics*, 34(8):1295–1303, 2018.

[43] J. Zhu, H. Zhang, S. C. Li, et al. Improving protein fold recognition by extracting fold-specific features from predicted residue–residue contaacts. *Bioinformatics*, 33(23):3749–3757, 2017.

[44] B. Liu, C.-C. Li, and K. Yan. DeepSVM-fold: protein fold recognition by combining support vector machines and pairwise sequence similarity scores generated by deep learning networks. *Briefings in Bioinformatics*, 2019.

[45] C.-C. Li and B. Liu. MotifCNN-fold: protein fold recognition based on fold-specific features

extracted by motif-based convolutional neural networks. *Briefings in Bioinformatics*, 21(6):2133–2141, 2020.

[46] Y. Pang and B. Liu. SelfAT-Fold: protein fold recognition based on residue-based and motif-based self-attention networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2020.

[47] Y. Liu, Y.-H. Zhu, X. Song, J. Song, and D.-J. Yu. Why can deep convolutional neural networks improve protein fold recognition? a visual explanation by interpretation. *Briefings in Bioinformatics*, 2021.

[48] A. Villegas-Morcillo, A. M. Gomez, J. A. Morales-Cordovilla, and V. Sanchez. Protein fold recognition from sequences using convolutional and recurrent neural networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 18(6):2848–2854, 2021.

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

[50] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[51] A. Vaswani, N. Shazeer, N. Parmar, et al. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008, 2017.

[52] K. He, X. Zhang, S. Ren, and J. Su. Deep residual learning for image recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[53] Y. Wen, K. Zhang, Z. Li, and Y. Qiao. A discriminative feature learning approach for deep face recognition. In *European Conference on Computer Vision (ECCV)*, pages 499–515, 2016.

[54] W. Liu, Y. Wen, Z. Yu, and M. Yang. Large-margin softmax loss for convolutional neural networks. In *International Conference on Machine Learning*, volume 2, page 7, 2016.

[55] W. Liu, Y. Wen, Z. Yu, et al. SphereFace: Deep hypersphere embedding for face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 212–220, 2017.

[56] H. Wang, Y. Wang, Z. Zhou, et al. CosFace: Large margin cosine loss for deep face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5265–5274, 2018.

[57] J. J. Thomson. XXIV. On the structure of the atom: an investigation of the stability and periods of oscillation of a number of corpuscles arranged at equal intervals around the circumference of a circle; with application of the results to the theory of atomic structure. *The London, Edinburgh, and Dublin Philosophical Magazine and journal of Science*, 7(39):237–265, 1904.

[58] P. Mettes, E. van der Pol, and C. G. M. Snoek. Hyperspherical prototype networks. In *Advances in Neural Information Processing Systems*, 2019.

[59] M. Steinegger and J. Söding. MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nature Biotechnology*, 35(11):1026–1028, 2017.

[60] W. Li and A. Godzik. Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*, 22(13):1658–1659, 2006.

[61] C. Camacho, G. Coulouris, V. Avagyan, et al. BLAST+: architecture and applications. *BMC Bioinformatics*, 10(1):1–9, 2009.

[62] C. N. Magnan and P. Baldi. SSpro/ACCpro 5: almost perfect prediction of protein secondary structure and relative solvent accessibility using profiles, machine learning and structural similarity. *Bioinformatics*, 30(18):2592–2597, 2014.

[63] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *International Conference on Machine Learning*, pages 448–456, 2015.

[64] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[65] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[66] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[67] P. Raman and J. Yang. Optimization on the surface of the (hyper)-sphere. *arXiv preprint arXiv:1909.06463*, 2019.

[68] M. Heinzinger, A. Elnaggar, Y. Wang, et al. Modeling aspects of the language of life through transfer-learning protein sequences. *BMC Bioinformatics*, 20(1):1–17, 2019.

[69] A. Rives, J. Meier, T. Sercu, et al. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, 118(15):e2016239118, 2021.

[70] A. Elnaggar, M. Heinzinger, C. Dallago, et al. ProtTrans: Towards cracking the language of life's code through self-supervised deep learning and high performance computing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–16, 2021.

[71] M. S. Klausen, M. C. Jespersen, H. Nielsen, et al. NetSurfP-2.0: Improved prediction of protein structural features by integrated deep learning. *Proteins: Structure, Function, and Bioinformatics*, 87(6):520–527, 2019.

[72] L. V. der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(11):2579–2605, 2008.

[73] A. Paszke, S. Gross, S. Chintala, et al. Automatic differentiation in PyTorch. In *Advances in Neural Information Processing Systems*, 2017.

[74] W. A. Falcon et al. PyTorch Lightning. 2019.

[75] F. Pedregosa, G. Varoquaux, A. Gramfort, et al. Scikit-learn: machine learning in Python. *journal of Machine Learning Research*, 12:2825–2830, 2011.

5.S Supplementary material

5.S.1 Training dataset and cross-validation subsets

Table 5.S1: Number of protein domains, families, superfamilies and folds in each structural class within the SCOPe 2.06 training dataset.

Structural class	Protein domains	Family classes	Superfamily classes	Fold classes
a	2684	943	480	273
b	3603	858	343	168
c	4739	900	233	144
d	3882	1193	528	368
e	319	100	65	65
f	296	137	98	56
g	610	213	115	80
Total	16133	4344	1862	1154

Table 5.S2: Cross-validation subsets for the SCOPe 2.06 training dataset. Here we split the 16133 protein domains into 5 groups, each one including domains from different family classes.

Subset	Protein domains		Family classes		Superfamily classes		Fold classes	
	Train	Valid	Train	Valid	Train	Valid	Train	Valid
CV1	12859	3274	3474	870	1641	615	1015	442
CV2	12895	3238	3474	870	1642	606	1032	429
CV3	12794	3339	3474	870	1633	630	1030	436
CV4	13023	3110	3474	870	1646	611	1035	436
CV5	12961	3172	3480	864	1626	632	1035	436

5.S.2 Thomson-derived hyperspherical prototypes

Optimization curves

In order to maximally separate our fold class prototypes in the hyperspherical space, we minimized a Thomson-related loss function to optimize the matrix $\mathbf{W} \in \mathbb{R}^{K \times d}$, being the number of fold classes $K = 1154$ and embedding dimension $d = 512$. We accelerated the optimization process by using the Adam optimizer [65] with a learning rate of 10^{-3}. To check the convergence of the algorithm, at each iteration we monitored both the sum of all inverse distances (THL−*sum*) and the maximum cosine similarity between all pairs of prototypes. Figure 5.S1 includes the optimization curves for the two variants, THL−*sum* or THL−*maxcos*, and initial matrices $\mathbf{W}^{softmax}$ from the CNN-GRU model or $\mathbf{W}^{random}$. When optimizing the THL−*sum* function, we see that the maximum cosine similarity between all pairs of prototypes decreases until a certain point in which starts increasing again. This suggests the algorithm is incorrectly trying to separate the majority of points, while bringing a few others together in order to further minimize the loss function. To avoid this unwanted behavior, we set the optimum iteration as the one with minimum value of maximum cosine similarity, which is 1130 for the $\mathbf{W}^{softmax}$ and 546 for the $\mathbf{W}^{random}$ initial matrices. On the other hand, the THL$-$*maxcos* function provides lower maximum cosine similarity, whereas the sum of inverse distances remains higher. In this case, we optimized for a huge number of iterations (50,000).

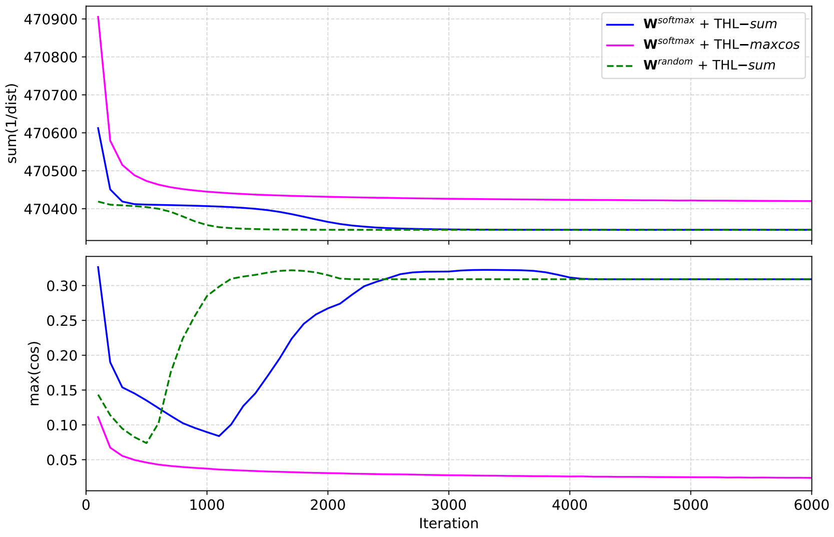


Figure 5.S1: Thomson optimization curves at each iteration monitoring two metrics: sum of inverse of distances (above) and maximum cosine similarity (below) between all pairs of prototypes. For both metrics, we compare different options for initialization and loss function: initial matrix $\mathbf{W}^{softmax}$ and THL$-$*sum* (blue line), $\mathbf{W}^{softmax}$ and THL$-$*maxcos* (magenta line), or $\mathbf{W}^{random}$ and THL$-$*sum* (green dashed line).

Cosine similarity of prototype vectors

We then examined several structural characteristics from the optimized prototypes, in comparison with the initial matrices $\mathbf{W}^{softmax}$ and $\mathbf{W}^{random}$. In Figure 5.S2, we plot the $K \times K$ cosine similarity matrix for each set of fold class vectors or prototypes, as well as the histogram of such pairwise cosine similarities. To ensure maximum angular separation between prototypes—given the number of points (fold classes) and dimensions in the hypersphere—the cosine values should be around 0 or negatives, as this translates into angles close to or greater than 90 degrees. We observe this in the cosine similarity histograms for the optimized prototypes. However, the cosine similarity matrix for the initial $\mathbf{W}^{softmax}$ suggests that the fold class vectors learned by softmax loss may contain rich information about the structural classes (i.e. 7 clusters can be observed, the same number as structural classes defined in SCOPe [8]).

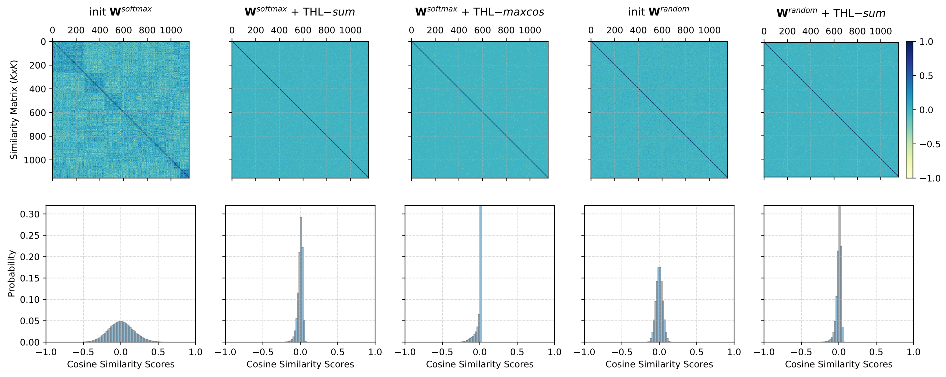


Figure 5.S2: Cosine similarity matrices (above) and cosine similarity probability histograms (below) computed for the K prototypes (corresponding to K different folds). The compared sets of prototypes, from left to right are: initial matrix $\mathbf{W}^{softmax}$, optimized $\mathbf{W}^{softmax}$ with THL−*sum*, optimized $\mathbf{W}^{softmax}$ with THL−*maxcos*, initial $\mathbf{W}^{random}$, or optimized $\mathbf{W}^{random}$ with THL−*sum*.

Intra- and inter-structural class prototype separation

To evaluate the structural class information contained in the optimized prototypes, we grouped the K fold prototypes into their respective structural classes (7 classes from a to g in SCOPe [8]). Then, we measured the average separation in terms of cosine distance, considering fold prototypes within the same structural class (intra-class separation), and prototypes from different structural classes (inter-class separation). We also computed the angular Fisher score (AFS) [55], defined as:

$$AFS = \frac{S_{inter}}{S_{intra}} = \frac{\sum_r \sum_{\mathbf{w}_j \in \mathbf{W}_r} (1 - \cos(\mathbf{w}_j, \mathbf{m}_r))}{\sum_r n_r (1 - \cos(\mathbf{m}_r, \mathbf{m}))}, \tag{5.9}$$

where S_{inter} and S_{intra} are the inter-class and intra-class scatter values, respectively. $\mathbf{W}_r$ is a subset of $\mathbf{W}$ containing n_r prototypes from structural class r, $\mathbf{m}_r$ is the mean vector of those n_r prototypes, and $\mathbf{m}$ is the mean vector of the whole set of prototypes.

In Figure 5.S3, we can see that the optimized prototypes from the $\mathbf{W}^{softmax}$ using the THL−*sum* function retain the structural class information, with higher intra-class cosine similarity values than those across different structural classes. However, this information is not preserved when using $\mathbf{W}^{random}$ as initial matrix or the THL−*maxcos* as a loss function. Additionally, in Table 5.S3 we observe that the initial $\mathbf{W}^{softmax}$ provides a lower angular Fisher score (AFS) value than the initial $\mathbf{W}^{random}$. Once more, this suggests that the prototypes in $\mathbf{W}^{softmax}$ are more informative about the structural classes. However, the AFS values decrease after optimizing both initial matrices with Thomson. This can be attributed to the significant increase in the cosine distance between all prototypes, which also increases the S_{intra} term in equation (5.9). Overall, the $\mathbf{W}^{softmax}$ with THL−*sum* option provides a better angular Fisher score (0.9073) than the rest of options.

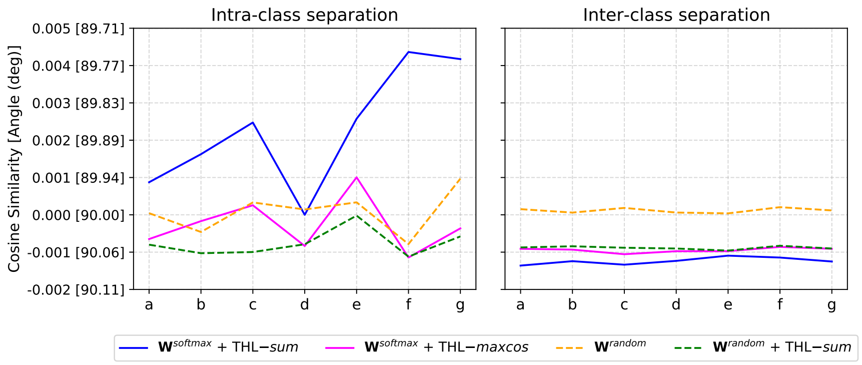


Figure 5.S3: Intra- and inter-structural class separation of the K prototypes (corresponding to K different folds) considering the 7 structural classes $\{a, b, c, d, e, f, g\}$ defined in SCOPe [8]. The separation values have been measured in terms of cosine similarity and converted to angles in degrees. The intra-class separation (left) is computed for all pairs within the same structural class, while the inter-class separation (right) is computed for each prototype in one structural class with the rest of classes. Here, we compare different options for the set of prototypes: initial matrix $\mathbf{W}^{softmax}$ and THL−*sum* (blue line), $\mathbf{W}^{softmax}$ and THL−*maxcos* (magenta line), $\mathbf{W}^{random}$ before optimizing (yellow dashed line), or $\mathbf{W}^{random}$ and THL−*sum* (green dashed line).

Table 5.S3: Angular Fisher score of different sets of prototypes: initial matrix $\mathbf{W}^{softmax}$, optimized $\mathbf{W}^{softmax}$ with THL−*sum*, optimized $\mathbf{W}^{softmax}$ with THL−*maxcos*, initial $\mathbf{W}^{random}$, or optimized $\mathbf{W}^{random}$ with THL−*sum*. For the optimized sets of prototypes, we also include the selected iteration from the Thomson optimization algorithm.

Matrix	initial $\mathbf{W}^{softmax}$	$\mathbf{W}^{softmax}$ THL−*sum*	$\mathbf{W}^{softmax}$ THL−*maxcos*	initial $\mathbf{W}^{random}$	$\mathbf{W}^{random}$ THL−*sum*
Iteration	−−	1130	50,000	−−	546
AFS	0.9524	0.9073	0.9441	1.7031	0.9309

Cross-validation performance and optimal set of prototypes

We then trained our neural network models CNN-GRU and ResCNN-GRU using the LMCL function and a fixed matrix of prototypes in the classification layer. In Figure 5.S4 we compare the cross-validation performance of three optimized matrices by Thomson: $\mathbf{W}^{softmax}$ from each model with either THL−*sum* or THL−*maxcos*, and $\mathbf{W}^{random}$ with THL−*sum*. Here we applied the tanh activation in the embedding layer and used a scale $s = 30$ in the LMCL function. These results show that the set of prototypes derived from the $\mathbf{W}^{softmax}$ matrix with the THL−*sum* loss function yields a better fold classification performance than the other two options.

Finally, we repeated the process for the rest of neural network models (considering their own matrix $\mathbf{W}^{softmax}$) and obtained the set of optimized prototypes by minimizing

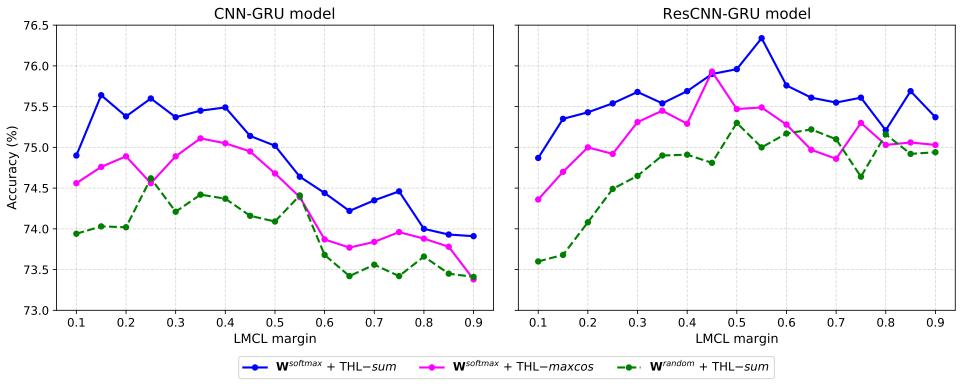


Figure 5.S4: Cross-validation fold classification accuracy (%) results for different LMCL margins (with $s = 30$). The results are provided for the CNN-GRU and ResCNN-GRU models trained with Thomson LMCL. Here, we compare three options for the Thomson-optimized set of prototypes: initial matrix $\mathbf{W}^{softmax}$ and THL−*sum* (blue line), $\mathbf{W}^{softmax}$ and THL−*maxcos* (magenta line), or $\mathbf{W}^{random}$ and THL−*sum* (green dashed line).

the THL−*sum*. The Thomson optimization curves and the optimal iteration for each model can be found in Figure 5.S5. These optimized matrices are the ones we used to train the neural network models with the Thomson LMCL option.

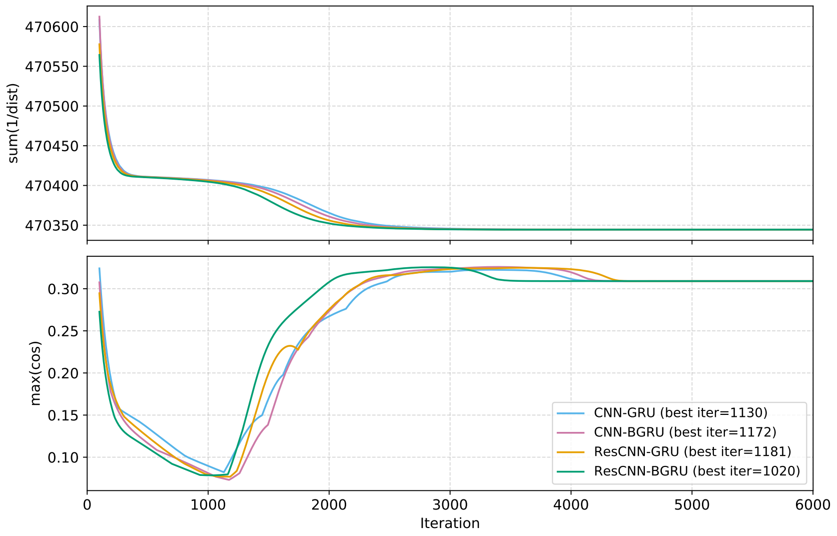


Figure 5.S5: Thomson optimization curves at each iteration monitoring two metrics: sum of inverse of distances (above) and maximum cosine similarity (below) between all pairs of prototypes. We minimized the THL−*sum* considering $\mathbf{W}^{softmax}$ as initial matrix, trained from the models: CNN-GRU (soft blue line), the CNN-BGRU (soft pink line), the ResCNN-GRU (soft yellow line), or ResCNN-BGRU (soft green line).

5.S.3 Effect of secondary structure predictions on performance

In this work we represented the protein domain using a set of 45 features for each amino acid residue in the sequence, which include a one-hot encoding of the amino acid, the PSSM profile, as well as secondary structure and solvent accessibility predictions. As other methods from the bibliography [42, 48], we used the SSPro/ACCPro programs from SCRATCH-1D [62] to obtain predictions for the secondary structure and solvent accessibility. However, it must be noted that SSPro/ACCPro use homology analysis, so when the protein domain can be found in the PDB database they provide nearly perfect predictions of these features. In order to study the impact of using different predictors, we replaced our predictions with those given by SSPro/ACCPro "ab-initio" (i.e. without homology analysis)

Table 5.S4: Effect of secondary structure and solvent accessibility predictions on FoldHSphere performance using the LINDAHL dataset. The fold recognition accuracy (%) results are provided at the family, superfamily and fold levels, considering both the top 1 and top 5 ranked templates. We compare the predictions given by SCRATCH (homology) [62], SCRATCH-AB (ab-initio) [62] and NetSurfP-2.0 (hhblits) [71] on the pre-trained ResCNN-GRU and ResCNN-BGRU neural network models, using different loss functions: **(a)** Softmax loss with sigmoid activation, **(b)** LMCL with tanh activation, and **(c)** Thomson LMCL with tanh activation.

Model	SS/ACC	Family		Superfamily		Fold	
	predictor	Top 1	Top 5	Top 1	Top 5	Top 1	Top 5
(a) Softmax Loss							
ResCNN-GRU	SCRATCH	72.6	90.3	59.4	77.0	58.9	75.1
	SCRATCH-AB	63.1	83.4	47.5	67.1	47.4	67.0
	NetSurfP-2.0	72.1	88.8	56.5	74.4	62.6	75.7
ResCNN-BGRU	SCRATCH	76.8	91.2	65.0	82.0	59.5	76.6
	SCRATCH-AB	67.2	84.0	52.3	69.1	46.1	67.3
	NetSurfP-2.0	73.9	87.4	56.7	75.3	55.1	74.1
(b) LMCL							
ResCNN-GRU	SCRATCH	75.7	89.7	66.4	81.1	67.6	80.1
	SCRATCH-AB	68.1	82.7	58.1	71.9	53.9	69.5
	NetSurfP-2.0	74.2	89.5	62.0	77.4	70.4	78.8
ResCNN-BGRU	SCRATCH	75.1	89.5	69.8	85.3	74.1	82.2
	SCRATCH-AB	69.2	84.9	60.4	73.3	60.7	72.6
	NetSurfP-2.0	75.1	90.1	64.3	84.1	70.4	80.1
(c) Thomson LMCL							
ResCNN-GRU	SCRATCH	76.9	89.5	69.1	82.9	69.5	79.4
	SCRATCH-AB	69.0	84.0	56.7	72.4	54.5	71.7
	NetSurfP-2.0	78.0	89.2	65.4	81.8	69.5	78.8
ResCNN-BGRU	SCRATCH	76.4	89.2	72.8	86.4	75.1	84.1
	SCRATCH-AB	68.5	84.1	62.7	79.0	60.4	71.7
	NetSurfP-2.0	75.5	88.3	66.8	83.6	67.3	81.0

and NetSurfP-2.0 (hhblits) [71]. As the NetSurfP-2.0 method has been introduced quite recently, we expect it to provide better results than SSPro/ACCPro "ab-initio" and closer to the ones given by SSPro/ACCPro "homology".

Table 5.S4 includes the results obtained at the test phase (using LINDAHL) by our ResCNN-GRU and ResCNN-BGRU models trained using either softmax loss, LMCL or Thomson LMCL. As we can see, a performance drop is shown when deactivating the homology analysis (SCRATCH-AB) but, in general, the proposed losses achieve better performance than softmax. NetSurfP-2.0, on the other hand, provides more similar results to those given by SCRATCH (homology), especially if we consider the top 5 accuracy results. However, it is difficult to draw conclusions without re-training our models using such predictions. Differences in performance could be explained by the fact that, in order to predict the fold class, some models might be more robust to secondary structure prediction errors than others.

5.S.4 Analysis of the hyperspherical embeddings

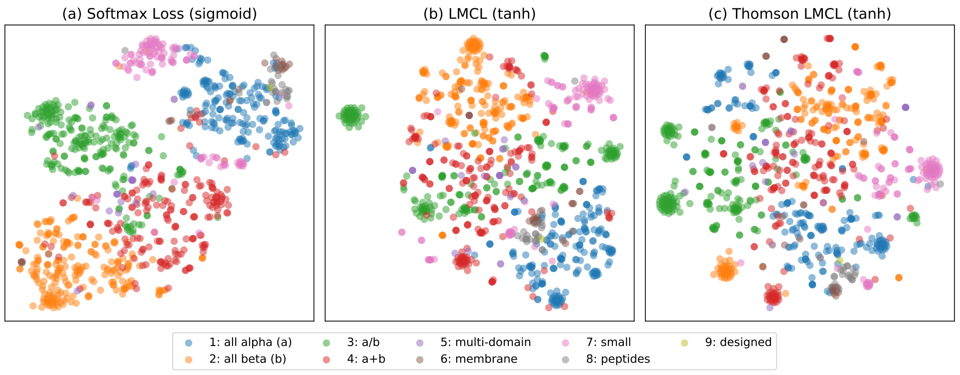


Figure 5.S6: Visualization of the embedding space learned by the ResCNN-BGRU model trained with either (a) softmax loss with sigmoid activation, (b) LMCL with tanh activation, and (c) Thomson LMCL with tanh activation. Here, the 976 embeddings within the LINDAHL dataset have been projected into two dimensions by means of t-distributed stochastic neighbor embedding (t-SNE) [72], using a perplexity of 50 and 'cosine' as metric. The resulting points have been colored according to the structural class of each domain, which are named from 1 to 9 in SCOP 1.37 [7].

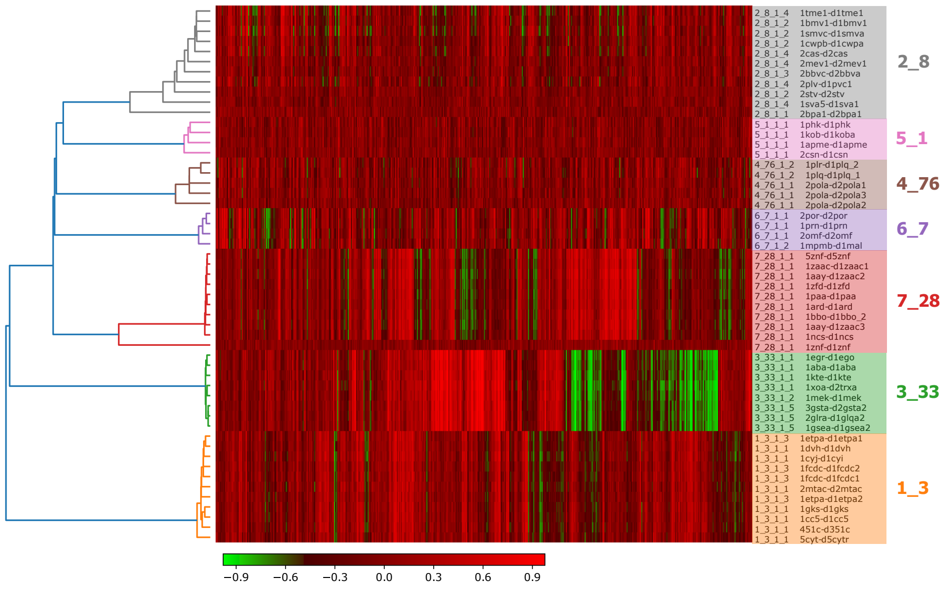


Figure 5.S7: Dendroheatmap of the 512-dimensional hyperspherical embeddings extracted from the ResCNN-BGRU model trained with Thomson LMCL ($s = 30$ and $m = 0.6$). The analysis has been done by running bi-clustering over 53 protein domains from the LINDAHL test set, covering 7 folds named 1_3, 2_8, 3_33, 4_76, 5_1, 6_7 and 7_28. We computed the cosine distance between embedding vectors (rows) and embedding components (columns) separately. We then applied hierarchical clustering with single linkage to group similar vectors and components together. The individual elements in each embedding vector are colored according to their values (lower values in green and higher values in red). Note that the legend values range from -1 to 1, as the embeddings were extracted after applying the tanh activation function. We can see how the protein domains cluster together according to their embedding vectors into 7 differentiated clusters, one for each selected fold.

5.S.5 Implementation details

We implemented our neural network models using Pytorch [73] (version 1.4.0) and Pytorch Lightning [74] (version 0.10.0), and were trained on a single GPU card (NVIDIA GTX Titan X, 12GB). The Thomson optimization algorithm was also implemented in Pytorch and run on a single GPU card. On the other hand, we used the Python Scikit-learn [75] package (version 0.23.1) implementations to train the random forest models and visualize the embedding space by means of t-SNE. Source code and data needed to reproduce the results of this paper can be found in http://sigmat.ugr.es/~amelia/FoldHSphere.

6

An analysis of protein language model embeddings for fold prediction

Amelia Villegas-Morcillo, Angel M. Gomez, and Victoria Sanchez

This chapter has been published as follows: A. Villegas-Morcillo, A.M. Gomez, and V. Sanchez. An analysis of protein language model embeddings for fold prediction. *Briefings in Bioinformatics*, 23(3):bbac142, 2022. DOI: 10.1093/bib/bbac142.

- Status: Received in February 2022. Accepted in March 2022. Published in May 2022.
- Impact Factor (JCR 2021): 13.994. Mathematical & Computational Biology. Rank 1/57 (D1).

Abstract

The identification of the protein fold class is a challenging problem in structural biology. Recent computational methods for fold prediction leverage deep learning techniques to extract protein fold-representative embeddings mainly using evolutionary information in the form of multiple sequence alignment (MSA) as input source. In contrast, protein language models (LM) have reshaped the field thanks to their ability to learn efficient protein representations (protein-LM embeddings) from purely sequential information in a self-supervised manner. In this paper, we analyze a framework for protein fold prediction using pre-trained protein-LM embeddings as input to several fine-tuning neural network models which are supervisedly trained with fold labels. In particular, we compare the performance of six protein-LM embeddings: the LSTM-based UniRep and SeqVec, and the transformer-based ESM-1b, ESM-MSA, ProtBERT, and ProtT5; as well as three neural networks: Multi-Layer Perceptron (MLP), ResCNN-BGRU (RBG), and Light-Attention (LAT). We separately evaluated the pairwise fold recognition (PFR) and direct fold classification (DFC) tasks on well-known benchmark datasets. The results indicate that the combination of transformer-based embeddings, particularly those obtained at amino acid-level, with the RBG and LAT fine-tuning models performs remarkably well in both tasks. To further increase prediction accuracy, we propose several ensemble strategies for PFR and DFC, which provide a significant performance boost over the current state-of-the-art results. All this suggests that moving from traditional protein representations to protein-LM embeddings is a very promising approach to protein fold-related tasks.

Key words: *Protein Fold Prediction, Protein Language Models, Fine-Tuning Neural Networks, Embedding Learning*

6.1 Introduction

Despite recent breakthroughs in predicting protein three-dimensional structures with high accuracy (AlphaFold [1, 2] and RoseTTAFold [3]), there is still a special interest in identifying the fold type of a protein. Although fold identification can be seen as an intermediate step in the prediction of the protein tertiary structure, this information also allows us to understand protein function through the identification of remote evolutionary relationships [4]. This step is crucial for a complete annotation of the newly solved structures (e.g. AlphaFold DB [5]), and it is often accomplished by classifying them according to structural and sequential similarities with respect to known proteins. In this regard, structural classification databases such as SCOP [6, 7] and CATH [8, 9] already provide a hierarchical grouping of protein domains from the protein data bank (PDB) [10, 11] into different categories. In SCOP these are named structural class, fold, superfamily and family, with increasing amino acid sequence similarity and closer evolutionary relationships at each level. Among them, the focus is on obtaining accurate predictions at the fold level, where protein domains have a similar arrangement of structural elements but substantially differ in the amino acid sequence, a problem widely known as protein fold recognition [12–16].

During the past few decades, many computational methods have been proposed to predict the fold class of a protein domain. These can be divided according to the task they aim to solve. The first one is *pairwise fold recognition* (PFR), in which the fold class of the query protein is inferred by comparing with templates with known structure [17–19]. PFR approaches mainly include methods based on homology modeling (sequence alignments [14], profile alignments [20], and Markov random fields [21]); threading [22–28]; machine learning for binary classification [29–31]; multi-view learning [32–34]; and learning to rank [35–37]. Another set of methods use deep learning to learn *fold-representative embeddings*, which are then used to measure the structural similarity between two protein domains. DeepFR [38] introduced this methodology, which has been followed by more recent approaches [39–45] using either predicted contact maps, or evolutionary information as input representation of the protein. The second task is *direct fold classification* (DFC), in which the protein sequences are directly mapped into a pre-defined group of fold classes [46]. Most of the proposed methods [47–56] had used evolutionary information and machine learning to classify only a small portion of all possible SCOP folds (i.e. the most populated ones). In contrast to them, DeepSF [57] was the first method to perform fold classification into one of the more than thousand existing folds in SCOP through deep learning.

Traditionally, the state-of-the-art methods for different protein-related tasks have used representations derived from the multiple sequence alignment (MSA), such as the ones described in [58], as input source. Following a more modern approach, recent methods use protein representations that are extracted from pre-trained protein language models (protein-LMs). These models have been taken from the field of NLP (natural language processing) [59], by treating the protein sequences as sentences, and the amino acids as word equivalents. More specifically, the models learn meaningful representations of the proteins (*protein-LM embeddings*) in a self-supervised manner [60] by using the vast amount of unlabeled sequences contained in protein databases such as Swiss-Prot [61], Pfam [62], and UniRef [63] (all based on UniProt [64]); and metagenomic databases such as the big fantastic database (BFD) [65, 66]. This way, ProtVec [67], based on word2vec [68, 69], was the first method proposed to extract protein representations from their sequences. Unlike word2vec, more sophisticated protein-LMs take into account both the context and order of amino acids in the sequence through LSTM (long short-term memory) recurrent units [70]. These include methods such as UDSMProt [71] and UniRep [72], using weight-dropped LSTMs and multiplicative LSTMs, respectively; as well as two methods based on the ELMo model [73]: SeqVec [74] and the language model part from [75]. The next generation of protein-LMs come from the use of transformer architectures based on self-attention mechanisms [76]. Examples are TAPE-Transformer [77], ESM-1b [78], ESM-MSA [79] and ProtBERT [80], which have been inspired in the BERT model [81]. In addition to BERT, the ProtTrans project [80] explored the use of other five transformer architectures [82–86] for protein representation learning.

These pre-trained models allow for transfer learning to different protein-related downstream tasks in which the amount of labeled sequences in the databases is significantly smaller. By using simple supervised models, protein-LM embeddings have proven to be successful in predicting protein secondary structure, subcelullar localization, and remote homologs [74, 77, 80, 87], as well as protein function [88–90] and sequence variation [91, 92]. It has been also found that the attention layers in transformer models are able to learn protein contact map information directly from self-supervised training on sequences [78, 79, 93].

Our proposal here (summarized in Figure 6.1) is to leverage several pre-trained protein-LM embeddings for fold prediction. We hypothesize that self-supervised training of the protein-LMs might capture fold information from millions of protein sequences, and therefore the learned representations could be useful for comparison of structurally similar proteins (PFR task), as well as classification into fold classes as defined by SCOP (DFC task). To test this, we followed the same idea of the DeepSF, DeepFR, and subsequent deep learning models for fold prediction. These neural network models allow for the fine-tuning of protein-LM embeddings (here *LMEmb*) to learn new fold-representative embedding vectors (here *FoldEmb*), while performing fold classification. In both tasks, PFR and DFC, we compared the performance of different neural network architectures working on either amino acid-level or protein-level embeddings. Comparison with previous methods also allowed us

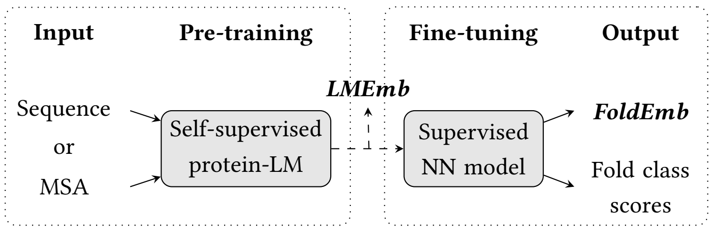


Figure 6.1: Overview of our approach for protein fold prediction. First, we extract a protein embedding representation from the amino acid sequence or multiple sequence alignment (MSA) using protein language models (protein-LMs) which have been pre-trained in a self-supervised manner (i.e. using the input sequential information itself). As a result, we obtain a protein-LM embedding (*LMEmb*) of size $L \times F$, where L is the length of the protein sequence and F is the size of the amino acid-level embedding. Then, we fine-tune this embedding through a neural network (NN) model that is trained, in a supervised manner, to map the input protein into K fold classes. The outputs of this model are, on the one hand, a fold-representative embedding of the protein (*FoldEmb* with fixed-size 512), used to perform the pairwise fold recognition (PFR) task; and, on the other hand, the scores for each fold class, used in the direct fold classification (DFC) task.

to analyze the impact of changing traditional protein evolutionary features by protein-LM embeddings as input representations. All in all, we found that transformer-based protein-LM embeddings are particularly useful for protein fold prediction, outperforming the state-of-the-art results for both fold recognition and fold classification.

6.2 Materials and methods

6.2.1 Input protein information

Our framework for fold prediction (Figure 6.1) takes as input sequential information of the protein, either the amino acid sequence or a multiple sequence alignment (MSA). To build the MSAs for our protein domains, we followed the pipeline specified in [79]. That is, we first generated the MSA by running HHblits [94] against the `uniclust30_2017_10` database [95], with number of iterations equal to 3. The resulting MSA was then subsampled by filtering the number of sequences down to 256 with hhfilter [94]. If more than 256 sequences were returned, we applied the diversity maximizing strategy from [79] to select those sequences with highest average hamming distance.

6.2.2 Pre-trained protein-LM embeddings

As protein representations, we used self-supervised embeddings from pre-trained protein language models (protein-LMs). We analyzed the performance of LSTM-based protein-LMs such as UniRep [72] and SeqVec [74], as well as several transformer-based models such as ESM-1b [78], ESM-MSA-1b (here ESM-MSA) [79], ProtBERT-BFD and ProtT5-XL-U50 (here ProtBERT and ProtT5) [80]. We denote these self-supervised embeddings as *LMEmb*, in order to differentiate them from our fine-tuned embeddings, *FoldEmb*, which are supervisedly trained using fold labels. The total size of an *LMEmb* embedding for a protein of length L is $L \times F$, where F is the size of the individual embedding provided by the protein-LM for each amino acid. By averaging the embedding matrix over the length dimension, we can obtain an alternative fixed-size representation for the protein domain (size F). We will refer to this representation as protein-level embeddings or *LMEmb-Prot* (size F) in contrast to the amino acid-level embeddings defined above and denoted as *LMEmb-AA* (size $L \times F$). Table 6.1 summarizes the training details for each protein-LM embedding included in the analysis.

LSTM-based models. UniRep [72] and SeqVec [74] are two protein-LMs trained using recurrent layers with long short-term memory (LSTM) [70] units. Both models were trained in an auto-regressive manner, trying to predict the next amino acid given all previous amino acids in a protein sequence. The *UniRep* model consists of one layer of multiplicative LSTM (mLSTM) [96] with 1900 hidden units, trained on 24 million protein sequences from

Table 6.1: Characteristics of the protein language model (protein-LM) embeddings used in this analysis.

Embeddings	Size (F)	Language Model	#Layers	#Parameters	Training Database
UniRep [72]	1900	mLSTM	1	18M	UniRef50 (24M seqs)
SeqVec [74]	1024	ELMo (BLSTM)	2	93M	UniRef50 (33M seqs)
ESM-1b [78]	1280	BERT (Transformer)	33	650M	UniRef50 (27M seqs)
ESM-MSA [79]	768	BERT (Transformer)	12	100M	UniRef50 (26M MSAs)
ProtBERT [80]	1024	BERT (Transformer)	30	420M	BFD (2B seqs)
ProtT5 [80]	1024	T5 (Transformer)	24	3B	BFD (2B seqs) + UniRef50 (45M seqs)

the UniRef50 database. We used the TAPE [77] implementation[1] and the pre-trained model from UniRep to extract $L \times 1900$ dimensional embeddings for our protein domains. On the other hand, *SeqVec* is based on the ELMo model [73] from NLP. The SeqVec model was trained on 33 million protein sequences from UniRef50. Its architecture is formed by one CharCNN layer to embed the input characters (amino acids), followed by two layers of bidirectional LSTMs (BLSTM) [97] with shared parameters for the forward and backward passes. We obtained $L \times 1024$ dimensional SeqVec embeddings by concatenating both directions of the LSTMs and then adding the outputs of the three layers. To do so, we used the official code[2] and the pre-trained model of SeqVec.

Transformer-based models. We consider two sets of transformer-based protein-LMs —evolutionary scale modeling (ESM) [78, 79] and ProtTrans [80]. The ESM models are based on the BERT transformer architecture [81]. Unlike auto-regressive LSTM-based protein-LMs, ESM models were trained to predict masked amino acids using all preceding and following amino acids in the sequence. This training objective is referred to as masked language modeling (MLM). In our analysis, we consider the pre-trained *ESM-1b* model [78], which has 33 transformer layers and a total of 650M parameters, and was trained on 27 million protein sequences from UniRef50. Instead of individual protein sequences, the *ESM-MSA* model [79] was trained on multiple sequence alignments (MSAs) constructed from sequences in UniRef50 (26 million of MSAs). This model uses the axial attention mechanism from [98] and has fewer transformer layers (12) and parameters (100M) than ESM-1b. To obtain an embedding for each protein domain in our datasets we used the official code[3] and the pre-trained ESM-1b and ESM-MSA models. For ESM-1b, we extracted amino acid-level embeddings from the last transformer layer, resulting in $L \times 1280$ vectors. In contrast, the ESM-MSA model provides embeddings for each sequence in the MSA, so the output of the last transformer layer is of size $256 \times L \times 768$. We averaged over all sequences in the MSA to obtain final embeddings of size $L \times 768$.

[1]`https://github.com/songlab-cal/tape`
[2]`https://github.com/mheinzinger/SeqVec`
[3]`https://github.com/facebookresearch/esm`

In contrast to the ESM models, the ProtTrans project [80] scaled up the transformer-based protein-LMs to leverage metagenomic databases with billions of protein sequences, leading to architectures with several billions of parameters. In this work, we consider models based on two auto-encoder transformers, BERT and T5 [84], denoted here as ProtBERT and ProtT5 respectively. The *ProtBERT* model has 30 layers and a total of 420M parameters, and has been trained on 2 billion protein sequences from the BFD database. Unlike ProtBERT and the previous ESM models, which only train the encoder component, *ProtT5* includes both the encoder and decoder, with 24 layers and a total of 3B parameters. It was trained on the BFD database and later refined using 45 million sequences from UniRef50. To extract $L \times 1024$ dimensional embeddings for our protein domains, we used the official ProtTrans implementation[4] and the pre-trained models for ProtBERT and ProtT5. Following the recommendations in [80], we only used the encoder part of ProtT5 to embed our protein domains.

6.2.3 Neural network models for protein embedding fine-tuning

In this subsection we describe the neural architectures and the corresponding supervised training procedures for fine-tuning the protein-LM embeddings *LMEmb* into the fold representative embeddings *FoldEmb* (see Figure 6.1).

Neural architectures

Our neural network models include a supervised embedding extractor followed by a linear classifier. Specifically, the embedding extractors accept as input either the amino acid-level protein-LM embeddings *LMEmb-AA* or the averaged protein-level embeddings *LMEmb-Prot*. We used three different neural architectures to extract supervised embeddings, *FoldEmb*, of fixed-size (512): a Multi-Layer Perceptron (MLP), our previously proposed Residual-Convolutional network and Bidirectional Gated Recurrent Unit (RBG) [45], and the Light-Attention (LAT) architecture from [87]. These architectures are illustrated in Figure 6.2. For the last step classifier we use a single fully-connected (FC) layer which projects the *FoldEmb* embeddings into K output elements (i.e. logits) corresponding to the fold classes. The supervised embedding extractor and classifier are trained together in an end-to-end fashion.

Multi-layer perceptron (MLP). As a reference model, we processed the *LMEmb-Prot* embeddings through a simple MLP. The architecture consists of two FC layers with 1024 and 512 neurons each (see Figure 6.2a) with ReLU activation. After each layer, we also apply batch-normalization [99] and dropout [100] with drop probability $p = 0.5$ to prevent overfitting during training.

[4] `https://github.com/agemagician/ProtTrans`

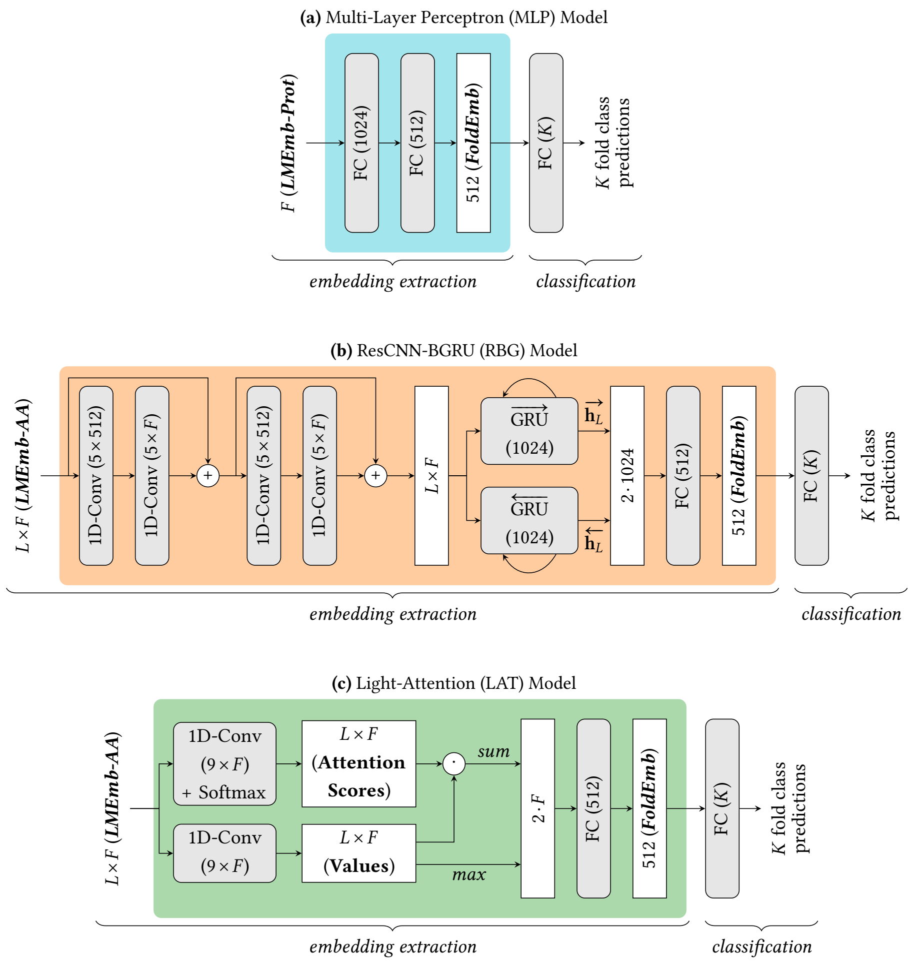


Figure 6.2: Neural network models used to fine-tune the protein-LM embeddings (*LMEmb*) to fold-representative embeddings (*FoldEmb*), as well as to perform direct fold classification. Three neural architectures are used as embedding extractors (identified with three distinct background colors). **(a)** The Multi-Layer Perceptron (MLP) model processes the protein-level embeddings (*LMEmb-Prot*) through two fully-connected (FC) layers. **(b)** The ResCNN-BGRU (RBG) model [45] processes the amino acid-level embeddings (*LMEmb-AA*) through two residual-convolutional blocks, a bidirectional gated recurrent unit (GRU) layer, and an FC layer. **(c)** The Light-Attention (LAT) model, adapted from [87], also processes *LMEmb-AA* through an attention mechanism followed by an FC layer.

Residual-convolutional network and bidirectional gated recurrent unit (RBG). To process *LMEmb-AA* embeddings, we consider our ResCNN-BGRU (RBG) model architecture (Figure 6.2b) which obtained state-of-the-art performance on protein fold recognition [45]. This architecture consists of three distinct parts, which we briefly describe here (further details can be found in [45]). First, the residual-convolutional part consists of two identical residual blocks with skip connections. Each block contains two 1D-convolutions with 512 and F filters of length 5, where F is the dimensionality of the input embedding (see Table 6.1). The 1D-convolutions are followed by ReLU activation, batch-normalization and dropout ($p = 0.2$). Second, a bidirectional recurrent layer is applied on top of the $L \times F$ outputs of the residual-convolutional part. This consists of a bi-directional gated recurrent unit (GRU) [101] layer with 1024 state units for each direction. To obtain a fixed-size vector, we concatenate the last states from both forward ($\overrightarrow{\mathbf{h}_L}$) and backward ($\overleftarrow{\mathbf{h}_L}$) GRU layers into a vector of 2048 elements. Finally, we project this vector into a 512-dimensional embedding (*FoldEmb*) using an FC layer of size 512, followed by hyperbolic tangent (tanh) activation and dropout ($p = 0.2$).

Light-attention (LAT). We also include in our analysis the Light-Attention (LAT) model from [87], which has been recently proposed for predicting protein subcellular location, using *LMEmb-AA* embeddings as inputs. The architecture is shown in Figure 6.2c. It applies two parallel 1D-convolutions with F filters of length 9, to produce the attention coefficients and value features separately. The attention weights are obtained from the coefficients by applying the softmax operation over the length dimension. The resulting weights are used to compute a weighted sum of the values, producing an F-dimensional vector independent of the protein length. This vector is then concatenated with the max-pooled values (across the length dimension) to produce a vector of size $2F$. To compute the 512-dimensional *FoldEmb* embeddings, we adapted the MLP part of the original LAT architecture by including an FC layer similar to that described above for the RBG model.

Model optimization

The fine-tuning models were trained to minimize the *large margin cosine loss* (LMCL) [102] between the predicted and true fold classes for each protein domain in the training dataset. The LMCL is an L_2-normalized and margin discriminative version of the softmax cross-entropy loss. The L_2 normalization enforces the *FoldEmb* vectors to be distributed on the surface of a hypersphere. It is formally defined as:

$$L_{lmc} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s(\cos(\theta_{y_i,i})-m)}}{e^{s(\cos(\theta_{y_i,i})-m)}+\sum_{k\neq y_i}e^{s\cos(\theta_{k,i})}}, \tag{6.1}$$

where N is the number of training samples in the mini-batch and K is the number of fold classes. The cosine is computed as $\cos(\theta_{k,i}) = \hat{\mathbf{w}}_k^T\hat{\mathbf{x}}_i$, where $\hat{\mathbf{w}}_k$ and $\hat{\mathbf{x}}_i$ are L_2-normalized versions of the k-th class weight vector $\mathbf{w}_k$ (from the last classification layer, bias is set to zero for simplicity) and the i-th embedding vector $\mathbf{x}_i$ (here the 512-dimensional *FoldEmb*). As can be noticed from Eq. 6.1, this loss introduces two hyperparameters, the scale and margin ($s, m \geq 0$). The scaling hyperparameter s controls the radius of the hypersphere on which the embeddings are distributed, while the margin m controls the decision boundaries between fold classes, and so the capacity of learning more discriminative embeddings.

Following previous results [45] and light hyperparameter tuning using cross-validation, we use the same scale for all our networks ($s = 30$) and a margin with value $m = 0.2$ for the MLP model and $m = 0.6$ for the models working on *LMEmb-AA* embeddings (that is, RBG and LAT).

For training, we use mini-batches with 64 protein domains each, and the Adam optimizer [103] with an initial learning rate equal to 10^{-3}. To prevent overfitting, along with the batch-normalization and dropout techniques specified before, we apply L_2 penalty with a weight decay of $5 \cdot 10^{-4}$. All models were trained for 80 epochs and we decreased the learning rate by a factor of 10 at epoch 40, as it proved to improve the performance in previous works [44, 45]. We implemented our models using PyTorch [104] and executed them on a single GPU (NVIDIA Tesla V100) with 32GB of memory.

6.2.4 Evaluation tasks

This subsection details the scoring procedures, ensembling strategies, and performance metrics used to evaluate the two protein fold-related tasks: pairwise fold recognition (PFR) and direct fold classification (DFC). As a reference, an evaluation where the protein-LM embeddings are directly used without fine-tuning (i.e. without additional supervised training) was also considered for both tasks.

Pairwise fold recognition (PFR)

PFR task. The pairwise fold recognition task involves evaluating the structural similarity of two protein domains. To do so, we used the 512-dimensional supervised embeddings (*FoldEmb*) extracted from our neural network models (Figure 6.2). These embeddings were used to compute a fold similarity score for each of two domains within the test set, indicating whether they belong to the same fold class or not. Following previous works, we used the cosine similarity as the similarity metric [38, 44, 45].

Ensemble strategies. Since ensemble methods have been found to be promising for PFR in previous works [38, 43–45], we also leveraged ensembling techniques here to obtain a better fold similarity score from our best performing *FoldEmb* embeddings. The first type of ensembling strategy, which we refer to as *average ensemble*, involves directly averaging the cosine similarity scores, provided by the chosen models, for each pair of protein domains in the test set. The second ensembling strategy consists of training a random forest (RF) model using our cosine similarity scores along with the 84 pairwise similarity measures from [30, 31]. We refer to this strategy as *random forest ensemble*. The RF model was trained to determine whether the protein domains in each pair share the same fold class using the vector of pairwise scores as input, and a total of 500 decision trees. Note that this strategy involves training an RF model, so we evaluated using a 10-stage cross-testing setting over the test set as in [38, 44, 45].

Ranking and evaluation. To evaluate an individual protein domain (query), we ranked the rest of domains in the test set (templates) by fold similarity score, and then assigned the fold class of the most similar one to the query. As originally proposed in [14] and following subsequent works [38, 44, 45], this evaluation was performed at three levels with increasing difficulty, according to the SCOP hierarchy—*family, superfamily* and *fold* [6]. At each level, a positive pair contains two protein domains sharing the same class in the selected level, but a different class in the level immediately below. For example, at the superfamily level, two domains within a positive pair belong to the same superfamily class (and therefore the same fold class), but different family classes. Irrespective of the level, all negative pairs were evaluated, each one containing two protein domains from different fold classes. After the ranking and fold assignment, we computed the ratio of hits (*top 1 accuracy*), as well as the ratio of finding the correct fold class within the 5 top-ranked templates (*top 5 accuracy*).

Direct fold classification (DFC)

DFC task. In the direct fold classification task, we evaluated the ability of our neural network-based models to classify the input test domains into K fold classes. This task was originally proposed in [57]. In this case, instead of extracting the supervised embeddings (*FoldEmb*), we obtained a score (i.e. logit) for each fold class from the last classification layer of our models (Figure 6.2), and the predicted fold as the one maximizing the class scoring vector.

Ensemble strategies. As with the PFR task, we combined the best performing fine-tuning models to generate several ensembles. In this case, we followed the *soft voting ensemble* approach to get a better prediction for each tested protein domain. This strategy involves computing the vector of logits from each model and accumulating them in a new class scoring vector. Then, we assign the fold which maximizes this scoring vector for each query domain.

Evaluation. We also assessed the DFC task at the *family, superfamily* and *fold* levels. In contrast to the PFR task, now the full test set is split into three subsets, each one containing test domains that only share the specific level with domains from the training dataset. As performance metric, we consider the ratio of protein domains that were correctly classified (*top 1 accuracy*), as well as the ratio of finding the correct fold within the 5 top-scoring classes (*top 5 accuracy*).

Baseline evaluation of protein-LM embeddings

To provide a baseline comparison, we also evaluated the PFR and DFC tasks directly with the pre-trained protein-LM embeddings (*LMEmb*). We used the cosine similarity metric for the protein-level embeddings *LMEmb-Prot*. In contrast, for the amino acid-level embeddings *LMEmb-AA*, we computed the soft symmetric alignment (SSA) proposed in [75] but using cosine similarity instead of L_1 distance. That is, given two sequences of embeddings $x_1, ..., x_{L_1}$ and $y_1, ..., y_{L_2}$, SSA is obtained as:

$$SSA = \frac{1}{A} \sum_{i=1}^{L_1} \sum_{j=1}^{L_2} a_{ij} \cos(x_i, y_j), \tag{6.2}$$

where $A = \sum_{i=1}^{L_1} \sum_{j=1}^{L_2} a_{ij}$ acts as a normalization factor, and $a_{ij} = \alpha_{ij} + \beta_{ij} - \alpha_{ij}\beta_{ij}$ represent the alignment matrix, whose components are computed as:

$$\alpha_{ij} = \frac{e^{\cos(x_i, y_j)}}{\sum_{k=1}^{L_2} e^{\cos(x_i, y_k)}}, \beta_{ij} = \frac{e^{\cos(x_i, y_j)}}{\sum_{k=1}^{L_1} e^{\cos(x_k, y_j)}}. \tag{6.3}$$

Note that, while the baseline PFR task is performed using pairs from the test set only, the DFC one involves computing the fold similarity metric for each test embedding against all the training ones, and then assigning the fold class of the closest training domain.

6.2.5 Datasets

To assess the aforementioned tasks, we trained the fine-tuning models using 16133 protein domains from SCOPe version v2.06 [38], which are classified into $K = 1154$ folds. For PFR, we tested the models using the well-known LINDAHL dataset [14] containing 976 protein domains from 330 distinct folds. For the DFC task, we used the updated version of LINDAHL to SCOP v1.75 [44] (named LINDAHL_1.75), where, in order to directly classify the test domains, we keep only those (of the 976) that share their fold class with one of the K seen during training. This resulted in a test set with 871 domains from 244 folds.

Table 6.2: Number of protein domains and fold classes evaluated in each test set.

Task	Test Set	Full Set		Family Level		Superfamily Level		Fold Level	
		#Domains	**#Folds**	**#Domains**	**#Folds**	**#Domains**	**#Folds**	**#Domains**	**#Folds**
PFR	LINDAHL	976	330	555	121	434	79	321	38
DFC	LINDAHL_1.75	871	244	591	177	210	107	70	37
	SCOP_2.06	2533	550	742	316	1754	418	37	15

The LINDAHL test set is evaluated on the pairwise fold recognition (PFR) task, whereas the LINDAHL_1.75 and SCOP_2.06 test sets are evaluated on the direct fold classification (DFC) task. We also provide the number of domains and folds evaluated at the family, superfamily and fold levels.

In addition, for DFC we also evaluated the performance over the SCOP_2.06 test set [57], which contains 2533 protein domains from 550 folds. To avoid overlap between this test set and the training set described above (both are derived from SCOPe v2.06), in this particular case we used the training set proposed in [57], which contains 16712 domains classified into $K = 1195$ folds (from SCOP v1.75).

For each task and test set, Table 6.2 summarizes the number of individual protein domains and distinct folds evaluated at each level (family, superfamily and fold). In all cases, the protein domains within each training set share at most 95% sequence similarity. Moreover, the maximum sequence identity within each of the test sets is 40%, as well as with respect to their respective training sets.

6.3 Results and discussion

6.3.1 Performance of self-supervised *LMEmb* embeddings in PFR and DFC tasks

We first evaluated how the self-supervised *LMEmb* embeddings perform in predicting structural similarity using the LINDAHL test set. In Table 6.3 we provide the pairwise fold recognition (PFR) results for the two types of embeddings, *LMEmb-AA* and *LMEmb-Prot*, from the 6 protein-LMs we considered. To gain insight into the PFR results in Table 6.3, we also plotted the histograms of the cosine similarity scores for the negative

Table 6.3: Performance of the *LMEmb* embeddings in the pairwise fold recognition (PFR) task, using the LINDAHL test set.

Embeddings		**Family**		**Superfamily**		**Fold**	
		Top 1	**Top 5**	**Top 1**	**Top 5**	**Top 1**	**Top 5**
LMEmb-AA ($L \times F$)	UniRep	28.5	49.2	25.3	38.9	14.3	35.8
	SeqVec	48.3	66.5	27.2	47.0	13.7	29.3
	ESM-1b	7.4	14.8	2.1	6.7	0.6	8.1
	ESM-MSA	27.9	47.7	13.8	29.5	12.1	22.4
	ProtBERT	18.9	34.8	4.4	15.9	4.4	15.9
	ProtT5	76.4	87.4	34.3	56.5	14.0	29.9
LMEmb-Prot (F)	UniRep	45.6	60.9	35.0	47.7	19.3	35.8
	SeqVec	62.3	77.8	44.9	60.6	18.4	37.7
	ESM-1b	81.6	88.5	44.9	59.7	21.2	37.1
	ESM-MSA	76.6	88.1	42.9	54.4	16.2	28.0
	ProtBERT	42.9	58.6	10.8	27.2	8.4	22.4
	ProtT5	81.1	90.8	40.3	62.0	16.5	32.4

The top 1 and top 5 accuracy (%) results are provided at the family, superfamily and fold levels. We compare the performance of the amino acid-level embeddings *LMEmb-AA* (using the SSA metric in Eq. 6.2) and protein-level embeddings *LMEmb-Prot* (using the cosine similarity metric). Underline indicates best performance.

and positive pairs within the LINDAHL test set in Supplementary Figure 6.S1. We find that aggregating embeddings across the protein length (*LMEmb-Prot*) helps in all cases when using cosine similarity. Note that, as can be seen in Supplementary Table 6.S1, using L_1 distance as comparison metric shows similar results to cosine similarity for all *LMEmb-Prot* embeddings.

Amongst the *LMEmb-Prot* embeddings, the ESM-1b ones yield better overall PFR results, while the ProtBERT embeddings perform the worst at the three levels. We also notice differences across levels. For example, the ESM-1b and ProtT5 embeddings perform the best at the family level, with a high accuracy (81% top 1). This suggests that these embeddings could be used for homology searching when the amino acid sequence similarities are high. Furthermore, ESM-1b outperforms the rest of embeddings at the fold level (21.2%). Nevertheless, the accuracy values are generally low at this level. A similar trend is observed in Supplementary Table 6.S2 for the direct fold classification (DFC) task evaluated on the LINDAHL_1.75 and SCOP_2.06 test sets. It should be noted that the overall poor performance of protein-LM embeddings at the fold level is to be expected, as they have been learned in a self-supervised manner without any information about the fold type.

6.3.2 Performance of fine-tuned *FoldEmb* embeddings in PFR task

Figure 6.3 summarizes the PFR results of the fold-representative embeddings, *FoldEmb*, resulting from fine-tuning the *LMEmb* embeddings through neural network models: MLP, RBG and LAT. As a baseline, we also include the results obtained directly using the *LMEmb-Prot* embeddings (from Table 6.3). The cosine similarity histograms for the fine-tuned *FoldEmb* embeddings can be found in Supplementary Figure 6.S1. As can be observed, the *LMEmb* embeddings can be significantly enhanced by fine-tuning even when a simple MLP model is used for the task (*LMEmb-Prot* vs MLP in Figure 6.3), especially at the superfamily and fold levels. In general, after fine-tuning, the transformer-based protein-LM embeddings (ESM-1b, ESM-MSA, ProtBERT, ProtT5) show better PFR performance than the LSTM-based ones (UniRep, SeqVec). Regarding the supervised models, the RBG and LAT, both working on *LMEmb-AA*, provide better PFR results than MLP at the superfamily and fold levels. This contrasts with the results in Table 6.3 for *LMEmb-AA*, suggesting that the RBG and LAT models successfully exploit the protein sequence information possibly contained in these self-supervised embeddings. Thus, the top-performing model at the fold level is ProtT5 + LAT, which obtains 82.6% top 1 and 88.5% top 5 accuracy values. In contrast, our ESM-MSA + RBG model provides the best PFR results at the family and superfamily levels, with 83.2% and 81.3% top 1 accuracy, respectively. It is therefore clear that the fine-tuned *FoldEmb* embeddings are necessary to identify structural similarity in the hardest cases—when the sequence similarities are low.

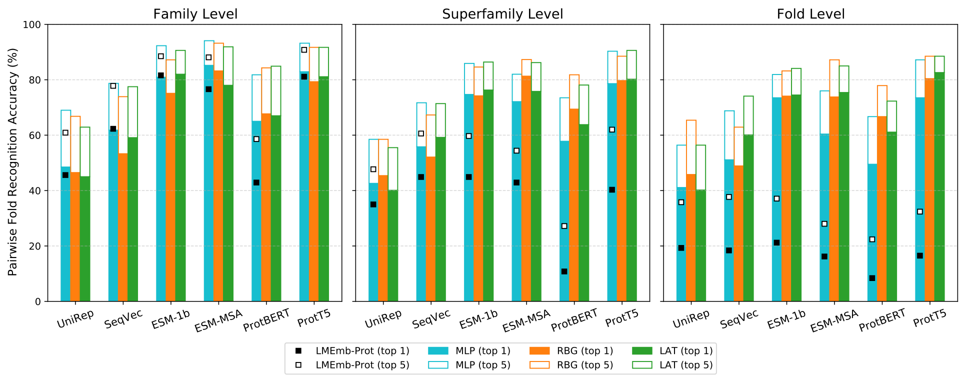


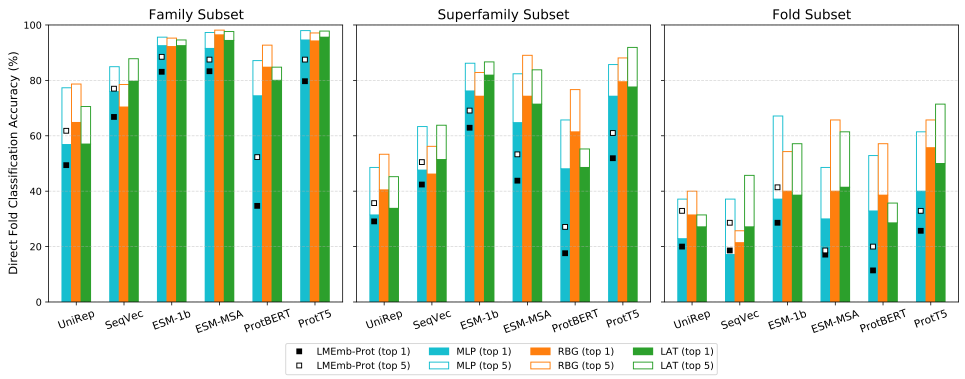
Figure 6.3: Pairwise fold recognition (PFR) accuracy (%) results on the LINDAHL test set. At each level (family, superfamily and fold), we compare the performance of the 6 protein-LM embeddings in Table 6.1 (UniRep, SeqVec, ESM-1b, ESM-MSA, ProtBERT, and ProtT5) fine-tuned by the 3 neural architectures from Figure 6.2 (MLP in cyan, RBG in orange, and LAT in green colored bars). For each one, top 1 accuracy is shown in filled bars, while top 5 accuracy is shown in empty bars. Baseline PFR results using the protein-level embeddings *LMEmb-Prot* are also included as square markers over the MLP bars (note that MLP uses these embeddings as input).

We additionally performed an ablation study (see Supplementary Table 6.S3) using the softmax cross-entropy as loss function instead of the LMCL. This modification leads to a performance drop for most combinations of input *LMEmb* and neural architecture, which is particularly noticeable at the fold level. This confirms that the LMCL is a more suitable loss function to learn a proper organization of the fold-representative *FoldEmb* vectors in the embedding space.

6.3.3 Performance of fine-tuning models in DFC task

We evaluated the ability of our neural network models to directly classify the input protein domains into fold classes (DFC task). The classification results for the LINDAHL_1.75 and SCOP_2.06 test sets are shown in Figure 6.4. As in the PFR task, we also include the results provided by the *LMEmb-Prot* embeddings as a baseline (from Supplementary Table 6.S2). The results for LINDAHL_1.75 in Figure 6.4a show a similar pattern to those in Figure 6.3 for the PFR task. However, in this case, the differences in performance between the family and fold levels are more pronounced. This behavior is expected. As test domains in the family subset share a higher sequence similarity with some domains from the training set, they are likely to be easier to predict for the models. This reinforces the idea that protein-LM embeddings capture sequence similarity in proteins. By contrast, the accuracy results at the fold level are indicative of the generalization capability of the models. Here we observe that the best performing model at both the superfamily and fold levels is ProtT5 + RBG (79.5% and 55.7% top 1 accuracy, respectively), followed by ProtT5 + LAT (77.6%

(a) LINDAHL_1.75 Test Set



(b) SCOP_2.06 Test Set

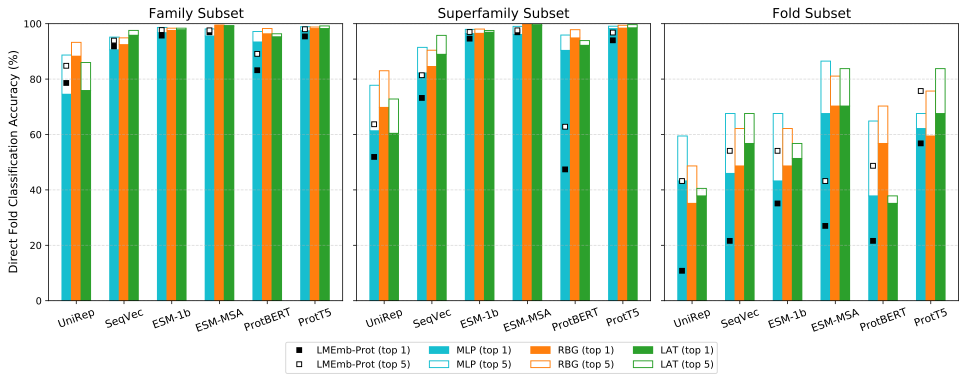


Figure 6.4: Direct fold classification (DFC) accuracy (%) results on the **(a)** LINDAHL_1.75 and **(b)** SCOP_2.06 test sets. For each subset (family, superfamily and fold), we compare the performance of the 6 protein-LM embeddings in Table 6.1 (UniRep, SeqVec, ESM-1b, ESM-MSA, ProtBERT, and ProtT5) fine-tuned by the 3 neural architectures from Figure 6.2 (MLP in cyan, RBG in orange, and LAT in green colored bars). For each one, top 1 accuracy is shown in filled bars, while top 5 accuracy is shown in empty bars. Baseline DFC results using the protein-level embeddings *LMEmb-Prot* are also included as square markers over the MLP bars (note that MLP uses these embeddings as input).

and 50.0%). It is also worth noting the huge differences between top 1 and top 5 accuracies of some protein-LM embeddings in the fold subset. For example, for the ESM-MSA + RBG model the top 1 accuracy is 40.0%, whereas the top 5 accuracy reaches 65.7%.

Figure 6.4b shows the results of the DFC task on the SCOP_2.06 test set. As can be observed, the models predict the family and superfamily levels subsets with high accuracy (close to 100% in some cases). However, similarly to the LINDAHL_1.75 test set, the fold subset in SCOP_2.06 is also difficult to classify correctly. Nevertheless, our fine-tuned models outperform the original *LMEmb* embeddings. Compared to the rest of embeddings, ESM-MSA works particularly well here for all considered neural architectures (MLP, RBG and LAT), with top 1 accuracy values around 70% at the fold level.

6.3.4 Ensemble approaches for increased accuracy in PFR and DFC tasks

Given the good performance of the transformer-based ESM-1b, ESM-MSA, and ProtT5 embeddings processed by the RBG and LAT models, we now explore the benefits of integrating them through ensemble strategies for the two assessed tasks; using average ensemble for PFR, and soft voting ensemble for DFC. Provided that the predictions from individual models are sufficiently uncorrelated (see Supplementary Figure 6.S2), we expect an increase in performance after ensembling. In Figure 6.5 we show the accuracy results at the fold level for the ensembling integration of all 6 aforementioned models (the 3 protein-LMs by the 2 neural architectures). This approach outperforms other integration options that can be found in the Supplementary Material (Table 6.S4 for PFR and Table 6.S5 for DFC). As a reference, we also include the performance of the 6 individual models used in the ensemble.

For the PFR task, the LINDAHL test set is used to evaluate the ensemble strategy (Figure 6.5a). By simply averaging the cosine similarity scores we obtain 86.3% top 1 accuracy (93.1% top 5) at the fold level, almost 4 percentage points over the best individual model (ProtT5 + LAT). For the DFC task evaluated on LINDAHL_1.75 (Figure 6.5b), the ensemble approach obtains 57.1% top 1 accuracy at the fold level, slightly better than the best individual ProtT5 + RBG model (55.7%). This suggests that accurate classification at the fold level is still difficult even after ensembling. However, a noticeable improvement is observed when considering the top 5 accuracy (81.4% over the 65.7% in ProtT5 + RBG). Additionally, the ensemble approach yields better performance for the SCOP_2.06 test set (Figure 6.5c) in terms of both top 1 and top 5 accuracy (75.7% and 86.5%, respectively), that is, more than 5 percentage points over the best individual model (ESM-MSA + RBG).

When taking into account the family and superfamily levels (Supplementary Tables 6.S4 and 6.S5), a performance boost over the individual models is also achieved by the ensembling approach in both tasks (PFR and DFC).

6.3.5 Comparison with state-of-the-art methods for fold recognition and fold classification

Finally, we compare the performance of our best individual models, as well as the ensemble strategy we propose, against the state-of-the-art results for fold recognition and fold classification. First, we compare to several methods intended for the PFR task, which can be grouped into three categories: (i) alignment and threading methods such as PSI-BLAST [14], HHpred [20], RAPTOR [23], BoostThreader [22], SPARKS-X [24], MRFalign

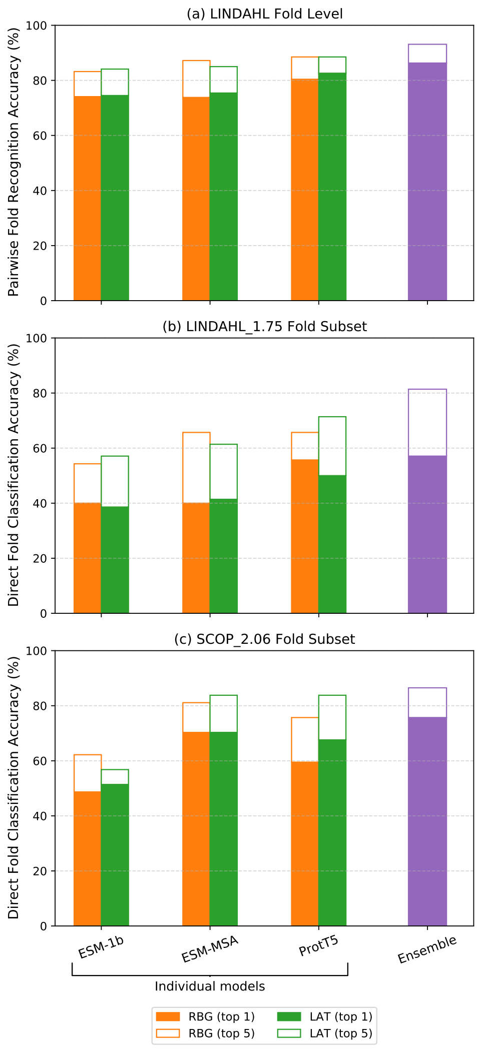


Figure 6.5: Ensemble strategy accuracy (%) results at the fold level for the **(a)** pairwise fold recognition (PFR) task using the LINDAHL test set, and the direct fold classification (DFC) task using both **(b)** LINDAHL_1.75 and **(c)** SCOP_2.06 test sets. Here the best performing ensemble strategy is shown (in purple), which involves the integration of 6 individual models. As a reference, the results of such models are also included—the 3 protein-LM embeddings (ESM-1b, ESM-MSA, and ProtT5) with neural architectures RBG (in orange) and LAT (in green). Top 1 and top 5 accuracies for each technique are represented in filled and empty bars, respectively.

[21], and CEthreader [28]; (ii) machine learning methods such as FOLDpro [29], RF-Fold [31], DN-Fold [31], RFDN-Fold [31]; and (iii) deep learning methods such as DeepFR [38], CNN-BGRU [44] VGGfold [42], FoldTR [43], and FoldHSphere [45]. Note that the PFR results of the alignment methods PSI-BLAST [14] and HHpred [20] can be fairly compared

to the performance of the *LMEmb* embeddings (in Table 6.3), as all these methods rely exclusively on sequential information. In addition, it must be noted that the CNN-BGRU and FoldHSphere methods use traditional amino acid-level features as input, including the PSSM profile matrix and one-hot encodings for each amino acid, and therefore constitute a strong baseline for comparing the performance of the protein-LM embeddings. We also provide the results of the FoldHSphere method without the Thomson-LMCL stage. That is, the RBG model (Figure 6.2b) trained on traditional features and using the LMCL as loss function [45], which we denote as PSSM/SS + RBG here.

Table 6.4 shows the PFR accuracy results achieved by these methods on the LINDAHL test set, as well as the best performing model ProtT5 + LAT and the average ensemble. We notice that ProtT5 + LAT alone outperforms all state-of-the-art methods at the superfamily and fold levels. At the family level, it also obtains better accuracy than the rest of deep learning-based methods which, at this level, tend to perform worse than alignment methods. Furthermore, as we discussed in the previous section, the ensemble strategy shows a performance boost at the fold level, but also at the family level. At this level, it achieves a

Table 6.4: Three-level LINDAHL pairwise fold recognition (PFR) results of the best individual model and the ensembling strategy (average ensemble), in comparison with the state-of-the-art.

Method	Family		Superfamily		Fold	
	Top 1	Top 5	Top 1	Top 5	Top 1	Top 5
PSI-BLAST [14]	71.2	72.3	27.4	27.9	4.0	4.7
HHpred [23]	82.9	87.1	58.0	70.0	25.2	39.4
RAPTOR [23]	**86.6**	89.3	56.3	69.0	38.2	58.7
BoostThreader [23]	86.5	90.5	66.1	76.4	42.6	57.4
SPARKS-X [24]	84.1	90.3	59.0	76.3	45.2	67.0
FOLDpro [31]	85.0	89.9	55.0	70.0	26.5	48.3
RF-Fold [31]	84.5	91.5	63.4	79.3	40.8	58.3
DN-Fold [31]	84.5	91.2	61.5	76.5	33.6	60.7
RFDN-Fold [31]	84.7	91.5	65.7	78.8	37.7	61.7
MRFalign [38]	85.2	90.8	72.4	80.9	38.6	56.7
CEthreader [44]	76.6	87.2	69.4	81.8	52.3	70.4
DeepFR (s1) [38]	67.4	80.9	47.0	63.4	44.5	62.9
DeepFR (s2) [38]	65.4	83.4	51.4	67.1	56.1	70.1
CNN-BGRU [44]	71.0	87.7	60.1	77.2	58.3	78.8
VGGfold [42]	67.9	84.3	53.2	68.4	58.3	73.5
FoldTR [43]	55.5	79.8	62.4	78.6	62.6	82.6
PSSM/SS + RBG [45]	75.1	89.5	69.8	85.3	74.1	82.2
FoldHSphere [45]	76.4	89.2	72.8	86.4	75.1	84.1
ProtT5 + LAT	81.1	91.7	80.2	90.6	82.6	88.5
Ensemble Strategy	86.5	**94.6**	**81.1**	**90.8**	**86.3**	**93.1**

The top 1 and top 5 accuracy (%) results are provided at the family, superfamily and fold levels. Boldface indicates best performance.

Table 6.5: Three-level LINDAHL pairwise fold recognition (PFR) results of the random forest (RF) ensemble, in comparison with the state-of-the-art.

Method	**Family**		**Superfamily**		**Fold**	
	Top 1	**Top 5**	**Top 1**	**Top 5**	**Top 1**	**Top 5**
DeepFRpro (s1) [38]	**85.6**	91.9	66.6	82.0	57.6	73.8
DeepFRpro (s2) [38]	83.1	92.3	69.6	82.5	66.0	78.8
CNN-BGRU-RF+ [44]	85.4	93.5	73.3	87.8	76.3	85.7
FoldTRpro [43]	83.8	92.8	76.0	89.2	58.3	87.2
FSD_XGBoostpro [43]	82.7	**94.6**	77.9	91.5	68.2	91.9
FoldHSpherePro [45]	85.2	93.0	79.0	89.2	81.3	90.3
RF Ensemble	79.6	92.8	**82.5**	**92.4**	**90.3**	**94.4**

The top 1 and top 5 accuracy (%) results are provided at the family, superfamily and fold levels. Boldface indicates best performance.

top 1 accuracy similar to the best alignment methods, and clearly outperforms all in top 5. Therefore, the use of the protein-LM embeddings as the input representation not only bridges the gap at the family and fold levels, but also increases the performance of fold recognition consistently at all levels.

In addition, in order to compare with DeepFRpro [38], CNN-BGRU-RF+ [44], FoldTRpro [43], FSD_XGBoostpro [43], and FoldHSphere [45], which apply a random forest (RF) ensemble, we implemented and tested the same RF strategy in our ensemble approach (see Materials and Methods section). It must be noted that these results cannot be directly compared to the previous ones, as this approach involves additional training of the RF models on the test set in a 10-stage cross-testing manner. From Table 6.5 we can see that the RF ensemble introduced here consistently outperforms all previous state-of-the-art methods at the superfamily and fold levels. However, it provides lower accuracy at the family level. Interestingly, this evaluation scenario seems to lead to unexpected results due to cross-testing. That is why we believe average ensembling provides more reliable results than the RF ensemble, and can be compared more fairly against the individual models.

For the DFC task we compare to deep learning methods that allow for the direct classification of protein domains into different folds. In Table 6.6 we show the results for the LINDAHL_1.75 and SCOP_2.06 test sets. Using both sets we evaluated the state-of-the-art DeepFR, CNN-BGRU, and FoldHSphere methods. In addition, for SCOP_2.06 we include the results of the DeepSF method [57], which originally introduced the SCOP_2.06 set and the DFC task. In the case of LINDAHL_1.75 (Table 6.6a), we observe that the top-performing ProtT5 + RBG model obtains better results than previous methods at all three levels, considerably outperforming them at the superfamily and fold levels. For SCOP_2.06 (Table 6.6b), the top-performing ESM-MSA + RBG model seems to fit the family and superfamily subsets almost perfectly, obtaining accuracy values above 99%. It also

Table 6.6: Three-level **(a)** LINDAHL_1.75 and **(b)** SCOP_2.06 direct fold classification (DFC) results of the best individual models and the ensembling strategy (soft voting ensemble), in comparison with the state-of-the-art.

(a) LINDAHL_1.75 Test Set

Method	**Full Set**		**Family**		**Superfamily**		**Fold**	
	Top 1	**Top 5**	**Top 1**	**Top 5**	**Top 1**	**Top 5**	**Top 1**	**Top 5**
DeepFR (s1)	57.3	72.5	70.7	83.9	34.8	55.7	11.4	25.7
DeepFR (s2)	66.5	76.0	79.5	86.5	45.2	57.6	20.0	42.9
CNN-BGRU	70.3	85.1	80.7	92.7	48.6	71.0	47.1	62.9
PSSM/SS + RBG	81.8	90.8	93.7	98.1	63.3	80.0	35.7	61.4
FoldHSphere	82.0	90.9	92.7	97.3	66.2	81.4	38.6	65.7
ProtT5 + RBG	87.6	92.4	94.3	97.1	79.5	88.1	55.7	65.7
Ensemble Strategy	**92.3**	**97.5**	**97.6**	**99.3**	**89.1**	**97.6**	**57.1**	**81.4**

(b) SCOP_2.06 Test Set

Method	**Full Set**		**Family**		**Superfamily**		**Fold**	
	Top 1	**Top 5**	**Top 1**	**Top 5**	**Top 1**	**Top 5**	**Top 1**	**Top 5**
DeepFR (s1)	89.2	94.2	88.1	93.9	91.1	95.4	24.3	37.8
DeepFR (s2)	91.2	95.6	91.1	95.8	92.3	96.4	37.8	54.1
DeepSF [57]	73.0	90.3	75.9	91.8	72.2	90.1	51.4	67.6
CNN-BGRU	89.9	96.5	91.5	96.8	90.1	97.0	48.7	64.9
PSSM/SS + RBG	96.1	98.3	96.8	98.5	96.7	98.8	54.1	70.3
FoldHSphere	96.5	98.3	97.2	98.4	97.1	98.9	51.4	67.6
ESM-MSA + RBG	99.2	99.4	**99.5**	99.5	99.7	99.8	70.3	81.1
Ensemble Strategy	**99.3**	**99.6**	99.3	**99.6**	**99.8**	**99.9**	**75.7**	**86.5**

The top 1 and top 5 accuracy (%) results are provided for the full test set, and the family, superfamily and fold subsets. Boldface indicates best performance per set.

generalizes better than previous methods in the fold subset, with a top 1 accuracy of 70.3% (81.1% top 5) which is more than 15 percentage points higher than the previous methods DeepSF and FoldHSphere (both obtained 51.5% top 1 and 65.7% top 5 accuracies). Moreover, as discussed before, the ensemble strategy already outperformed the best individual models on both test sets and, therefore, all the considered methods from the state-of-the-art.

Taken together, these results show the superiority of protein-LM embeddings over other sequence representations such as the PSSM and secondary structure (DeepSF, CNN-BGRU, and FoldHSphere), or two-dimensional representations such as contact maps (DeepFR, VGGfold, and FoldTR).

6.4 Conclusion

This work provides a comparative analysis of different pre-trained embeddings from protein language models (protein-LMs) in protein fold prediction. To do so, we fine-tuned the protein-LM embeddings (*LMEmb*) through several state-of-the-art neural network models using fold labels for supervised training. The performance was evaluated in two differentiated tasks: pairwise fold recognition (PFR) and direct fold classification (DFC). For the PFR task, we extracted a fold-representative embedding vector (*FoldEmb*), which we used to predict the protein fold class by pairwise comparison with known protein domains. In contrast, in the DFC task we directly mapped the protein domain into one of the more than thousand existing fold classes in the SCOP database. For both tasks, the protein-LM embeddings learned by pre-trained transformer-based models proved to be more effective at identifying the protein fold than those extracted from LSTM-based models. Thus, the ESM-MSA and ProtT5 amino acid-level embeddings in combination with the RBG and LAT architectures provided the best PFR and DFC results. Compared to the state-of-the-art, these models alone were able to predict the fold class with higher accuracy at the three levels—family, superfamily and fold. Furthermore, our proposed ensemble approaches provided a significant performance boost over these individual models and thus over the current state-of-the-art. These results demonstrate the suitability of protein-LM embeddings over other traditional protein representations for fold prediction.

Data availability

Input data, protein-LM embeddings, and trained models can be downloaded from `http://sigmat.ugr.es/~amelia/FoldEmbeddings/`. The source code used in this paper is available at `https://github.com/amelvim/FoldEmbeddings`.

Funding

This work has been supported by the project PID2019-104206GB-I00 funded by MCIN/ AEI /10.13039/ 501100011033, as well as the FPI grant BES-2017-079792.

Acknowledgments

Amelia would like to thank Chirag Raman for his valuable input and comments on the manuscript.

Key points

- Pre-trained protein language models (protein-LMs) provide us with embedding representations that can be efficiently used in a wide range of protein-related tasks, including protein fold prediction.
- While protein-LM embeddings can be used for homology searching when the amino acid sequence similarities are high, fine-tuned embeddings are necessary to predict structural similarity in the most difficult cases (i.e. when the sequence similarities are low).

- Despite amino acid-level protein-LM embeddings alone fail to identify the protein fold class, they yield the best results after being fine-tuned by the RBG and LAT models.
- When compared to the state-of-the-art methods for fold prediction, the use of protein-LM embeddings proves superior to more traditional protein representations based on multiple sequence alignments (MSA) as input source.

References

[1] A. W. Senior, R. Evans, J. Jumper, et al. Improved protein structure prediction using potentials from deep learning. *Nature*, 577(7792):706–710, 2020.

[2] J. Jumper, R. Evans, A. Pritzel, et al. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, 2021.

[3] M. Baek, F. DiMaio, I. Anishchenko, et al. Accurate prediction of protein structures and interactions using a three-track neural network. *Science*, 373(6557):871–876, 2021.

[4] D. Whitford. *Proteins: structure and function.* John Wiley & Sons, 2013.

[5] M. Varadi, S. Anyango, M. Deshpande, et al. AlphaFold Protein Structure Database: massively expanding the structural coverage of protein-sequence space with high-accuracy models. *Nucleic Acids Research*, 2021.

[6] A. G. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia. SCOP: A structural classification of proteins database for the investigation of sequences and structures. *journal of Molecular Biology*, 247(4):536–540, 1995.

[7] J.-M. Chandonia, L. Guan, S. Lin, et al. SCOPe: improvements to the structural classification of proteins–extended database to facilitate variant interpretation and machine learning. *Nucleic Acids Research*, 50(D1):D553–D559, 2022.

[8] C. A. Orengo, A. D. Michie, S. Jones, et al. CATH — a hierarchic classification of protein domain structures. *Structure*, 5(8):1093–1109, 1997.

[9] I. Sillitoe, N. Bordin, N. Dawson, et al. CATH: increased structural coverage of functional space. *Nucleic acids research*, 49(D1):D266–D273, 2021.

[10] H. M. Berman, J. Westbrook, Z. Feng, et al. The protein data bank. *Nucleic Acids Research*, 28(1):235–242, 2000.

[11] S. K. Burley, C. Bhikadiya, C. Bi, et al. RCSB Protein Data Bank: powerful new tools for exploring 3d structures of biological macromolecules for basic and applied research and education in fundamental biology, biomedicine, biotechnology, bioengineering and energy sciences. *Nucleic Acids Research*, 49(D1):D437–D451, 2021.

[12] C. Chothia and A. V. Finkelstein. The classification and origins of protein folding patterns. *Annual Review of Biochemistry*, 59(1):1007–1035, 1990.

[13] D. T. Jones, W. R. Taylor, and J. M. Thornton. A new approach to protein fold recognition.

Nature, 358(6381):86, 1992.

[14] E. Lindahl and A. Elofsson. Identification of related proteins on family, superfamily and fold level. *Journal of Molecular Biology*, 295(3):613–625, 2000.

[15] R. D. Schaeffer and V. Daggett. Protein folds and protein folding. *Protein Engineering, Design & Selection*, 24(1-2):11–19, 2010.

[16] R. Kolodny, L. Pereyaslavets, A. O. Samson, and M. Levitt. On the universe of protein folds. *Annual Review of Biophysics*, 42:559–582, 2013.

[17] M. S. Abual-Rub and R. Abdullah. A survey of protein fold recognition algorithms. *Journal of Computer Science*, 4(9):768–776, 2008.

[18] J. Chen, M. Guo, X. Wang, and B. Liu. A comprehensive review and comparison of different computational methods for protein remote homology detection. *Briefings in Bioinformatics*, 19(2):231–244, 2018.

[19] K. Stapor, I. Roterman-Konieczna, and P. Fabian. Machine learning methods for the protein fold recognition problem. In *Machine Learning Paradigms*, volume 149, pages 101–127. Springer, 2019.

[20] J. Söding. Protein homology detection by HMM–HMM comparison. *Bioinformatics*, 21(7):951–960, 2005.

[21] J. Ma, S. Wang, Z. Wang, and J. Xu. MRFalign: Protein homology detection through alignment of Markov random fields. *PLoS Computational Biology*, 10(3):e1003500, 2014.

[22] J. Xu, M. Li, D. Kim, and Y. Xu. RAPTOR: optimal protein threading by linear programming. *Journal of Bioinformatics and Computational Biology*, 1(1):95–117, 2003.

[23] J. Peng and J. Xu. Boosting protein threading accuracy. In *Annual International Conference on Research in Computational Molecular Biology*, pages 31–45, 2009.

[24] Y. Yang, E. Faraggi, H. Zhao, and Y. Zhou. Improving protein fold recognition and template-based modeling by employing probabilistic-based matching between predicted one-dimensional structural properties of query and corresponding native properties of templates. *Bioinformatics*, 27(15):2076–2082, 2011.

[25] J. Ma, J. Peng, S. Wang, and J. Xu. A conditional neural fields model for protein threading. *Bioinformatics*, 28(12):i59–i66, 2012.

[26] J. A. Morales-Cordovilla, V. Sanchez, and M. Ratajczak. Protein alignment based on higher order conditional random fields for template-based modeling. *PLoS ONE*, 13(6):e0197912, 2018.

[27] D. W. A. Buchan and D. T. Jones. EigenTHREADER: analogous protein fold recognition by efficient contact map threading. *Bioinformatics*, 33(17):2684–2690, 2017.

[28] W. Zheng, Q. Wuyun, Y. Li, et al. Detecting distant-homology protein structures by aligning deep neural-network based contact maps. *PLoS Computational Biology*, 15(10):1–27, 2019.

[29] J. Cheng and P. Baldi. A machine learning information retrieval approach to protein fold

recognition. *Bioinformatics*, 22(12):1456–1463, 2006.

[30] T. Jo and J. Cheng. Improving protein fold recognition by random forest. *BMC Bioinformatics*, 15(11):S14, 2014.

[31] T. Jo, J. Hou, J. Eickholt, and J. Cheng. Improving protein fold recognition by deep learning networks. *Scientific Reports*, 5:17573, 2015.

[32] K. Yan, X. Fang, Y. Xu, and B. Liu. Protein fold recognition based on multi-view modeling. *Bioinformatics*, 35(17):2982–2990, 2019.

[33] K. Yan, J. W. an Yong Xu, and B. Liu. Protein fold recognition based on auto-weighted multi-view graph embedding learning model. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2020.

[34] K. Yan, J. Wen, Y. Xu, and B. Liu. MLDH-Fold: Protein fold recognition based on multi-view low-rank modeling. *Neurocomputing*, 421:127–139, 2021.

[35] B. Liu, Y. Zhu, and K. Yan. Fold-LTR-TCP: protein fold recognition based on triadic closure principle. *Briefings in Bioinformatics*, 2019.

[36] J. Shao, K. Yan, and B. Liu. FoldRec-C2C: protein fold recognition by combining cluster-to-cluster model and protein similarity network. *Briefings in Bioinformatics*, 2020.

[37] J. Shao and B. Liu. ProtFold-DFG: protein fold recognition by combining Directed Fusion Graph and PageRank algorithm. *Briefings in Bioinformatics*, 2020.

[38] J. Zhu, H. Zhang, S. C. Li, et al. Improving protein fold recognition by extracting fold-specific features from predicted residue–residue contaacts. *Bioinformatics*, 33(23):3749–3757, 2017.

[39] B. Liu, C.-C. Li, and K. Yan. DeepSVM-fold: protein fold recognition by combining support vector machines and pairwise sequence similarity scores generated by deep learning networks. *Briefings in Bioinformatics*, 2019.

[40] C.-C. Li and B. Liu. MotifCNN-fold: protein fold recognition based on fold-specific features extracted by motif-based convolutional neural networks. *Briefings in Bioinformatics*, 21(6):2133–2141, 2020.

[41] Y. Pang and B. Liu. SelfAT-Fold: protein fold recognition based on residue-based and motif-based self-attention networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2020.

[42] Y. Liu, Y.-H. Zhu, X. Song, J. Song, and D.-J. Yu. Why can deep convolutional neural networks improve protein fold recognition? a visual explanation by interpretation. *Briefings in Bioinformatics*, 2021.

[43] Y. Liu, K. Han, Y.-H. Zhu, et al. Improving protein fold recognition using triplet network and ensemble deep learning. *Briefings in Bioinformatics*, 22(6):bbab248, 2021.

[44] A. Villegas-Morcillo, A. M. Gomez, J. A. Morales-Cordovilla, and V. Sanchez. Protein fold recognition from sequences using convolutional and recurrent neural networks. *IEEE/ACM*

Transactions on Computational Biology and Bioinformatics, 18(6):2848–2854, 2021.

[45] A. Villegas-Morcillo, V. Sanchez, and A. M. Gomez. FoldHSphere: deep hyperspherical embeddings for protein fold recognition. *BMC Bioinformatics*, 22(1):1–21, 2021.

[46] L. Wei and Q. Zou. Recent progress in machine learning-based methods for protein fold recognition. *International journal of Molecular Sciences*, 17(12):2118, 2016.

[47] C. H. Q. Ding and I. Dubchak. Multi-class protein fold recognition using support vector machines and neural networks. *Bioinformatics*, 17(4):349–358, 2001.

[48] H.-B. Shen and K.-C. Chou. Ensemble classifier for protein fold pattern recognition. *Bioinformatics*, 22(14):1717–1722, 2006.

[49] Q. Dong, S. Zhou, and J. Guan. A new taxonomy-based protein fold recognition approach based on autocross-covariance transformation. *Bioinformatics*, 25(20):2655–2662, 2009.

[50] J.-Y. Yang and X. Chen. Improving taxonomy-based protein fold recognition by using global and local features. *Proteins: Structure, Function, and Bioinformatics*, 79(7):2053–2064, 2011.

[51] J. Lyons, A. Dehzangi, R. Heffernan, et al. Advancing the accuracy of protein fold recognition by utilizing profiles from hidden Markov models. *IEEE Transactions on Nanobioscience*, 14(7):761–772, 2015.

[52] D. Chen, X. Tian, B. Zhou, and J. Gao. ProFold: Protein fold classification with additional structural features and a novel ensemble classifier. *BioMed Research International*, 2016:1–10, 2016.

[53] J. Xia, Z. Peng, D. Qi, H. Mu, and J. Yang. An ensemble approach to protein fold classification by integration of template-based assignment and support vector machine classifier. *Bioinformatics*, 33(6):863–870, 2016.

[54] W. Ibrahim and M. S. Abadeh. Protein fold recognition using deep kernelized extreme learning machine and linear discriminant analysis. *Neural Computing and Applications*, 31(8):4201–4214, 2019.

[55] S. Bankapur and N. Patil. An enhanced protein fold recognition for low similarity datasets using convolutional and skip-gram features with deep neural network. *IEEE Transactions on NanoBioscience*, 20(1):42–49, 2020.

[56] W. Elhefnawy, M. Li, J. Wang, and Y. Li. DeepFrag-k: a fragment-based deep learning approach for protein fold recognition. *BMC Bioinformatics*, 21(6):1–12, 2020.

[57] J. Hou, B. Adhikari, and J. Cheng. DeepSF: deep convolutional neural network for mapping protein sequences to folds. *Bioinformatics*, 34(8):1295–1303, 2018.

[58] X. Jing, Q. Dong, D. Hong, and R. Lu. Amino acid encoding methods for protein sequences: a comprehensive review and assessment. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 17(6):1918–1931, 2019.

[59] M. Zhou, N. Duan, S. Liu, and H.-Y. Shum. Progress in neural nlp: Modeling, learning, and

reasoning. *Engineering*, 6(3):275–290, 2020.

[60] D. Ofer, N. Brandes, and M. Linial. The language of proteins: Nlp, machine learning & protein sequences. *Computational and Structural Biotechnology Journal*, 19:1750–1758, 2021.

[61] B. Boeckmann, A. Bairoch, R. Apweiler, et al. The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003. *Nucleic Acids Research*, 31(1):365–370, 2003.

[62] J. Mistry, S. Chuguransky, L. Williams, et al. Pfam: The protein families database in 2021. *Nucleic Acids Research*, 49(D1):D412–D419, 2021.

[63] B. E. Suzek, Y. Wang, H. Huang, et al. UniRef clusters: a comprehensive and scalable alternative for improving sequence similarity searches. *Bioinformatics*, 31(6):926–932, 2015.

[64] U. Consortium. UniProt: the universal protein knowledgebase in 2021. *Nucleic Acids Research*, 49(D1):D480–D489, 2021.

[65] M. Steinegger and J. Söding. Clustering huge protein sequence sets in linear time. *Nature Communications*, 9(1):1–8, 2018.

[66] M. Steinegger, M. Mirdita, and J. Söding. Protein-level assembly increases protein sequence recovery from metagenomic samples manyfold. *Nature Methods*, 16(7):603–606, 2019.

[67] E. Asgari and M. R. K. Mofrad. Continuous distributed representation of biological sequences for deep proteomics and genomics. *PLoS ONE*, 10(11):e0141287, 2015.

[68] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations*, 2013.

[69] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

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

[71] N. Strodthoff, P. Wagner, M. Wenzel, and W. Samek. UDSMProt: universal deep sequence models for protein classification. *Bioinformatics*, 36(8):2401–2409, 2020.

[72] E. C. Alley, G. Khimulya, S. Biswas, M. AlQuraishi, and G. M. Church. Unified rational protein engineering with sequence-based deep representation learning. *Nature Methods*, 16(12):1315–1322, 2019.

[73] M. E. Peters, M. Neumann, M. Iyyer, et al. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[74] M. Heinzinger, A. Elnaggar, Y. Wang, et al. Modeling aspects of the language of life through transfer-learning protein sequences. *BMC Bioinformatics*, 20(1):1–17, 2019.

[75] T. Bepler and B. Berger. Learning protein sequence embeddings using information from structure. In *International Conference on Learning Representations*, 2019.

[76] A. Vaswani, N. Shazeer, N. Parmar, et al. Attention is all you need. In *Advances in Neural*

Information Processing Systems, volume 30, pages 5998–6008, 2017.

[77] R. Rao, N. Bhattacharya, N. Thomas, et al. Evaluating protein transfer learning with TAPE. In *Advances in Neural Information Processing Systems*, 2019.

[78] A. Rives, J. Meier, T. Sercu, et al. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, 118(15):e2016239118, 2021.

[79] R. M. Rao, J. Liu, R. Verkuil, et al. MSA Transformer. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8844–8856, 2021.

[80] A. Elnaggar, M. Heinzinger, C. Dallago, et al. ProtTrans: Towards cracking the language of life's code through self-supervised deep learning and high performance computing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–16, 2021.

[81] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[82] Z. Dai, Z. Yang, Y. Yang, et al. Transformer-XL: attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, page 2978–2988, 2019.

[83] Z. Yang, Z. Dai, Y. Yang, et al. XLNet: generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, 2019.

[84] C. Raffel, N. Shazeer, A. Roberts, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

[85] Z. Lan, M. Chen, S. Goodman, et al. ALBERT: a lite BERT for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2020.

[86] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning. ELECTRA: pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*, 2020.

[87] H. Stärk, C. Dallago, M. Heinzinger, and B. Rost. Light attention predicts protein location from the language of life. *Bioinformatics Advances*, 11 2021. vbab035.

[88] A. Villegas-Morcillo, S. Makrodimitris, R. C. H. J. van Ham, et al. Unsupervised protein embeddings outperform hand-crafted sequence and structure features at predicting molecular function. *Bioinformatics*, 37(2):162–170, 2021.

[89] M. Littmann, M. Heinzinger, C. Dallago, T. Olenyi, and B. Rost. Embeddings from deep learning transfer GO annotations beyond homology. *Scientific reports*, 11(1):1–14, 2021.

[90] I. van den Bent, S. Makrodimitris, and M. Reinders. The power of universal contextualized protein embeddings in cross-species protein function prediction. *Evolutionary Bioinformatics*, 17:1–15, 2021.

[91] J. Meier, R. Rao, R. Verkuil, et al. Language models enable zero-shot prediction of the effects of mutations on protein function. In *Advances on Neural Information Processing Systems*, 2021.

[92] C. Marquet, M. Heinzinger, T. Olenyi, et al. Embeddings from protein language models predict conservation and variant effects. *Human Genetics*, pages 1–19, 2021.

[93] J. Vig, A. Madani, L. R. Varshney, et al. BERTology meets biology: Interpreting attention in protein language models. In *International Conference on Learning Representations*, 2021.

[94] M. Steinegger, M. Meier, M. Mirdita, et al. HH-suite3 for fast remote homology detection and deep protein annotation. *BMC Bioinformatics*, 20(1):1–15, 2019.

[95] M. Mirdita, L. von den Driesch, C. Galiez, et al. Uniclust databases of clustered and deeply annotated protein sequences and alignments. *Nucleic Acids Research*, 45(D1):D170–D176, 2017.

[96] B. Krause, L. Lu, I. Murray, and S. Renals. Multiplicative lstm for sequence modelling. *arXiv preprint arXiv:1609.07959*, 2016.

[97] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[98] J. Ho, N. Kalchbrenner, D. Weissenborn, and T. Salimans. Axial attention in multidimensional transformers. *arXiv preprint arXiv:1912.12180*, 2019.

[99] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *International Conference on Machine Learning*, pages 448–456, 2015.

[100] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[101] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[102] H. Wang, Y. Wang, Z. Zhou, et al. CosFace: Large margin cosine loss for deep face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5265–5274, 2018.

[103] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[104] A. Paszke, S. Gross, S. Chintala, et al. Automatic differentiation in PyTorch. In *Advances in Neural Information Processing Systems*, 2017.

6.S Supplementary material

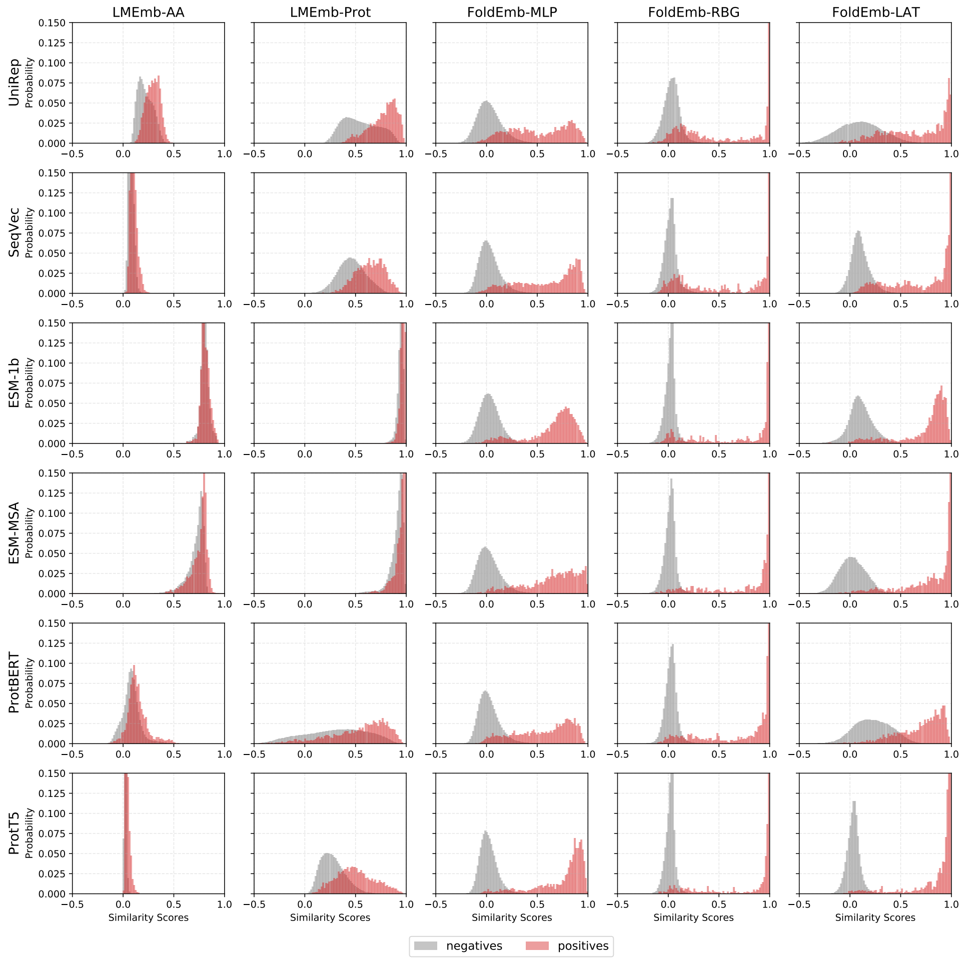


Figure 6.S1: Cosine similarity probability histograms computed for all unique pairs within the LINDAHL test set (976 domains), grouping the negative pairs in gray color, and positive pairs in red. Here we compare the scores computed over pre-trained *LMEmb-AA* (using the SSA metric), and *LMEmb-Prot* embeddings, as well as the fine-tuned *FoldEmb* embeddings using the MLP, RBG, and LAT models, for the 6 protein-LM (UniRep, SeqVec, ESM-1b, ESM-MSA, ProtBERT, and ProtT5). We observe that the *FoldEmb* embeddings provide a better separation of the two classes, especially when considering the transformer-based ESM-1b, ESM-MSA, and ProtT5 embeddings as input to the RBG and LAT fine-tuning models.

6

Table 6.S1: Performance of the *LMEmb* embeddings in the pairwise fold recognition (PFR) task, using the LINDAHL test set. The top 1 and top 5 accuracy (%) results are provided at the family, superfamily and fold levels. We compare the performance of the amino acid-level embeddings *LMEmb-AA* and protein-level embeddings *LMEmb-Prot*, using L_1 distance as comparison metric. Underline indicates best performance.

Embeddings		Family		Superfamily		Fold	
		Top 1	Top 5	Top 1	Top 5	Top 1	Top 5
LMEmb-AA ($L \times F$)	UniRep	51.4	64.0	30.0	41.7	17.1	32.1
	SeqVec	31.7	45.4	5.8	12.9	0.3	4.4
	ESM-1b	60.2	66.5	14.5	22.4	3.7	9.0
	ESM-MSA	83.1	90.1	55.8	64.5	20.2	35.8
	ProtBERT	33.2	44.3	6.0	12.0	3.4	13.4
	ProtT5	74.6	84.9	25.6	40.6	4.7	13.7
LMEmb-Prot (F)	UniRep	45.6	62.5	34.1	45.6	19.6	36.8
	SeqVec	59.1	75.0	39.9	53.9	16.8	36.1
	ESM-1b	82.2	89.0	47.0	65.7	21.2	39.6
	ESM-MSA	76.9	88.3	43.3	54.4	15.9	29.3
	ProtBERT	45.9	62.0	13.8	28.8	9.7	20.6
	ProtT5	79.8	89.4	35.0	56.0	16.2	30.5

Table 6.S2: Performance of the *LMEmb-Prot* embeddings in the direct fold classification (DFC) task, using the **(a)** LINDAHL_1.75 and **(b)** SCOP_2.06 test sets, and cosine similarity as comparison metric. The top 1 and top 5 accuracy (%) results are provided for the full test set, and the family, superfamily and fold subsets. Underline indicates best performance per set.

(a) LINDAHL_1.75 Test Set

Embeddings	Full Set		Family		Superfamily		Fold	
	Top 1	Top 5	Top 1	Top 5	Top 1	Top 5	Top 1	Top 5
UniRep	42.1	53.2	49.4	61.8	29.1	35.7	20.0	32.9
SeqVec	57.1	66.7	66.8	77.0	42.4	50.5	18.6	28.6
ESM-1b	73.8	80.0	83.1	88.5	62.9	69.1	28.6	41.4
ESM-MSA	68.4	73.7	83.3	87.5	43.8	53.3	17.1	18.6
ProtBERT	28.7	43.6	34.7	52.3	17.6	27.1	11.4	20.0
ProtT5	68.7	76.7	79.7	87.5	51.9	61.0	25.7	32.9

(b) SCOP_2.06 Test Set

Embeddings	Full Set		Family		Superfamily		Fold	
	Top 1	Top 5	Top 1	Top 5	Top 1	Top 5	Top 1	Top 5
UniRep	59.1	69.6	78.6	84.8	51.9	63.7	10.8	43.2
SeqVec	77.9	84.6	91.8	93.9	73.2	81.4	21.6	54.1
ESM-1b	94.0	96.6	95.7	97.7	94.6	97.0	35.1	54.1
ESM-MSA	95.9	96.8	96.9	97.6	96.9	97.6	27.0	43.2
ProtBERT	57.5	70.3	83.2	89.1	47.4	62.8	21.6	48.7
ProtT5	93.9	96.8	95.4	98.0	94.0	96.8	56.8	75.7

Table 6.S3: Pairwise fold recognition (PFR) top 1 accuracy (%) results on the LINDAHL test set. At each level (family, superfamily and fold), we compare the performance of the 6 protein-LM embeddings and the 3 neural architectures trained using either softmax cross-entropy loss or large margin cosine loss (LMCL). Underline indicates best performance for each loss function and model, boldface indicates best overall.

Embeddings	MLP Model			RBG Model			LAT Model		
	Family	Superfamily	Fold	Family	Superfamily	Fold	Family	Superfamily	Fold
Softmax Loss									
UniRep	49.5	38.9	28.0	36.6	40.1	37.4	34.1	37.6	29.0
SeqVec	64.3	50.7	34.6	45.8	46.3	43.0	61.4	48.6	42.4
ESM-1b	82.3	71.4	58.9	64.1	62.4	60.7	81.6	77.2	63.9
ESM-MSA	85.6	63.8	34.6	82.0	76.3	67.9	82.3	79.7	58.3
ProtBERT	68.5	45.2	29.6	56.4	54.8	49.5	69.7	54.8	30.8
ProtT5	**86.7**	77.0	60.1	68.5	68.2	73.5	**85.0**	**81.1**	64.5
LMCL									
UniRep	48.5	42.6	41.1	46.5	45.4	45.8	45.0	40.1	40.2
SeqVec	61.8	55.8	51.1	53.3	52.1	48.9	59.1	59.2	60.1
ESM-1b	80.9	74.7	**73.5**	75.1	74.2	74.1	82.0	76.3	74.5
ESM-MSA	85.2	72.1	60.4	**83.2**	**81.3**	73.8	78.0	75.8	75.4
ProtBERT	65.0	57.8	49.5	67.7	69.4	66.7	67.0	63.8	61.1
ProtT5	82.9	**78.6**	**73.5**	79.3	79.7	**80.4**	81.1	80.2	**82.6**

Table 6.S4: Ensemble results for the pairwise fold recognition (PFR) task using LINDAHL. Here we include the results of the 3 best performing protein-LM embeddings (ESM-1b, ESM-MSA, and ProtT5) with 2 neural architectures (RBG and LAT); as well as the average ensemble for different combinations of the 6 individual models. For each one, the top 1 and top 5 accuracy (%) results are provided at the family, superfamily and fold levels. Underline indicates best performance per group, boldface indicates best overall.

Embeddings	Models	Family		Superfamily		Fold	
		Top 1	Top 5	Top 1	Top 5	Top 1	Top 5
Individual Models							
ESM-1b	RBG	75.1	87.2	74.2	84.6	74.1	83.2
	LAT	82.0	90.6	76.3	86.4	74.5	84.1
ESM-MSA	RBG	83.2	93.2	81.3	87.3	73.8	87.2
	LAT	78.0	91.9	75.8	86.2	75.4	85.0
ProtT5	RBG	79.3	91.7	79.7	88.5	80.4	88.5
	LAT	81.1	91.7	80.2	90.6	82.6	88.5
Average Ensemble							
ESM-1b		81.4	89.9	75.3	84.8	75.4	86.0
ESM-MSA	RBG + LAT	84.1	93.9	80.2	86.6	76.9	88.5
ProtT5		81.4	92.4	80.2	91.0	84.7	89.7
	RBG	82.9	93.5	**83.6**	**91.2**	84.7	92.5
ESM-1b + ESM-MSA + ProtT5	LAT	**87.2**	**95.0**	80.9	90.8	82.9	91.0
	RBG + LAT	86.5	94.6	81.1	90.8	**86.3**	**93.1**

Table 6.S5: Ensemble results for the direct fold classification (DFC) task using **(a)** LINDAHL_1.75 and **(b)** SCOP_2.06. Here we include the results of the 3 best performing protein-LM embeddings (ESM-1b, ESM-MSA, and ProtT5) with 2 neural architectures (RBG and LAT); as well as the soft voting ensemble for different combinations of the 6 individual models. For each one, the top 1 and top 5 accuracy (%) results are provided for the full test set, and the family, superfamily and fold subsets. Underline indicates best performance per group, boldface indicates best overall.

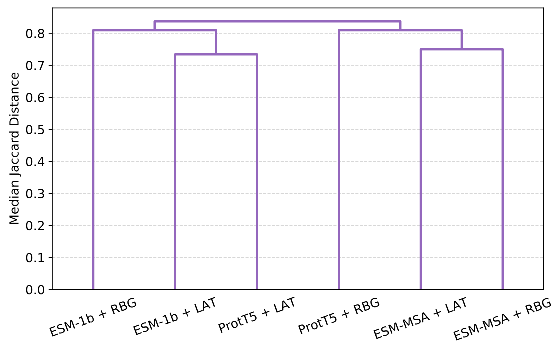
(a) LINDAHL_1.75 Test Set

Embeddings	Model	Full Set		Family		Superfamily		Fold	
		Top 1	Top 5	Top 1	Top 5	Top 1	Top 5	Top 1	Top 5
Individual Models									
ESM-1b	RBG	83.7	89.0	92.2	95.3	74.3	82.9	40.0	54.3
	LAT	85.7	89.7	92.6	94.6	81.9	86.7	38.6	57.1
ESM-MSA	RBG	86.6	93.3	96.5	98.1	74.3	89.1	40.0	65.7
	LAT	84.6	91.4	94.4	97.6	71.4	83.8	41.4	61.4
ProtT5	RBG	87.6	92.4	94.3	97.1	79.5	88.1	55.7	65.7
	LAT	87.6	94.3	95.6	97.8	77.6	91.9	50.0	71.4
Soft Voting Ensemble									
ESM-1b		85.9	90.7	93.7	96.1	77.6	86.2	44.3	58.6
ESM-MSA	RBG + LAT	88.1	94.5	97.0	97.8	79.1	91.9	40.0	74.3
ProtT5		89.8	94.4	96.3	98.0	81.9	91.9	**58.6**	71.4
	RBG	92.2	97.0	**97.8**	**99.3**	88.1	96.7	57.1	78.6
ESM-1b + ESM-MSA + ProtT5	LAT	90.8	96.3	97.5	**99.3**	85.7	95.7	50.0	72.9
	RBG + LAT	**92.3**	**97.5**	97.6	**99.3**	**89.1**	**97.6**	57.1	**81.4**

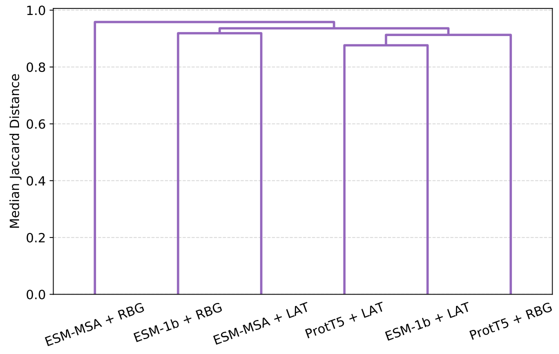
(b) SCOP_2.06 Test Set

Embeddings	Model	Full Set		Family		Superfamily		Fold	
		Top 1	Top 5	Top 1	Top 5	Top 1	Top 5	Top 1	Top 5
Individual Models									
ESM-1b	RBG	96.1	97.6	97.6	98.4	96.5	98.1	48.7	62.2
	LAT	96.5	97.2	97.8	98.4	96.9	97.6	51.4	56.8
ESM-MSA	RBG	99.2	99.4	**99.5**	99.5	99.7	99.8	70.3	81.1
	LAT	98.9	99.4	99.1	99.3	99.4	99.8	70.3	83.8
ProtT5	RBG	97.8	98.9	98.3	98.8	98.4	99.4	59.5	75.7
	LAT	98.0	99.3	98.3	99.2	98.5	99.7	67.6	83.8
Soft Voting Ensemble									
ESM-1b		97.0	98.2	98.4	99.1	97.4	98.5	51.4	67.6
ESM-MSA	RBG + LAT	99.1	99.5	99.3	99.5	99.5	**99.9**	**75.7**	83.8
ProtT5		98.5	99.2	98.9	99.3	98.8	99.5	73.0	83.8
	RBG	**99.3**	**99.6**	99.3	**99.6**	**99.8**	**99.9**	73.0	83.8
ESM-1b + ESM-MSA + ProtT5	LAT	99.2	**99.6**	99.3	**99.6**	99.7	**99.9**	70.3	83.8
	RBG + LAT	**99.3**	**99.6**	99.3	**99.6**	**99.8**	**99.9**	**75.7**	**86.5**

(a) LINDAHL Fold Level



(b) LINDAHL_1.75 Fold Subset



(c) SCOP_2.06 Fold Subset

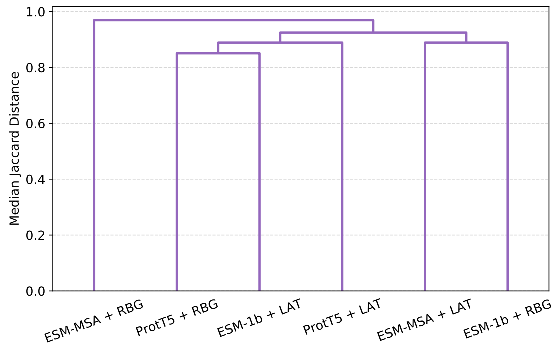


Figure 6.S2: Hierarchical clustering with complete linkage of the 6 best performing models (ESM-1b, ESM-MSA, and ProtT5 embeddings with RBG and LAT architectures). As metric, we used the median of the Jaccard distance distribution between each two models. **(a)** For the LINDAHL test set (PFR task), we compared the 50 closest neighbors of each protein domain at the fold level, using the cosine similarity scores of their fold-representative embeddings. For the **(b)** LINDAHL_1.75 and **(c)** SCOP_2.06 test sets (DFC task), we compared the 50 fold classes with maximum score for each domain in the fold subset. In all cases, the models cluster at very high values of median Jaccard distance, suggesting that the scores provided by each model are quite dissimilar.

III

Conclusions

7

Conclusions

We are just an advanced breed of monkeys
on a minor planet of a very average star.
But we can understand the universe.
That makes us something very special.

— Stephen Hawking

7.1 Conclusions

In this Thesis we aimed at developing protein fold identification systems that overcome the challenges found in previous approaches, with a view to advance the state of the art. To do so, we leveraged recent progress in deep learning techniques to learn meaningful representations of the protein fold. The main conclusions drawn from this research are listed below:

- **Native contact maps are good predictors of protein fold type.** Our results in Chapters 2 and 3 confirm that the fold-related representations learned from native contact maps outperform the previous state-of-the-art method DeepFR, which uses estimated contact maps.

- **However, relying on native structure alone is a limitation.** There is a general lack of solved protein structures, and results from Chapter 4 suggest that comparable predictions can be obtained by using information from the protein sequence itself.

- **Protein sequence can be directly exploited for fold type prediction, given a model able to handle its arbitrary length.** Our results in Chapter 4 show that the embeddings learned by our CNN-BGRU model perform better than those learned from estimated contact maps.

- **Combining protein sequence representations with estimated structure information from contact maps is, nonetheless, an effective approach.** The integration of our fold similarity score (from residue-level features) with the DeepFR's score (from estimated contact maps) along with other pairwise similarity measures provides better results than the previous state-of-the-art ensemble method DeepFR-pro (Chapter 4).

- **Learning a discriminative embedding space of the protein fold types is a promising approach when dealing with a large number of classes.** The hyperspherical embeddings learned by our FoldHSphere model described in Chapter 5 have proven to be discriminative of the protein fold and quite effective in finding template proteins even when the sequence similarities are low. Moreover, the ensemble method FoldHSpherePro yields high accuracy at the fold level, successfully bridging the performance gap between the different levels of evaluation (fold, superfamily, and family).

- **Protein language model embeddings can replace traditional representations from multiple sequence alignments (MSA) for fold type prediction.** Our experiments in Chapter 6 show that the use of protein-LM embeddings outperforms all previous state-of-the-art approaches, which mainly used representations derived from the MSA. It seems a more reasonable approach to use a deep neural network (protein-LM) that learns evolutionary relationships from the millions of protein sequences used during self-supervised training, rather than computing heuristic alignments and processing them afterwards.

- **Transformer-based protein-LM embeddings perform best at predicting fold types.** For both fold-related tasks presented in our analysis in Chapter 6, the protein-LM embeddings learned by transformer-based models have proven more effective than those extracted from LSTM-based models. In addition to scaling well to hundreds of millions of parameters, transformer models rely on the self-attention mechanism, which appears to successfully capture the relationships between amino acids within the protein.

- **Raw protein-LM embeddings are useful mainly when sequences are similar. Nevertheless, they do contain protein fold information that can be exploited for dissimilar sequences.** Results in Chapter 6 indicate that the raw embeddings could be used for homology searching when the amino acid sequence similarities are high. However, fine-tuning helps in predicting structural similarity in the most difficult cases, suggesting that these embeddings encode generally useful information about the fold type. This is in agreement with our previous work on protein function prediction, where we show that protein-LM embeddings alone provide the best results, and these are not outperformed when other structure information is included. This reinforces the idea that these embeddings already contain information about the protein fold.

Conclusiones

En esta Tesis se ha propuesto el desarrollo de sistemas de identificación del tipo de plegamiento de las proteínas que superen los desafíos encontrados en anteriores aproximaciones, con el fin de avanzar en el estado del arte. Para ello, se han aprovechado los recientes avances en técnicas de aprendizaje profundo para obtener representaciones significativas del tipo de plegamiento de la proteína. A continuación se enumeran las principales conclusiones extraídas de esta investigación:

- **Los mapas de contactos nativos son buenos predictores del tipo de plegamiento de la proteína.** Los resultados de los Capítulos 2 y 3 confirman que las representaciones relacionadas con el tipo de plegamiento aprendidas a partir de los mapas de contactos nativos superan al anterior método del estado del arte DeepFR, el cual utilizaba mapas de contacto estimados.

- **Sin embargo, basarse únicamente en la estructura nativa supone una limitación.** Existe una carencia general de estructuras de proteínas resueltas, y los resultados del Capítulo 4 sugieren que se puede obtener una predicción comparable utilizando información de la propia secuencia de la proteína.

- **La secuencia de la proteína puede ser explotada directamente para la predicción del tipo de plegamiento, dado un modelo capaz de manejar su longitud arbitraria.** Los resultados del Capítulo 4 muestran que los *embeddings* aprendidos por nuestro modelo CNN-BGRU tienen un mejor rendimiento que los aprendidos a partir de mapas de contactos estimados.

- **Combinar representaciones secuenciales de la proteína con la información estructural estimada de los mapas de contactos es, sin embargo, una aproximación efectiva.** La integración de nuestro *score* de similitud del plegamiento (a partir de características a nivel de aminoácido) con el de DeepFR (a partir de mapas de contactos estimados) junto con otras medidas de similitud por pares proporciona mejores resultados que el anterior método de *ensemble* del estado del arte DeepFRpro

(Capítulo 4).

- **Aprender un espacio de *embedding* discriminativo de los tipos de plegamiento de proteínas es un enfoque prometedor cuando tratamos con un gran número de clases.** Los *embeddings* hiperesféricos aprendidos por el modelo FoldHSphere descrito en el Capítulo 5 han demostrado ser discriminativos del plegamiento de la proteína y bastante efectivos en la búsqueda de proteínas modelo incluso cuando las similitudes en secuencia son bajas. Además, el método de *ensemble* FoldHSpherePro produce una alta precisión a nivel de plegamiento, reduciendo con éxito la brecha de rendimiento entre los diferentes niveles de evaluación (*fold, superfamily* y *family*).

- **Los *embeddings* de modelos de lenguaje (LM) aplicados a proteínas pueden reemplazar a representaciones tradicionales de alineamientos múltiples de secuencias (MSA) para la predicción del tipo de plegamiento.** Los experimentos realizados en el Capítulo 6 muestran que el uso de *protein-LM embeddings* supera todas las aproximaciones anteriores del estado del arte, las cuales utilizaban principalmente representaciones derivadas del MSA. Parece un enfoque más razonable utilizar una red neuronal profunda (*protein-LM*) que aprenda las relaciones evolutivas a partir de los millones de secuencias de proteínas utilizadas durante el entrenamiento *self-supervised*, en lugar de calcular alineamientos heurísticos y procesarlos después.

- **Los *protein-LM embeddings* basados en modelos tipo *transformer* proporcionan el mejor rendimiento a la hora de predecir el tipo de plegamiento.** Para las dos tareas de predicción del plegamiento presentadas en el análisis del Capítulo 6, los *protein-LM embeddings* aprendidos por los modelos basados en *transformer* han resultado ser más eficaces que los extraídos de los modelos tipo LSTM. Además de escalar bien a cientos de millones de parámetros, los modelos de tipo *transformer* se basan en el mecanismo de *self-attention*, el cual parece captar con éxito las relaciones entre los aminoácidos dentro de la proteína.

- **Los *protein-LM embeddings* en bruto son útiles principalmente cuando las secuencias son similares. Sin embargo, estos contienen información sobre el plegamiento de la proteína que puede ser explotada para secuencias disímiles.** Los resultados del Capítulo 6 indican que los *embeddings* en bruto podrían utilizarse para la búsqueda de homólogos cuando las similitudes de las secuencias de aminoácidos son altas. Sin embargo, el *fine-tuning* ayuda en la predicción de la similitud estructural en los casos más difíciles, lo cual sugiere que estos *embeddings* codifican información útil sobre el tipo de plegamiento. Esto concuerda con nuestro trabajo previo en la predicción de la función de la proteína, en el que mostramos que los *protein-LM embeddings* solos producen los mejores resultados, los cuales no son superados cuando se incluye otra información estructural. Esto refuerza la idea de que estos *embeddings* ya contienen información acerca del plegamiento de la proteína.

7.2 Future work

The conclusions obtained from this Thesis have opened the doorway to different research directions for advancing the field:

- **Augment the protein structure classification databases with predicted structures.** It takes a lot of effort to manually annotate the thousands of new structures that are added to the PDB database every year, let alone the almost one million structures produced by the latest structure prediction methods (e.g. AlphaFold DB). In this regard, we could extend the fold classification systems developed here to increase the pool of fold-labeled templates in the databases. We could also fine-tune the models to learn from fold predictions in a self-distillation mode (when the network transfers knowledge within the network itself), which has proven to be effective for protein 3D structure prediction.

- **Protein language models for 3D protein structure prediction.** It has been demonstrated in this Thesis that protein language models can learn meaningful representations of the protein fold. Since their full potential is yet to be determined, they could succeed in predicting the 3D coordinates of protein atoms directly without the need to rely on deep sequence alignments as has traditionally been done.

- **Further extension to other protein-related tasks.** Nowadays there exist many predictions tasks related to protein structure that are far from being solved. Examples are the structure prediction of protein complexes (two or more proteins interacting), or the prediction of disorder regions in the protein conformation (i.e. segments that lack of a fixed or stable 3D structure). We foresee that the use of proteins-LM may shed light on these problems as well, while providing a better understanding of the biology and chemistry involved in these processes.