La Aritmética del Futuro: una Reflexión sobre los Planes de Estudio

Raul Murillo, Alberto A. Del Barrio, and Guillermo Botella

Departamento de Arquitectura de Computadores y Automática,
Universidad Complutense de Madrid, España
{ramuri01,abarriog,gbotella}@ucm.es

Resumen. Con los nuevos planes de estudio, el número de horas dedicadas a la Aritmética de Computadores ha disminuido de forma notable. Sin embargo, con la explosión en la última década de las técnicas de Machine Learning y sobre todo las redes neuronales, nuevos formatos aritméticos han entrado en escena. En este trabajo hacemos un repaso de los más significativos, especialmente de los posits, introducidos en 2017 por John L. Gustafson como un reemplazo directo del punto flotante. Por tanto consideramos que en los futuros planes de estudio, la Aritmética de Computadores tiene que estar presente para garantizar que los nuevos ingenieros puedan trabajar exitosamente con estos nuevos formatos en aplicaciones críticas como las redes neuronales.

Palabras Clave: Aritmética de computadores · Posit · Punto flotante · Redes neuronales profundas.

Abstract. With the new study plans, the number of hours devoted to Computer Arithmetic have decreased. However, in the last decade, with the explosion of Machine Learning techniques and especially neural networks, new arithmetic formats have entered the scene. In this paper we review the most significant of these, especially the posits, introduced in 2017 by John L. Gustafson as a direct drop-in replacement for the floating point. We therefore consider that in future study plans, Computer Arithmetic has to be present to ensure that new engineers can successfully work with these new formats in essential applications as neural networks.

Keywords: Computer arithmetic · Posit · Floating-point · Deep neural netwoks.

1. Introducción

Históricamente la Aritmética de Computadores ha sido una rama esencial de la Informática en general y de la Arquitectura de Computadores en particular [1,2,3]. Con el auge de la Computación Aproximada y los Aceleradores Específicos de Dominio [4], la Aritmética está recobrando importancia. Especialmente,

la aparición de las redes neuronales profundas [5,6], aka *Deep Neural Networks* (DNNs), ha hecho que los investigadores reflexionen [7] sobre cuáles son los formatos más adecuados para entrenar y realizar la fase de inferencia en las DNNs. Además del IEEE 754 [8,9], han de notarse bfloat16 [10], el formato logarítmico [11,12] y los *posits* [7,13,14].

No obstante, en los actuales planes de estudio para los Grados de Informática [15,16,17,18] la importancia de la Aritmética de Computadores va en descenso. Es cierto que algunas universidades españolas han incorporado un Doble Grado en Informática y Matemáticas, pero la Aritmética desde el punto de vista computacional no se analiza con cierto nivel de detalle. La implantación de los nuevos grados de 4 años [19,20] ha obligado a fusionar asignaturas y reducir créditos, con la consecuente pérdida de contenido docente. Los contenidos estudiados quedan en su mayoría reducidos a Aritmética entera básica y a conceptos sencillos sobre el formato IEEE 754 de precisión simple (32-bits) [15,16,17,18]. Por ejemplo, en el campo de la Aritmética entera ya no se estudian multiplicadores con recodificación Booth o árboles de sumas con formato carry-save [1,2,21,22].

En este artículo planteamos una reflexión profunda respecto al estudio de la Aritmética de Computadores y evidenciamos que es un campo que está en auge y que requerirá que los nuevos graduados posean un conocimiento más profundo del mismo.

El resto del artículo se organiza de la siguiente manera: en la Sección 2 mostramos un estudio sobre la evolución del formato IEEE 754 a lo largo de los últimos años; en la Sección 3 mostramos nociones básicas sobre formatos alternativos para representar números reales en el escenario de las DNNs, prestando mayor atención a los posits; en la Sección 4 presentamos un caso de uso de aplicación de los posits y en la Sección 5 los correspondientes experimentos; finalmente en la Sección 6 mostramos las conclusiones del artículo.

2. Estado del Arte

Desde 1985 [8], el IEEE 754 se convirtió en el estándar para operaciones con números reales dentro de un computador. Desde entonces ha sufrido varias revisiones a lo largo de las últimas décadas, incluyendo en la última el estándar Half Precision, aka 16-bits, fundamentalmente debido a la demanda de las aplicaciones de Machine Learning (ML) y, en concreto, las redes neuronales profundas DNNs.

No obstante, el IEEE 754 presenta ciertos problemas inherentes a su construcción, y de los cuales muchos estudiantes no son conocedores. Entre otros, podemos citar los siguientes:

- El resultado de operaciones en punto flotante puede variar de un computador a otro. El modo de redondeo no es el mismo en todas las plataformas.
- La existencia de patrones para expresar excepciones, aka *Not a Number* (NaN).
- La existencia del doble cero ($+0$ y -0) y del doble infinito ($+\infty$ y $-\infty$).

- Cada operación se redondea individualmente, por lo que la propiedad asociativa no se mantiene.

Con el objetivo de eliminar los problemas inherentes al formato IEEE 754, John L. Gustafson propuso en 2017 [13,14] el formato de números universales versión 3, también conocido como *unum-v3* o *posit*. Además de eliminar los problemas anteriormente mencionados, los posits presentan otras ventajas, entre otras:

- Un mayor rango dinámico que punto flotante.
- Precisión cónica, por lo que son más precisos en el entorno de cero que punto flotante. Por ello, los posits son muy interesantes en DNNs [23,24,25,26,27], ya que los pesos suelen seguir una distribución gaussiana.
- La comparación entre posits se realiza como la comparación de enteros, lo cual es una gran ventaja frente a punto flotante.
- Aproximaciones rápidas de funciones no elementales, como por ejemplo el sigmoide [14].

Los posits se han presentado por tanto como un sustituto directo del punto flotante. Pese a que no son estándar por el momento, las grandes compañías como Facebook [7] IBM, Google, Intel, etc. [28] están comenzando a evaluarlo y con resultados prometedores, como apunta el ingeniero de Facebook J. Johnson [7]. Por tanto, es muy posible que en los próximos años los posits continúen aumentando su popularidad y tal vez terminen estandarizándose. De hecho, gracias a la ISA abierta RISC-V ya existen algunos cores con soporte hardware para posits [29,30].

3. Formatos Alternativos para el entrenamiento de DNNs

En esta sección vamos a comentar algunos de los formatos más utilizados en DNNs, ahora mismo una de las aplicaciones estrella dentro del ML. En concreto, nos centraremos en la fase de entrenamiento, que es la más pesada computacionalmente.

Aunque es cierto que hay muchas arquitecturas y modelos de red disponibles públicamente, también es cierto que éstas suelen utilizarse de base para realizar transfer-learning [5,7], es decir, el entrenamiento parte de unos pesos conocidos, ej. generados para ImageNet, y se recalculan para la nueva aplicación objetivo. Por tanto, es muy posible que los nuevos graduados tengan que desarrollar DNNs para aplicaciones concretas, ej. reconocimiento de células tumorales, y que, por los motivos anteriormente mencionados, los tipos de datos no sean IEEE 754 de 32-bits. En concreto, en esta sección introduciremos los formatos IEEE 754 de 16-bits, brain floating point y posit.

3.1. IEEE 754: Single y Half Precision

Tal y como comentan Hennessy y Patterson [4], el entrenamiento de DNNs típicamente se ha realizado con punto flotante de 32-bits (Single Precision),

aunque en los últimos años se ha empezado a utilizar el nuevo estándar de 16-bits (Half Precision).

El formato de representación en punto flotante se asemeja bastante a la notación científica tradicional, pues consta de tres partes: un signo S, un exponente E en una base dada β (que normalmente es 2 por tratarse de un formato binario, aunque existen otros sistemas como el de punto flotante decimal [31] que utiliza base 10) y una parte significativa o mantisa M. De esta forma, el valor de un número en punto flotante F se calcula mediante la Ecuación (1).

$$F = (-1)^S \times \beta^E \times M \ . \tag{1}$$

El estándar del IEEE para aritmética en punto flotante (IEEE 754) establecido en 1985 [8] define los formatos aritméticos y de intercambio, las reglas de redondeo, las operaciones y el manejo de excepciones para aritmética de punto flotante binaria, y está implementado en la gran mayoría de computadores modernos. En particular, el estándar IEEE 754 define los formatos de Single Precision y Half Precision según la codificación que se muestra en la Figura 1. Cabe destacar que el estándar añade un sesgo o *bias* al exponente con el fin de representar números reales tanto muy grandes como muy pequeños, de forma que se toma el exponente $e = E - bias$. Además, al valor que codifica la mantisa se le añade un bit oculto, que toma el valor 1 para los números normales y el valor 0 para los números desnormalizados, empleados para evitar el subdesbordamiento (*underflow*) en las proximidades del cero [8].

	S	Exponente E	Mantisa M
32 bits:	1 bit	8 bits, bias = 127	23 bits
16 bits:	1 bit	5 bits, bias = 15	10 bits

Figura 1: Codificación de números en punto flotante según el estándar IEEE 754.

3.2. bfloat16

Este formato, también conocido como Brain Floating Point, se trata una versión truncada (con 16-bits) del formato de punto flotante de 32-bits del IEEE 754, como se muestra en la Figura 2. Al conservar 8 bits para el exponente, el formato mantiene aproximadamente el mismo rango dinámico que el formato de 32-bits, a cambio de reducir la precisión al emplear menos bits para la fracción.

	S	Exponente E	Mantisa M
16 bits:	1 bit	8 bits, bias = 127	7 bits

Figura 2: Codificación de números en punto flotante en formato bfloat16.

Al contrario que los números en punto flotante de 32-bits, los números bfloat16 no son adecuados para realizar cálculos con números enteros debido al truncamiento de bits. No obstante, no es ese su propósito, sino que este formato se emplea para reducir el almacenamiento y aumentar la velocidad de los cómputos en tareas de ML. A pesar de ello, este formato hereda todos los problemas inherentes del IEEE 754.

3.3. Posits

En el año 2017 John L. Gustafson presentó el tipo de datos posit (también conocido como *unum Tipo III*) como una alternativa frente al para IEEE 754 para números de punto flotante [13]. Desde entonces, el uso de los posits ha sido explorado en una gran variedad de áreas, y en especial en aplicaciones de aprendizaje automático [7,23,24,25,27].

El formato posit, que se codifica según se muestra en la Figura 3 consta de los siguientes campos: signo régimen (en inglés, *regime*), exponente y fracción.

- **Signo.** Al igual que para números en punto flotante o enteros con signo, el primer bit indica el signo: 0 para números positivos, 1 para números negativos. En el caso negativo, se debe tomar el complemento a 2 de los bits restantes para extraer los siguientes campos correctamente.
- **Régimen.** Este campo es exclusivo de este formato numérico. El régimen codifica el valor k del factor de escalado del número. Este valor está determinado por el número de bits idénticos (indicados por r en la Figura 3), cuya secuencia termina con el valor negado ($\overline{r}$), si es que existe. Sea m el número de bits idénticos del régimen; si este campo consta de 0's a la izquierda, entonces $k = -m$; si por el contrario se trata de una secuencia de 1's, entonces $k = m - 1$. Por ejemplo, si el régimen tiene 4 bits de ancho, el valor `1110` se interpretaría como $k = 2$, mientras que el valor `0001` sería $k = -3$. Por lo tanto, detectar el número de 1's o 0's a la izquierda es fundamental para realizar este paso. Sin embargo, la principal dificultad para detectar el régimen y, en consecuencia, decodificar el posit, es que su longitud varía dinámicamente. De forma analítica, el valor de k viene dado por la Ecuación (2). Nótese que este campo es de longitud variable, lo que hace posible que los campos siguientes puedan no estar presentes en determinados números, en cuyo caso se les asignará el valor 0.

$$k = -x_{n-2} + \sum_{i=n-2}^{x_i \neq x_{n-2}} (-1)^{1-x_i} \tag{2}$$

- **Exponente.** Los bits de exponente (indicados en azul) codifican el valor e, empleado para el factor de escalado 2^e. A diferencia de los números en punto flotante, los posits carecen de sesgo o bias. Como la longitud del régimen es variable, puede haber hasta es bits de exponente, ya que el primer bit de este campo se ubica directamente después del campo de régimen (por lo que existe la posibilidad de que no haya bits de exponente). Además, al contrario que en

el estándar IEEE 754, la longitud de este campo no está fijada, por lo que se pueden destinar más o menos bits para el exponente según se desee (incluso puede considerarse $es = 0$). Nótese que un valor grande para es permite trabajar con mayor escala, mientras que valores pequeños de es garantizan una mayor precisión, pues se destinan más bits al campo de fracción.

- **Fracción.** Los bits restantes después del exponente corresponden al campo de fracción, y representan el valor de la fracción f. Esto es igual que para la mantisa en el caso de los números en punto flotante, pero existe una gran diferencia: en el caso de los posits el bit oculto siempre es 1, y no hay números desnormalizados con un bit oculto de 0, como en el estándar IEEE 754.

Como puede observarse, cada configuración de posits está completamente determinada por la longitud total de bits n y el número máximo de bits de exponente es. Por lo tanto, es común usar la notación Posit$\langle n, es\rangle$ para denotar una configuración de posits con n bits, de los cuales un máximo de es pertenecen al exponente.

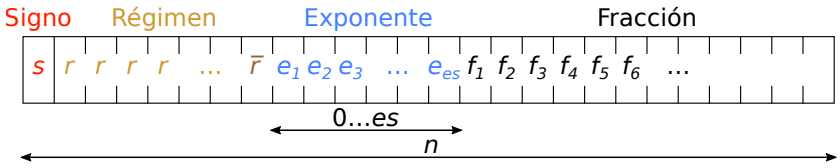


Figura 3: Codificación de un número posit $\langle n, es\rangle$.

Con todo ello, el valor numérico de un número en formato posit X, cuyos bits se distribuyen como se muestra en la Figura 3, viene dado por la Ecuación (3):

$$X = (-1)^s \times (useed)^k \times 2^e \times (1.f) \ , \tag{3}$$

donde:

$useed$ es el factor de escalado, con valor $2^{2^{es}}$,
k es el valor indicado por el régimen, dado por la Ecuación (2),
e es el valor indicado por el campo de exponente,
f es el valor indicado por el campo de fracción.

Propiedades del formato posit. Como ya se ha mencionado, el formato posit tiene algunas propiedades realmente interesantes y que hacen que este formato supere en ciertos aspectos al punto flotante:

- Un único patrón de bits para representar el 0 (todos los bits `0`) y para representar el ∞ (el primer bit `1` y el resto `0`).
- Ausencia de NaNs. De esta forma es posible representar muchos más valores con la misma cantidad de bits.

- Los posits no desbordan, ni con valores muy grandes (*overflow*) ni con valores cercanos a cero (*underflow*). A pesar de no tener un desbordamiento gradual como los números en punto flotante, los posits poseen lo que se conoce como precisión cónica.
- Operaciones fusionadas. Aunque la primera revisión del estándar IEEE [31] incluyó este tipo de operaciones, como la multiply-accumulate en la que se efectúa una multiplicación que se suma a un acumulador con un único redondeo en el proceso, en el caso de los posits para estas operaciones se define también un registro de mayor tamaño que sirve de acumulador (de forma que se evitan los redondeos intermedios) denominado *quire*.
- Bajo ciertas configuraciones (especialmente cuando $es = 0$), los posits permiten realizar aproximaciones rápidas (realizadas a nivel de bit, sin necesidad de decodificar todo el número) de funciones elementales pero complejas, como el recíproco, y de funciones no elementales, como la sigmoide o la tangente hiperbólica (estas últimas son muy utilizadas en el área de la DNNs) [32].

Inconvenientes del formato posit. Aunque hasta el momento se han presentado múltiples ventajas del formato posit, existe una serie de desventajas con respecto a otros formatos tradicionales de aritmética decimal como los de punto flotante y de punto fijo.

El principal problema que se asocia con este formato es la longitud variable del campo del régimen. Dado que no posee una longitud fija, como sí la tiene el exponente en el formato de punto flotante, no es posible decodificar paralelamente los campos de un número posit, como sí ocurre con el formato del IEEE 754. Esto supone un cierto sobre-coste, desde el punto de vista del diseño de operadores. No obstante, los defensores de este formato aseguran que el hecho de que posits con menor número de bits posean la misma precisión que números de punto flotante compensa este inconveniente [33].

Por otro lado, existen situaciones y problemas determinados, en áreas como la simulación de física de partículas en las que los posits ofrecen peores resultados que los números en punto flotante, principalmente debido al error cometido en el valor de constantes físicas del sistema internacional [34].

Finalmente, el coste de desarrollo de hardware para el formato posit, junto con la ausencia de herramientas suficientes, como compiladores, ralentizan la investigación y los avances en este novedoso formato, y hacen que sea difícil evaluar el progreso de los posits.

4. Posits y DNN: Deep PeNSieve

Deep PeNSieve [27] es un framework de código abierto[1] para realizar el entrenamiento y la inferencia de DNNs enteramente con posits, sin convertir de y a punto flotante. Como se aprecia en la Figura 4, durante el entrenamiento de la red (que puede incluir diferentes capas, incluidas convolucionales), el framework realiza todos los cómputos en formato posit.

[1] https://github.com/RaulMurillo/deep-pensieve

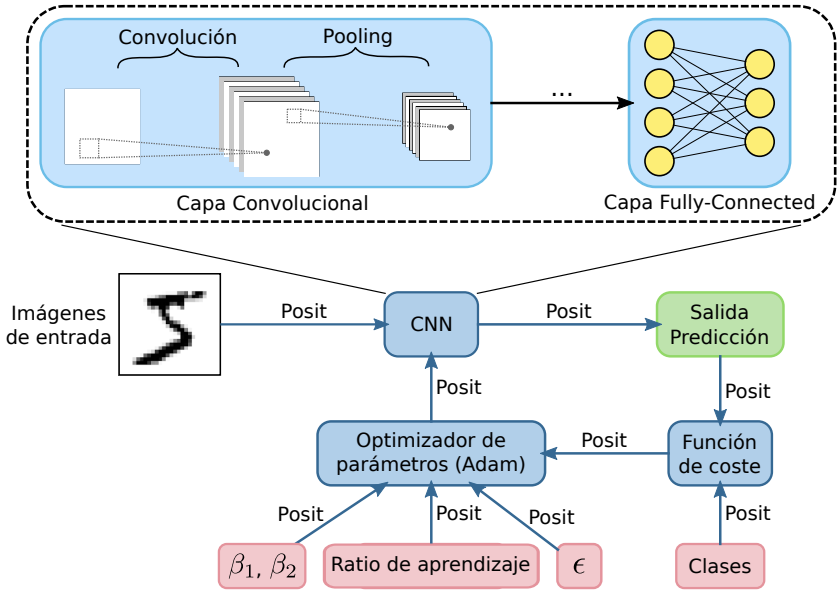


Figura 4: Flujo de entrenamiento de DNNs con Deep PeNSieve. Las cajas rojas representan los datos de entrada, mientras que las azules corresponden con las operaciones implementadas por TensorFlow.

Por otro lado, una vez se ha entrenado la red en el formato deseado, Deep PeNSieve permite realizar inferencia cuantificando los pesos internos a un formato de menor precisión. En particular, para el caso de Posit$\langle 8, 0\rangle$ permite realizar las multiplicaciones matriciales (que son la principal operación en las redes neuronales) como un producto escalar fusionado, utilizando un registro acumulador quire de 32 bits para evitar que la precisión de la red se degrade demasiado al emplear tan sólo 8 bits.

5. Experimentos

Para comprobar el desempeño del formato posit en comparación con el de punto flotante, utilizamos dos datasets clásicos: MNIST, con la arquitectura LeNet-5 [5], y CIFAR-10 [35] con la arquitectura CifarNet [27].

Para el entrenamiento se han comparado Posit$\langle 32, 2\rangle$ y Posit$\langle 16, 1\rangle$ con el formato IEEE 754 de precisión simple. Los resultados de la fase de entrenamiento, tras 30 iteraciones, se muestran en la Tabla 1. Los formatos posit muestran resultados similares a los obtenidos con punto flotante, e incluso en el caso de CIFAR-10 el formato Posit$\langle 16, 1\rangle$ incrementa la precisión de la red en un 4 %.

Con respecto a los resultados de inferencia con baja precisión, en la Tabla 2 se muestran los valores obtenidos tras cuantificar las anteriores redes neuronales de 32 bits con 16 y 8 bits. Cabe destacar que con estas cuantificaciones se obtienen

Tabla 1: Resultados tras la fase de entrenamiento de las CNN.

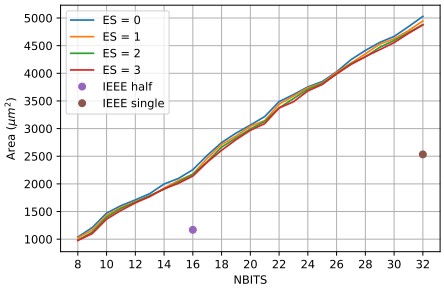
	MNIST		CIFAR-10	
Formato	Top-1	Top-5	Top-1	Top-5
Float 32	99.17 %	100 %	68.06 %	95.15 %
Posit$\langle 32, 2\rangle$	99.09 %	99.98 %	69.32 %	96.59 %
Posit$\langle 16, 1\rangle$	99.18 %	100 %	72.51 %	97.40 %

reducciones en el tamaño de los datos de $1/4$ y $1/2$, respectivamente, con respecto a los modelos originales. Los resultados obtenidos ponen de manifiesto que el uso de operaciones fusionadas con quire hace posible la inferencia en redes neuronales con posits de 8-bits sin reducir excesivamente la precisión de la red, de igual forma que con técnicas usuales de cuantización en punto flotante de 16-bits o enteros de 8-bits, pero con menos recursos que el primer formato y con menos limitaciones que el segundo.

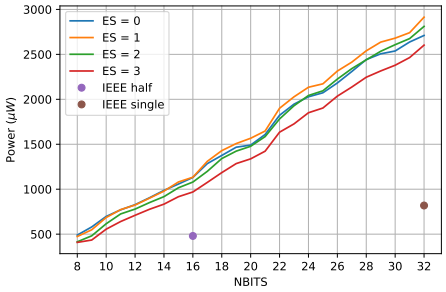
Tabla 2: Resultados de inferencia tras cuantificar las CNNs a formatos de baja precisión.

	MNIST		CIFAR-10	
Formato	Top-1	Top-5	Top-1	Top-5
Float 16	99.17 %	100 %	68.05 %	96.15 %
INT8	99.16 %	100 %	68.15 %	96.14 %
Posit$\langle 8, 0\rangle$	98.77 %	99.99 %	43.89 %	86.49 %
Posit$\langle 8, 0\rangle_{\text{quire}}$	99.07 %	99.99 %	68.88 %	96.47 %

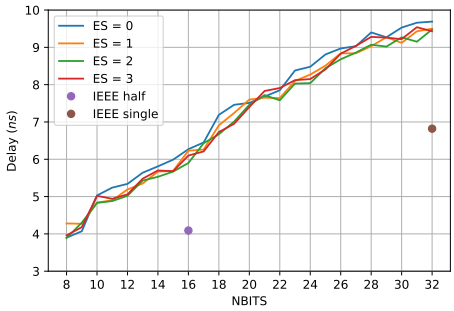
A la vista de los resultados, uno podría preguntarse si no sería más conveniente trabajar en adelante con el formato posit, al menos en el campo del ML. Sin embargo, uno de los principales problemas de este formato es que aún no existen diseños de unidades que sean competitivos con respecto al punto flotante. Para mostrar esto, se ha tomado el diseño de un sumador en formato posit de libre acceso [26] y se ha sintetizado con la herramienta Synopsys Design Compiler (utilizando una biblioteca de 65 nm) para obtener datos del área, potencia y retardo requeridos por una unidad de este tipo. Los resultados, para distintas configuraciones de bits y de longitud de exponente, y en comparación con las análogas unidades en punto flotante IEEE 754, se muestran en la Figura 5. Como se puede apreciar, un sumador posit de n-bits requiere de, aproximadamente, la misma cantidad de recursos que el sumador de punto flotante para $2n$-bits. Es por ello que aún queda margen de mejora y trabajo por realizar en este campo.



(a) Área



(b) Potencia



(c) Retardo

Figura 5: Resultados de síntesis respecto al área, potencia y retardo de un sumador posit para distintas configuraciones $\langle n, es \rangle$.

6. Conclusiones

En este trabajo se ha presentado un breve resumen del panorama actual en Aritmética para DNNs, fundamentalmente en la etapa de entrenamiento. Como puede observarse, con el nacimiento de nuevos formatos diferentes a IEEE 754, especialmente los posits, la Aritmética de Computadores está retomando su importancia.

Es por ello que consideramos esencial que los futuros planes de estudio amplíen el contenido relativo a ésta, ya que los nuevos graduados en Informática y derivados tendrán que trabajar muy posiblemente con aplicaciones basadas en DNNs, y muy posiblemente el IEEE 754 no sea el formato utilizado. Por tanto, para comprender el comportamiento de estas aplicaciones será fundamental comprender la Aritmética sobre la que se basan.

A modo de ejemplo, hemos presentado Deep PeNSieve, un framework de entrenamiento de DNNs con posits. Ahora mismo todas las operaciones son emuladas, pero la presencia cada vez mayor de la ISA RISC-V posibilitará en un futuro no muy lejano disponer de cores lógicos sobre los que ejecutar este tipo de aplicaciones de manera nativa y reducir así el tiempo de ejecución.

Referencias

1. Ercegovac, M.D., Lang, T.: Digital arithmetic. Elsevier, 1st edn. (2004)
2. Koren, I.: Computer Arithmetic Algorithms. A. K. Peters, Ltd., USA, 2nd edn. (2002)
3. Hennessy, J.L., Patterson, D.A.: Computer architecture: a quantitative approach. Morgarn Kaufmann, 5th edn. (2011)
4. Hennessy, J.L., Patterson, D.A.: A new golden age for computer architecture. Communications of the ACM **62**(2), 48–60 (2019)
5. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998). https://doi.org/10.1109/5.726791
6. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems 25. pp. 1097–1105 (2012)
7. Johnson, J.: Rethinking floating point for deep learning. ArXiv **abs/1811.01721** (2018)
8. IEEE standard for binary floating-point arithmetic. ANSI/IEEE Std 754-1985 pp. 1–20 (Oct 1985). https://doi.org/10.1109/IEEESTD.1985.82928
9. Del Barrio, A.A., Bagherzadeh, N., Hermida, R.: Ultra-low-power adder stage design for exascale floating point units. ACM Transactions on Embedded Computing Systems (TECS) **13**(3s), 1–24 (2014)
10. Kalamkar, D.D., Mudigere, D., Mellempudi, N., Das, D., Banerjee, K., Avancha, S., Vooturi, D.T., Jammalamadaka, N., Huang, J., Yuen, H., Yang, J., Park, J., Heinecke, A., Georganas, E., Srinivasan, S., Kundu, A., Smelyanskiy, M., Kaul, B., Dubey, P.: A study of bfloat16 for deep learning training. ArXiv **abs/1905.12322** (2019)
11. Kim, M.S., Del Barrio, A.A., Oliveira, L.T., Hermida, R., Bagherzadeh, N.: Efficient mitchell's approximate log multipliers for convolutional neural networks. IEEE Transactions on Computers **68**(5), 660–675 (2018)
12. Kim, H., Kim, M.S., Del Barrio, A.A., Bagherzadeh, N.: A cost-efficient iterative truncated logarithmic multiplication for convolutional neural networks. In: 2019 IEEE 26th Symposium on Computer Arithmetic (ARITH). pp. 108–111. IEEE (2019)
13. Gustafson, J.L., Yonemoto, I.T.: Beating floating point at its own game: Posit arithmetic. Supercomputing Frontiers and Innovations **4**(2), 71–86 (2017)
14. Posit arithmetic. https://posithub.org/docs/Posits4.pdf, [Online; último acceso 01-06-2020]
15. Planes de estudio de los grados en informática (UCM). https://informatica.ucm.es/grado, [Online; último acceso 03-06-2020]
16. Planes de estudio de los grados en informática (UPM). https://www.fi.upm.es/, [Online; último acceso 03-06-2020]
17. Planes de estudio de los grados en informática (UPC). https://www.upc.edu/es/grados/ingenieria-informatica-barcelona-fib, [Online; último acceso 03-06-2020]
18. Planes de estudio de los grados en informática (UGR). https://grados.ugr.es/informatica/, [Online; último acceso 03-06-2020]
19. Reinalda, B., Kulesza, E.: The Bologna Process–Harmonizing Europe's Higher Education: Including the Essential Original Texts. Verlag Barbara Budrich (2006)
20. Del Barrio, A.A., Manzano, J.P., Maroto, V.M., Villarín, Á., Pagán, J., Zapater, M., Ayala, J., Hermida, R.: HackRF+ GNU Radio: A software-defined radio to

teach communication theory. The International Journal of Electrical Engineering & Education pp. 1–18 (2019)
21. Del Barrio, A.A., Hermida, R., Ogrenci-Memik, S.: A combined arithmetic-high-level synthesis solution to deploy partial carry-save radix-8 booth multipliers in datapaths. IEEE Transactions on Circuits and Systems I: Regular Papers **66**(2), 742–755 (2019)
22. Del Barrio, A.A., Hermida, R., Memik, S.O.: A partial carry-save on-the-fly correction multispeculative multiplier. IEEE Transactions on Computers **65**(11), 3251–3264 (2016)
23. Carmichael, Z., Langroudi, H.F., Khazanov, C., Lillie, J., Gustafson, J.L., Kudithipudi, D.: Deep positron: A deep neural network using the posit number system. In: 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE). pp. 1421–1426. IEEE (2019)
24. Murillo, R., Barrio, A.A.D., Juan, G.B.: Template-based posit multiplication for training and inferring in neural networks. ArXiv **abs/1907.04091** (2019)
25. Langroudi, H.F., Carmichael, Z., Kudithipudi, D.: Deep learning training on the edge with low-precision posits. ArXiv **abs/1907.13216** (2019)
26. Murillo, R., Del Barrio, A.A., Botella, G.: Customized posit adders and multipliers using the flopoco core generator. In: 2020 IEEE International Symposium on Circuits and Systems (ISCAS) (2020)
27. Murillo, R., Del Barrio, A.A., Botella, G.: Deep pensieve: A deep learning framework based on the posit number system. Digital Signal Processing p. 102762 (2020)
28. New approach could sink floating-point computation. https://www.nextplatform.com/2019/07/08/new-approach-could-sink-floating-point-computation/, [Online; último acceso 03-06-2020]
29. Tiwari, S., Gala, N., Rebeiro, C., Kamakoti, V.: Peri: A posit enabled RISC-V core. ArXiv **abs/1908.01466** (2019)
30. Arunkumar, M., Bhairathi, S.G., Hayatnagarkar, H.G.: Perc: Posit enhanced rocket chip. In: Fourth Workshop on Computer Architecture Research with RISC-V at International Symposium on Computer Architecture (ISCA) (2020)
31. IEEE standard for floating-point arithmetic. IEEE Std 754-2008 pp. 1–70 (Aug 2008). https://doi.org/10.1109/IEEESTD.2008.4610935
32. Cococcioni, M., Rossi, F., Ruffaldi, E., Saponara, S.: Fast approximations of activation functions in deep neural networks when using posit arithmetic. Sensors **20**(5), 1515 (2020)
33. Uguen, Y., Forget, L., de Dinechin, F.: Evaluating the hardware cost of the posit number system. In: 2019 29th International Conference on Field Programmable Logic and Applications (FPL). pp. 106–113. IEEE (2019)
34. De Dinechin, F., Forget, L., Muller, J.M., Uguen, Y.: Posits: the good, the bad and the ugly. In: Proceedings of the Conference for Next Generation Arithmetic 2019. pp. 1–10 (2019)
35. Krizhevsky, A.: Learning multiple layers of features from tiny images. Master's thesis, University of Toronto, Canada (2009)