ISSN 1870-4069

Diseño empírico de una arquitectura de perceptrón multicapa binario residual

Agustín Solís Winkler, José Luis Tapia Fabela,
Santiago Osnaya Baltierra

Universidad Autónoma del Estado de México,
México

{asolisw, jtapiaf, sosnayab}@uaemex.mx

Resumen. Desde su introducción en 2015 por Courbariaux, las redes neuronales binarias han surgido como una alternativa prometedora para reducir los grandes requisitos de cómputo, memoria y almacenamiento en modelos de aprendizaje profundo, permitiendo su implementación en dispositivos con recursos limitados, como los utilizados en el cómputo de frontera. No obstante, estos modelos presentan una notable degradación en su precisión en comparación con sus equivalentes de punto flotante. En este estudio, se emplean las últimas recomendaciones de diseño de arquitecturas binarias para desarrollar empíricamente una arquitectura de perceptrón multicapa binario residual que reduce los efectos adversos del proceso de binarización. La intención es proporcionar un modelo sencillo de implementar que no requiera un nivel experto en diseño de redes neuronales para su uso en dispositivos de baja potencia.

Palabras clave: Red neuronal binaria, red neuronal, perceptrón multicapa, compresión de redes, cuantización, binarización.

Empirical Design of a Residual Binary Multilayer Perceptron Architecture

Abstract. Since its publication in 2015 by Courbariaux, binary neural networks have proven to be a good alternative for reducing computing, memory, and storage requirements in deep learning models, which allows them to be implemented on devices with limited resources, such as those used in edge computing. However, binary models present a notable degradation in accuracy compared to their floating-point counterparts. Considering the multilayer perceptron as a flexible model of machine learning, we use the design recommendations of binary architectures of the state of the art in this work to empirically develop a binary residual multilayer perceptron architecture, which reduces the degradation caused by the binarization process. The aim is to have a model that is simple to implement and that does not require an expert level in the design of neural networks on the part of its users to deploy models that work on hardware-constrained devices.

Keywords: Binary neural network, neural network, multilayer perceptron, model compression, quantization, binarization.

1. Introducción

Como han explicado [1], los métodos de aprendizaje profundo basados en el uso de redes neuronales prealimentadas de múltiples capas o perceptrones multicapa, los cuales son una extensión del modelo original del perceptrón propuesto por [2], han tenido gran éxito académico y comercial en muchas áreas de interés como el reconocimiento y procesamiento de imágenes, reconocimiento del habla, procesamiento del lenguaje natural, traducción automática, desempeño superior al humano en juegos y hasta conducción autónoma de automóviles, por lo que han sido fundamentales en el éxito y consolidación de la inteligencia artificial en nuestros días como hacen notar [3, 4].

Aunque los mejores modelos de aprendizaje profundo obtienen resultados espectaculares en el laboratorio, su tamaño los vuelve poco útiles en aplicaciones del mundo real [5]. Desde los trabajos de [6, 7] se ha planteado la necesidad de reducir los requerimientos de los modelos de aprendizaje profundo para permitir su uso en dispositivos de bajos recursos [8].

Dado que las aplicaciones basadas en el uso de redes profundas tienen gran auge en la actualidad, esta necesidad ha conducido al desarrollo del campo de la compresión de redes neuronales [9]. Dentro de este campo, la cuantización [10] y la binarización [11,12] presentan gran interés debido a la sencillez de sus conceptos e implementación.

La binarización consiste en reducir la precisión de los parámetros de la red neuronal hasta un solo bit, lo que representa la forma más extrema de la cuantización [12]. Esta técnica permite ahorrar espacio de almacenamiento y memoria, además de reducir los requerimientos de procesamiento y tiempo de ejecución al reemplazar las operaciones de punto flotante por operaciones lógicas y de conteo de bits como lo proponen [13] y confirman otros autores como [14, 15].

A pesar de su atractivo, las redes neuronales binarias presentan problemas de notable degradación y pérdida de exactitud en comparación con sus equivalentes de punto flotante según se ha expresado en estudios previos [15, 16].

Sin embargo, se han desarrollado técnicas y distintos enfoques para abordar estos problemas, los cuales ha sido resumidos por [12], entre los que podemos contar la reducción de errores de cálculo durante la cuantización, la mejora de la función de pérdida y la aproximación del gradiente; la aplicación de diferentes estrategias de entrenamiento y el diseño de arquitecturas específicas para redes binarias.

Este último enfoque resulta particularmente interesante ya que busca crear modelos más eficientes y compactos al mismo tiempo que trata de optimizar la red y mejorar su exactitud. Aunque aún no se cuenta con una explicación matemática formal del funcionamiento de las redes neuronales binarias como explican [17], se sabe que las redes neuronales son aproximadores universales de funciones continuas, según describen [18].

Adicionalmente, [19] han demostrado la aproximación universal de funciones mediante redes neuronales binarias de tres capas. Estos hallazgos sugieren que es posible diseñar un perceptrón multicapa binario capaz de aproximar la solución de problemas que puedan ser modelados. Por lo tanto, el presente trabajo propone desarrollar, de manera empírica, una arquitectura de perceptrón multicapa binario residual utilizando ampliación e inserción capas adicionales, así como el uso atajos.

El objetivo es determinar si la exactitud obtenida con estos cambios de arquitectura puede compensar la pérdida de información derivada de la binarización y si la precisión obtenida es comparable con la del perceptrón multicapa de punto flotante, con un menor costo computacional.

2. Trabajos relacionados

Se han desarrollado diversos trabajos para mejorar la precisión y exactitud de las redes neuronales binarias, y se han realizado importantes avances en la teoría desde su introducción en 2015. El modelo original de red neuronal binaria, BinaryConnect de [20] binariza los pesos, pero no las activaciones. Al año siguiente el mismo equipo presentó BinaryNet que ya utiliza activaciones binarias [11].

Posteriormente, el modelo XNOR-NET de [13], incluyó todos los métodos propuestos en la BinaryConnect original, pero agregando un valor de ganancia para compensar la pérdida de información durante la binarización y cambiando el orden de algunas capas para mejorar el entrenamiento. Sin embargo, aunque los términos de ganancia mejoran la precisión de la red, su cálculo resulta costoso.

En los últimos años, se han presentado varios enfoques para mejorar la precisión y eficiencia de las redes neuronales binarias. Uno de estos enfoques es el propuesto por [21], denominado Binarized Neural Networks (BNN), que utiliza una aproximación probabilística para la binarización de los pesos y activaciones, lo que ha demostrado ser más eficiente en términos de memoria y cómputo que los modelos anteriores.

Sin embargo, esta aproximación muestra una disminución en la precisión, lo que ha sido abordado mediante el uso de técnicas como la cuantización de pesos y activaciones y la adición de capas con precisión de punto flotante.

Otra propuesta de 2016 es la DoReFa-Net por [22], que trata de mejorar las ineficiencias de XNOR-NET, utilizando diferente precisión y longitud para los pesos, las activaciones y los cálculos inversos durante el entrenamiento. Sin embargo, su método es complejo y no ha demostrado una mejora significativa sobre las operaciones de bits. Junto con esto, en los últimos años se han presentado avances en el diseño de arquitecturas de redes binarias, como Binary DenseNet por [23], que utiliza conexiones de atajo para aumentar el flujo de información entre capas.

Además, aconsejan utilizar el uso de capas de reducción con precisión de punto flotante para preservar la información y mejorar la exactitud de las redes binarias. Otro enfoque de los mismos autores es la MeliusNet publicada por [24] y mencionada por [25] que utiliza un Bloque Denso para incrementar la capacidad de las características. Estos avances en el diseño de arquitecturas muestran que se están realizando esfuerzos para mejorar la precisión de las redes binarias y superar sus limitaciones iniciales.

3. Conceptos

Esta sección describe los principios de la binarización de redes neuronales, y algunos de los conceptos principales que se utilizan en la implementación del modelo de perceptrón propuesto.

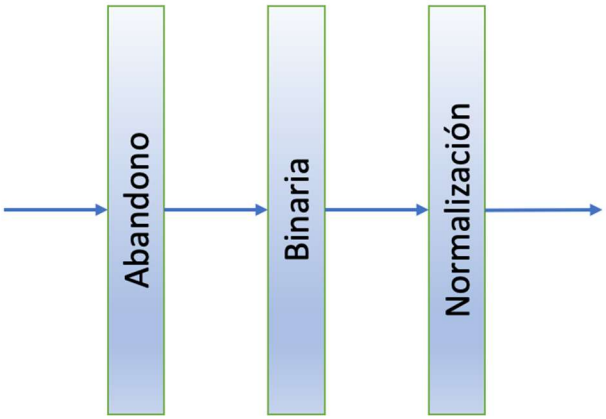

Fig. 1. Bloque de construcción de un perceptrón multicapa binario.

3.1. Red neuronal binaria

Una red neuronal binaria es un tipo de modelo de red neuronal propuesto por primera vez en 2015 por [20] en el cual solo las capas de entrada y salida se representan con valores de punto flotante y tanto los pesos como activaciones en las capas ocultas se representan con valores binarios.

La idea detrás de la binarización es restringir los valores de los pesos y las activaciones a +1 y -1 durante el entrenamiento, lo que permite reemplazar las operaciones de multiplicación y acumulación de punto flotante por operaciones XNOR y POPCOUNT de 1 bit que se ejecutan a mayor velocidad que las operaciones de 32 bits (Simons & Lee, 2019).

3.2. Implementación de la binarización

La función signo se utiliza tanto para binarizar las entradas, como activación, transformando en binarios los valores de punto flotante [11]:

$$signo(x) = \begin{cases} +1 \; si \; x \geq 0, \\ -1 \; si \; x < 0. \end{cases} \tag{1}$$

3.3. Retropropagación

Para entrenar una red utilizando los pesos binarizados, (Simons & Lee, 2019) aplican la propuesta de [11], para el uso de la técnica del estimador directo (STE) propuesta por [26] para poder entrenar una red binarizada con la función signo [12]:

$$\boldsymbol{b_w} = signo(\boldsymbol{w}). \tag{2}$$

El método del estimador directo aproxima el gradiente pasando por alto el gradiente de la capa en cuestión, y convirtiendo el gradiente problemático en una función de identidad [14]:

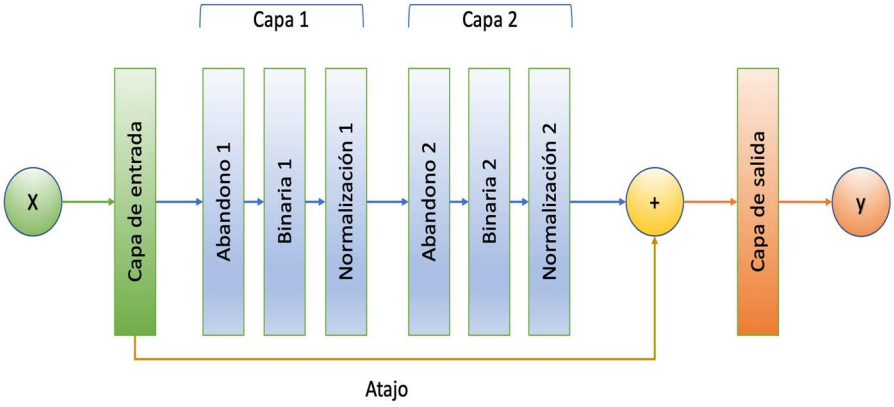

Fig. 2. Perceptrón multicapa binario residual de dos capas ocultas.

$$\frac{\partial L}{\partial \boldsymbol{w}} = \frac{\partial L}{\partial \boldsymbol{b_w}}. \tag{3}$$

Para las binarizar activaciones, se les aplica también la función signo y se utiliza el estimador directo en el recorrido hacia atrás, de la misma forma en que se binarizan los pesos. La función signo se utiliza también como función de activación en la red binaria. Si la entrada a la función de activaciones muy grande, se cancela el gradiente en el recorrido hacia atrás utilizando [14]:

$$\frac{\partial L}{\partial \boldsymbol{a}} = \frac{\partial L}{\partial \boldsymbol{b_a}} \times 1_{|a|\leq 1}, \tag{4}$$

donde $\boldsymbol{a}$ es la entrada de valor real de la función de activación y $\boldsymbol{b_a}$ es la salida binarizada de la función de activación.

3.4. Atajos

Un atajo es una conexión que lleva la información de la entrada de un bloque a la salida del mismo mediante una conexión de identidad [27]. Los atajos ayudan a aumentar el flujo de información entre dos bloques de la red, que se denominan residuales, evitando el problema de desvanecimiento del gradiente y sobreajuste en redes muy profundas. Los atajos son mencionados en el trabajo de [28].

3.5. Cuellos de botella

En una red neuronal se dice que en un diseño existe un cuello de botella, cuando el ancho de una capa es menor que el ancho de la capa anterior. Aunque la eliminación de los cuellos de botella es aconsejada por [29, 23], su efectividad es mayor en redes convolucionales que en perceptrones.

Tabla 1. Parámetros de los conjuntos de datos.

Conjunto	Objetos	Entrada	Clases	Instancias de Entrenamiento	Instancias de Prueba
MNIST	Imágenes 28x28 grises 8 bits	784	10	60000	10000
IMDB	Reseñas de películas	10000	2	25000	25000
Reuters	Noticias de 1986	5000	46	8982	2246
CIFAR10	Imágenes de 32x32 en color de 24 bits	3072	10	50000	10000

3.6. Capas de normalización por lotes

La normalización por lotes se aplica a los resultados de la capa anterior antes de la siguiente función de activación. Calcula la media y la varianza de un lote durante el entrenamiento, normalizando los datos con estos valores, ayudando a estabilizar el gradiente y reduciendo la dependencia del tamaño del lote durante el entrenamiento.

Cuando se aplica a las capas de las redes binarias, la normalización por lotes ayuda a mejorar la exactitud. [30] sugieren el abandono de las capas de normalización; desafortunadamente, el método propuesto en su lugar solo es aplicable para redes convolucionales. Sin embargo, reconocen que la normalización es una técnica bien conocida para estabilizar y acelerar el entrenamiento de los modelos.

Para el caso de redes binarias, nuestras pruebas iniciales confirmaron que el uso de capas de normalización por lotes ayuda al incremento de la exactitud. Una parte significativa de las pruebas consistió en estudiar el efecto de las capas de normalización en los modelos binarios.

3.7. Capas de abandono

Adicionalmente a la técnica de capas de normalización por lotes, la técnica de capas de abandono es muy utilizada para retardar el sobreajuste y mejorar la capacidad de generalización de los modelos. La idea es cambiar a cero algunas de las entradas durante el entrenamiento con una frecuencia definida. Las entradas que no son cero se incrementan en uno de forma que la suma de todas las entradas no cambia [31]. La ventaja de este método es que no agrega costo computacional.

4. Implementación y pruebas

El modelo multicapa binario que se propone, está basado en un bloque de construcción que reemplaza las capas totalmente conectadas en un perceptrón tradicional. Esta sección describe la arquitectura del modelo, la forma en que se

Tabla 2. Valores de los hiperparámetros.

Parámetro	**Valor**
Eras	10
Tamaño de lote	32
Ritmo de entrenamiento	0.001
Ritmo de abandono	0.05
Cuantización	8 bits
Tamaño de grupo (capas de agrupamiento)	8

implementan los atajos, y los resultados preliminares de las pruebas realizadas con los cuatro conjuntos de datos.

4.1. Bloque de construcción

Para el desarrollo del perceptrón multicapa binario residual, se consideraron las recomendaciones presentadas en especial por [23, 32], así como los componentes descritos en la sección anterior, en línea con la definición de red neuronal binaria.

Para construir el modelo, las capas de entrada y de salida se mantienen en punto flotante mientras que se utiliza un bloque de construcción que consiste en una secuencia ordenada de las siguientes capas: capa de abandono, capa totalmente conectada binaria y una capa de normalización por lotes como se muestra en la figura 1.

Este bloque se repite según el número de capas requeridas por el modelo. Si el perceptrón tiene más de una capa oculta se agrega una conexión de atajo, pero ajustando su anchura al de la última capa oculta.

4.2. Implementación de atajos

La función del atajo es pasar la información de la capa de entrada al final de la red, donde se combina con la salida de la última capa oculta para que contribuya al valor determinado en la capa de salida. Por lo general las conexiones de atajo se implementan utilizando capas convolucionales con filtros de 1x1, pero para el caso de este modelo se toma un enfoque diferente.

Se realizaron diferentes pruebas para la implementación de atajos siendo las más adecuadas las siguientes:

- Implementación mediante capa totalmente conectada sin función de activación.
- Implementación mediante capa cuantizada totalmente conectada sin función de activación. Para los experimentos, se utilizó cuantización de 8 bits, para reducir la memoria requerida por el atajo a la cuarta parte.
- Implementación de punto flotante utilizando una capa de agrupamiento máximo de una dimensión. Esta implementación requiere otra capa adicional totalmente conectada para ajustar con el tamaño de la siguiente capa cuando el ancho de la capa siguiente no se ajusta con exactitud a la salida de la capa de agrupamiento, lo

Tabla 3. Resultados con MNIST.

Modelo	Exactitud	Tamaño	Comentario
F-512	0.9811	1590.04	Punto flotante
F-128,88	0.9797	440.32	Punto flotante
F-64	0.9729	198.79	Punto flotante
B-128N, 88N, Q	0.9543	87.35	Binaria, normalizada, atajo
B-128N, 128N,128N	0.9538	25.79	Binaria normalizada
B-128N, 128N, Q	0.9502	120.79	Binaria, normalizada, atajo
B-128N, 128N	0.9490	22.29	Binaria, normalizada
B-128N, 88N	0.9488	19.63	Binaria, normalizada
B-128, 128N	0.9479	21.29	Binaria, normalizada
B-128, 88N	0.9478	16.63	Binaria, normalizada
B-128, 88	0.9361	17.95	Binaria equivalente
B-64	0.9093	9.91	Binaria equivalente
B-512	0.8969	71.04	Binaria equivalente

cual puede ocurrir cuando el cociente de la entrada y el tamaño de grupo no corresponden con el ancho de la capa destino.

4.3. Modelo propuesto

Utilizando el bloque de construcción y la conexión de atajo, la cual se suma con la última capa oculta, se obtiene un perceptrón binario multicapa residual se representa como se puede ver en la figura 2. Como puede observarse, la arquitectura propuesta reemplaza cada capa oculta de un perceptrón multicapa estándar por un bloque compuesto de una capa de abandono, una capa densa totalmente conectada binaria y una capa de normalización por lotes.

Para las pruebas realizadas, el ancho de la capa totalmente conectada binaria se mantiene igual al del perceptrón equivalente, para poder comprobar que las mejoras de precisión se deben a la adición de capas y no al incremento del ancho estas, aunque en aplicaciones reales se puede utilizar un factor de escala para mejorar la representatividad.

4.4. Pruebas

Conjuntos de datos e hiperparámetros

Para las pruebas se utilizaron cuatro conjuntos de datos: MNIST, IMDB, Reuters y CIFAR10 con la idea de mostrar la flexibilidad del perceptrón multicapa en tareas de clasificación de imágenes y textos, tanto binarias como categóricas. De cada uno de estos conjuntos se separan 10000 instancias de entrenamiento para el conjunto de validación, excepto del conjunto Reuters, del cual, por su tamaño, solo se separan 1000 ejemplos. La tabla 1 muestra las características de estos conjuntos.

Para enfocarnos en el cambio de la exactitud al modificar la arquitectura de la red, para todos los casos se utilizó un conjunto de valores similar para los hiperparámetros, en particular [17] sugiere utilizar tamaños de lote pequeños, por lo que en nuestro caso de utiliza 32. Los valores de los hiperparámetros se resumen en la Tabla 2.

Tabla 4. Resultados con IMDB.

Modelo	Exactitud	Tamaño	Comentario
F-250	0.8684	9767.58	Punto flotante
F-32,32	0.8569	1254.38	Punto flotante
F-64,32,16	0.8637	2510.5	Punto flotante
F 16,16	0.8510	626.19	Punto flotante
B-16, 16, Q	0.8596	176.07	Binaria, atajo
B-32, 32, Q	0.8443	352.19	Binaria normalizada, atajo
B-64,32N, 16, Q	0.8457	235.58	Binaria, normalizada, atajo
B-32,32N,32	0.8455	40.07	Binaria, normalizada
B-32, 32N, 32, Q	0.8583	352.69	Binaria, normalizada, atajo
B-250	0.8579	306.79	Binaria equivalente
B-32,32	0.8431	39.57	Binario equivalente
B-64,32,16	0.8442	78.94	Binaria equivalente
B-16.16	0.8450	19.75	Binaria equivalente

Identificación de modelos

La arquitectura del modelo se puede obtener de su descripción, para la que se utiliza se utiliza una notación de la forma (B -DxN, DxN, DxN, N, A) en donde:

B: designa a un modelo binario.

D: indica que se utiliza una capa de abandono antes de una capa totalmente conectada.

x: es un valor numérico que denota el ancho de la capa.

N: indica que se utiliza una capa de normalización en esa posición.

A: esta posición indica el uso de una conexión de atajo. Los valores posibles pueden ser:

F: Denota un atajo de puto flotante implementado mediante una capa totalmente conectada.

P: Denota un atajo de punto flotante implementado mediante una capa de agrupamiento.

Q: Denotan atajos implementados mediante una capa totalmente conectada cuantizadas a 8 bits.

4.5. Resultados preliminares con MNIST.

Para MNIST, se utilizaron como base modelos de punto flotante de una a tres capas ocultas. Los modelos binarios incluyeron de la misma manera de una a tres capas ocultas, más normalización. Para este conjunto de datos aún no se hacen pruebas exhaustivas de todas las combinaciones, y tampoco se utilizaron capas de abandono que se comenzaron a utilizar en un punto más tardío.

Están pendientes por realizar pruebas exhaustivas de todas las combinaciones por lo que podrían encontrase cambios en las arquitecturas de mejor desempeño. Sin embargo, podemos observar en la tabla 3, que los modelos de punto flotante obtienen la mayor

Tabla 5. Resultados con el conjunto Reuters.

Modelo	Exactitud	Tamaño	Comentario
F-64, 64	0.7845	3778.18	Punto Flotante
B-64N, 64N	0.7850	365.38	Binaria normalizada
B-64N, 64N, QN	0.7796	381.49	Binaria, normalizada, atajo
B-128N, 64N, 64N, Q	0.7787	407.05	Binaria normalizada, atajo
B-64N, 64N, 64N, Q	0.7685	N/D	Binaria normalizada, atajo
B-64N, 64N, P	0.7631	52.74	Binaria normalizada, atajo de agrupamiento
B-64, 64	0.7311	128.87	Binaria equivalente

exactitud entre todos los modelos, y los tres modelos binarios equivalentes la menor exactitud.

De los modelos modificados, el modelo de dos capas con atajo cuantizado (B-128N, 88N, Q) es el que tiene el mayor desempeño seguido del modelo de tres capas con normalización (B-128N, 128N, 128N), y en tercer lugar el modelo (B-128N, 128N, Q).

Estos resultados sugieren que, para la tarea de clasificación de imágenes, los atajos cuantizados representan una mejora que, aunque más costosa que una red sin atajos, da mejores resultados. También podemos notar que tres capas con normalización y sin cuellos de botella obtienen buenos resultados.

4.6. Resultados preliminares con IMDB

El desempeño con el conjunto de datos de IMDB resulta muy diferente al de MNIST. En la tabla 4 podemos notar un comportamiento bastante diferente, en donde dos de los perceptrones binarios equivalentes logran un buen desempeño, seguidos de un perceptrón de dos capas con atajo cuantizado y otro de 3 capas con atajo cuantizado también.

4.7. Resultados preliminares con el conjunto Reuters

Para el conjunto de datos de Reuters, se prueban modelos de dos y tres capas, cuyos resultados se muestran en la tabla 5.

4.8. Resultados preliminares con el conjunto CIFAR10

El conjunto de datos CIFAR10, supera las posibilidades de clasificación de los perceptrones, con los cuales se puede logran una exactitud máxima de 0.59 siempre y cuando se haga un preprocesamiento más extenso de la entrada, incluyendo aumento de las características y recorte de la imagen [33]. Sin embargo, en la tabla 6 se presentan las pruebas realizadas sin preprocesamiento para mostrar las bondades del modelo propuesto.

Se puede observar que los modelos B256N, 256N, P y el B-256N, 256N, 256N, P logran una exactitud de 0.4507 y 0.4580 respectivamente. Se puede notar además que los modelos sin modificaciones obtienen los promedios más bajos.

Tabla 5. Resultados con CIFAR10.

Modelo	Exactitud	Tamaño	Comentario
F-256	0.4224	3083.04	Punto flotante
F-256, 256	0.4576	3340.04	Punto flotante
F-256, 256, 256	0.4350	3597.04	Punto flotante 3 capas
B-256N, 256N, P	0.4507	464.04	Binaria, normalizada
B-256N, 256N, QN	0.4301	891.04	Binaria, normalizada, atajo
B-256N, P	0.4061	451.04	Binaria normalizada, atajo de agrupamiento
B-256N, QN	0.3755	880.04	Binaria, normalizada, atajo
B- 256N, 256N, 256N, PN	0.4580	N/D	Binaria normalizada, atajo de agrupamiento
B- 256N, 256N, 256N, Q	0.4201	N/D	Binaria normalizada, atajo
B-256	0.2612	107.04	Binaria equivalente
B-256, 256	0.1000	116.04	Binaria equivalente
B-256, 256, 256	0.1000	125.04	Binaria equivalente

5. Conclusiones y trabajo futuro

En conclusión, se realizaron pruebas sobre variantes del modelo de perceptrón multicapa binario residual con cuatro diferentes conjuntos de datos.

De los experimentos realizados se puede notar que efectivamente los perceptrones multicapa binarios equivalentes en general tienen menor desempeño que los de punto flotante.

También podemos observar que al agregar capas de normalización la exactitud del perceptrón mejora, y si adicionalmente se utilizan atajos siempre mejora la exactitud del perceptrón, lo cual confirma que estos mejoran el flujo de información. Un atajo cuantizado supera a una capa adicional en el desempeño, aunque podemos notar que es más costoso en el tamaño del modelo.

Aunque falta realizar más experimentos, podemos adelantar que el modelo de perceptrón multicapa binario residual cumple su función de ser una alternativa sencilla de implementar y siempre proporciona mejores resultados que el perceptrón binario simple equivalente al de punto flotante.

Hasta donde tenemos información, este tipo de atajos no se utilizado en modelos de perceptrón multicapa, son más comunes en modelos convolucionales, en los que se utilizan para regenerar la información de entrada al concluir un bloque de convolución.

Una vez que se terminen de realizar todas las series de experimentos, podremos contar con información suficiente para intentar un análisis de correlación y realizar un análisis de sensibilidad sobre los datos recolectados, que se espera arrojen más información sobre la exactitud del modelo.

A futuro, se continuará trabajando en redes neuronales binarias, pero ampliando al campo a redes convolucionales. Aunque el perceptrón multicapa es un modelo flexible, existen aplicaciones que se benefician más con modelos más sofisticados.

Referencias

1. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. The MIT Press (2016)
2. Rosenblatt, F.: The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review, vol. 65, no. 6, pp. 386–408 (1958) doi: 10.1037/h0042519
3. Chollet, F.: Deep learning with python. Manning Publications Co (2017)
4. Aggarwal, C. C.: Artificial intelligence: A textbook. Springer Nature Switzerland AG (2021) doi: 10.1007/978-3-030-72357-6
5. Pokhrel, S.: 4 Popular model compression techniques explained. Xailient (2022) xai lient.com/blog/4-popular-model-compression-techniques-explained/
6. LeCun, Y., Denker, J., Solla, S.: Optimal brain damage. Advances in Neural Information Processing Systems, vol. 2, pp. 598–605 (1989)
7. Han, S., Pool, J., Tran, J., Dally, W. J.: Learning both weights and connections for efficient neural networks (2015) doi: 10.48550/ARXIV.1506.02626
8. Neill, J. O.: An overview of neural network compression (2020) doi: 10.48550/ ARXIV.2006.0366
9. Cheng, Y., Wang, D., Zhou, P., Zhang, T.: Model compression and acceleration for deep neural networks: the principles, progress, and challenges. IEEE Signal Processing Magazine, vol. 35, no. 1, pp. 126–136 (2018) doi: 10.1109/msp.2017.2765695
10. Novac, P., Hacene, G. B., Pegatoquet, A., Miramond, B., Gripon, V.: Quantization and deployment of deep neural networks on microcontrollers. Sensors, vol. 21, no. 9, pp. 2984 (2021) doi: 10.3390/s21092984
11. Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R., Bengio, Y.: Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or –1 (2016) doi: 10.48550/ARXIV.1602.02830
12. Yuan, C., Agaian, S. S.: A comprehensive review of binary neural network. Artificial Intelligence Review (2023) doi: 10.1007/s10462-023-10464-w
13. Rastegari, M., Ordonez, V., Redmon, J., Farhadi, A.: XNOR-Net: ImageNet classification using binary convolutional neural networks (2016) doi: 10.48550/ARXIV.1603.05279
14. Simons, T., Lee, D.: A review of binarized neural networks. Electronics, vol. 8, no. 6, pp. 661 (2019) doi: 10.3390/electronics8060661
15. Qin, H., Gong, R., Liu, X., Bai, X., Song, J., Sebe, N.: Binary neural networks: A survey. Pattern Recognition, vol. 105, pp. 107281 (2020) doi: 10.1016/j.patcog.2020.107281
16. Bethge, J., Yang, H., Bornstein, M., Meinel, C.: Back to simplicity: how to train accurate BNNs from scratch? (2019) doi: 10.48550/ARXIV.1906.08637
17. Alizadeh, M., Fernández-Marqués, J., Lane, N. D., Gal, Y.: An empirical study of binary neural networks' optimisation. In: International Conference on Learning Representations, Conference Blind Submission (2018)
18. Hagan, M. T., Demuth, H. B., Beale, M. H., de-Jesús, O.: Neural network design. Martin Hagan, 2nd Edition (2014)
19. Redfern, A. J., Zhu, L., Newquist, M. K.: BCNN: A binary CNN with all matrix ops quantized to 1-bit precision. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (2021) doi: 10.1109/cvprw53098.2021.00518
20. Courbariaux, M., Bengio, Y., David, J.: BinaryConnect: Training deep neural networks with binary weights during propagations. In: Proceedings of the 28th International Conference on Neural Information Processing Systems, vol. 2, pp. 3123–3131 (2015) doi: 10.48550/ ARXIV.1511.00363
21. Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., Bengio, Y.: Quantized neural networks: Training neural networks with low precision weights and activations. Journal of Machine Learning Research, vol. 18, no. 1, pp. 6869–6898 (2017)

22. Zhou, S., Wu, Y., Ni, Z., Zhou, X., Wen, H., Zou, Y.: DoReFa-Net: Training low bitwidth convolutional neural networks with low bitwidth gradients (2016) doi: 10.48550/ARXIV.1606.06160
23. Bethge, J., Yang, H., Bornstein, M., Meinel, C.: BinaryDenseNet: Developing an architecture for binary neural networks. In: IEEE/CVF International Conference on Computer Vision Workshop, pp. 1951–1960 (2019) doi: 10.1109/iccvw.2019.00244
24. Bethge, J., Bartz, C., Yang, H., Chen, Y., Meinel, C.: MeliusNet: Can binary neural networks achieve MobileNet-level accuracy? (2020) doi: 10.48550/ARXIV.2001.05936
25. Yuan, C., Agaian, S. S.: A comprehensive review of binary neural network. Artificial Intelligence Review (2023) doi: 10.1007/s10462-023-10464-w
26. Hinton, G., Teleman, T.: Divide the gradient by a running average of its recent magnitude (2012) www.aminer.org/pub/5b076eb4da5629516ce741dc/lecture-rmsprop-divide-the-gradient-by-a-running-average-of-its-recent
27. Liu, Z., Wu, B., Luo, W., Yang, X., Liu, W., Cheng, K.: Bi-Real Net: Enhancing the performance of 1-bit CNNs with improved representational capability and advanced training algorithm. In: Computer Vision - European Conference on Computer Vision, pp. 747–763 (2018) doi: 10.1007/978-3-030-01267-0_44
28. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
29. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–9 (2015) doi: 10.1109/cvpr.2015.7298594
30. Chen, T., Zhang, Z., Ouyang, X., Liu, Z., Shen, Z., Wang, Z.: "BBB - BN = ?": Training binary neural networks without batch normalization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4619–4629 (2021)
31. Google: Keras: The python deep learning API (2023) keras.io
32. Li, H., De, S., Xu, Z., Studer, C., Samet, H., Goldstein, T.: Training quantized nets: a deeper understanding. In: 31st Conference on Neural Information Processing Systems (2017) doi: 10.48550/ARXIV.1706.02379
33. Parvathi, C.: Classification of cifar-10 dataset using a multi-layer perceptron model and a convolutional neural network model (2023) github.com/ parvathic/ml-cifar10