UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MÉXICO

UNIDAD ACADÉMICA PROFESIONAL TIANGUISTENCO

DESARROLLO EMPÍRICO DE UNA ARQUITECTURA DE PERCEPTRÓN MULTICAPA BINARIO RESIDUAL.

TESIS

QUE PARA OBTENER EL GRADO DE

MAESTRO EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA:

ING. AGUSTÍN SOLÍS WINKLER

TUTOR ACADÉMICO:

DR. SANTIAGO OSNAYA BALTIERRA

TUTORES ADJUNTOS:

DR. ASDRÚBAL LÓPEZ CHAU

DR. JOSÉ LUIS TAPIA FABELA

TIANGUISTENCO MÉXICO, JULIO 2023

CONTENIDO

LISTA DE ILUSTRACIONES, GRÁFICAS Y TABLAS

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 inherentes a los modelos de aprendizaje profundo, permitiendo su implementación en dispositivos con recursos limitados, como los utilizados en el cómputo de frontera. No obstante, estas estructuras presentan una notable degradación en su exactitud en comparación con sus equivalentes de punto flotante.

Se han ideado técnicas con variados enfoques para abordar este problema, entre las que se encuentran la reducción del error de cuantización, la mejora de las funciones de pérdida y de aproximación del gradiente, diversas estrategias de entrenamiento y el diseño de arquitecturas específicas para redes binarias.

Este trabajo de tesis propone utilizar las últimas recomendaciones del estado del arte en el diseño de redes binarias para desarrollar empíricamente una arquitectura de perceptrón multicapa binario residual que reduce los efectos adversos del proceso de binarización.

La intención es presentar un modelo que siendo compacto y eficiente, obtiene exactitud comparable a la de un perceptrón de punto flotante.

1. INTRODUCCIÓN

1.1. DEL APRENDIZAJE AUTOMÁTICO A LAS REDES NEURONALES BINARIAS

El aprendizaje automático es la rama de la inteligencia artificial que trata de los métodos utilizados para atacar problemas caracterizados por no disponer de algoritmos procedimentales de solución conocidos, como lo enuncian Shalev-Shwartz & Ben-David [1] y Kasabov [2]; partiendo de ejemplos, los algoritmos de aprendizaje construyen modelos que generalizan una solución a la dificultad planteada según explican Aggarwal [3] y Munkata [4], de manera análoga a como los humanos aprendemos de forma inductiva.

De acuerdo con Goodfellow [5], entre los métodos más utilizados en el aprendizaje automático, se encuentran los de aprendizaje profundo, los cuales están basados en el uso de redes neuronales prealimentadas de múltiples capas, también conocidas como redes profundas o perceptrones multicapa, nombrados así por extensión del modelo original de Rosenblatt [6].

Estas redes han obtenido 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 sobrehumano en distintos juegos y hasta conducción autónoma de automóviles. Estos resultados, han sido fundamentales para el éxito y consolidación de la inteligencia artificial en la época actual, como describen Chollet [7] y Aggarwal [3].

Para Alqahtani [8] y Cheng [9], la gran utilidad y resultados espectaculares obtenidos por las redes profundas como las utilizadas para resolver el reto de clasificación ImageNet, cuyo ganador en 2015 de acuerdo con los datos de Pokhrel ya utilizaba 152 capas [10], vienen acompañados de grandes requerimientos para su entrenamiento y operación, especialmente en intensidad computacional, capacidad de memoria y almacenamiento. Estas necesidades se traducen en un gran consumo energía como expone Mishra [11], y claramente ejemplifica Hao [12], quien describe que para entrenar el transformador GPT2 de OpenAI, se requirió un consumo de 656,347 kW, con un costo superior a tres millones de dólares. Ganesh [13] y O'Neill [14] concuerdan con Hao [12] en que entrenar un modelo del estado del arte puede generar una huella de carbono mayor que la de cinco automóviles de gasolina durante todo su tiempo de vida.

Pokhrel sostiene [10] que muchas aplicaciones de la vida diaria requieren la capacidad de ser procesadas en tiempo real en el dispositivo que genera la información y en muchas ocasiones con requerimientos estrictos de latencia y privacidad de acuerdo con Cheng [9]. Muchos de estos son dispositivos incrustados, que por regla general tienen capacidad limitada de hardware [15]; de manera similar, los dispositivos de frontera como sensores, cámaras inteligentes,

microcontroladores; los destinados al internet de las cosas [16] y otros artefactos populares como los monitores de estado físico, caen dentro de esta categoría.

Aunque los mejores modelos de aprendizaje profundo obtienen resultados fuera de lo común en el laboratorio, su tamaño los vuelve inútiles en aplicaciones del mundo real [10]. Desde los trabajos de LeCun [17] y Han [18], se expone la necesidad de reducir los requerimientos de estos sistemas de aprendizaje para permitir su uso en dispositivos de bajos recursos como explica O´Neill [14]. Considerando 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 compresión de redes neuronales cuyos avances resume el trabajo de Cheng [9].

La cuantización, según explica Novac [19] y la binarización propuesta por Courbariaux [20], que son los subcampos de interés del presente trabajo, se derivan del área de compresión.

La compresión de redes neuronales puede definirse como el conjunto de técnicas cuyo objetivo es disminuir la complejidad de los modelos manteniendo al mismo tiempo su precisión y desempeño en la predicción como exponen Abbasi-Asi & Yu [21] y Marinó [22]. La compresión implica la reducción del conjunto de parámetros de entrenamiento, lo cual decrece los requerimientos de proceso y espacio de la red profunda. Si la disminución es significativa, Aggarwal [23] y Alqahtani [8] concuerdan que esta red comprimida puede emplearse en plataformas con potencia de cómputo y memoria limitadas.

Entre las técnicas de compresión, la cuantización es una de las categorías de mayor interés, debido a que de acuerdo con O'Neill [14], al utilizar representaciones de baja precisión numérica se disminuyen las necesidades de cálculo y almacenamiento logrando soluciones rápidas al problema de comprimir un modelo neuronal como destacan Cheng [9] y Qin [24].

En palabras de Yuan & Agaian [25], la binarización es la forma más extrema de la cuantización. Es el proceso de reducir la precisión de los parámetros de la red hasta un solo bit. El propósito no es solo ahorrar espacio de almacenamiento y memoria, sino también 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 queda consignado por Yuan & Agaian [25], Simons [26] y Qin [24].

El problema de las redes neuronales binarias es que sufren de notable degradación y pérdida de exactitud como concuerdan Quin [24], Yuan & Agaian [25] junto con Bethge [27] al comparar los resultados de estas con sus equivalentes de punto flotante. Estos problemas se han tratado de minimizar utilizando diferentes técnicas que atacan distintos aspectos de los modelos, tales como los cálculos requeridos durante el entrenamiento, las estrategias de entrenamiento y el desarrollo de arquitecturas específicas para redes binarias, siendo esta última una de las más

interesantes, porque se enfoca en crear modelos más eficientes y compactos al mismo tiempo que trata de optimizar la red y mejorar su exactitud.

1.2. PLANTEAMIENTO DEL PROBLEMA

1.2.1. ENUNCIADO

Desde la introducción de las redes neuronales binarias por Courbariaux [28], se han hecho esfuerzos en todos los sentidos para reducir la degradación y mejorar su exactitud. Con distintos métodos como los factores de escala del modelo XNOR-Net de Rastegari [29], las mejoras a la función de cuantización empleadas por Zhou [30] en Do-Re-Fa-Net, la ABC-Net de Lin [31] que utiliza capas cuantizadas de diferente precisión, o el uso de la función sigmoide en lugar del estimador directo en la Bi-Real-Net de Liu [32] que no emplea estimaciones del gradiente, estos autores han dedicado a explorar métodos de mejorar la precisión mediante estrategias diversas de mayor o menor complejidad.

Con el diseño enfocado a redes binarias propuesto por Bethge [33] con BinaryDenseNet, o la Binarized Mobile Net de Phan [34], la ReactNet de Liu [35] y la BCNN de Redfern [36], se procura mejorar la exactitud mediante modificaciones de la arquitectura.

La mayoría de los trabajos mencionados investigan el uso de redes neuronales binarias para reducir el costo computacional y mejorar la eficiencia en aplicaciones de visión por computadora. Es importante destacar que muchos de los trabajos referidos y del estado del arte se enfocan en el uso de capas convolucionales binarias. Se ha observado que hay una falta de investigación sobre el uso de perceptrones multicapa binarios en problemas genéricos, lo que indica una brecha en la literatura actual en cuanto a la aplicación de estas redes neuronales en diferentes áreas, lo cual conduce a la interrogante que se presenta a continuación.

1.2.2. PREGUNTA DE INVESTIGACIÓN

¿Es posible desarrollar una arquitectura modificada para un perceptrón multicapa binario que tenga una exactitud comparable a la de un perceptrón multicapa de punto flotante mediante la exploración de diferentes configuraciones de capas utilizando un enfoque empírico?

1.2.3. HIPÓTESIS

Si se modifica la arquitectura de un perceptrón multicapa binario mediante la inclusión de capas de abandono y normalización, más la adición de conexiones residuales, de manera que se aumente la expresividad de la red, entonces su exactitud será comparable a la de un perceptrón multicapa de punto flotante con un menor requerimiento de memoria.

1.3. Objetivos

1.3.1. Objetivo general

Tomando en consideración la propiedad exhibida por las redes neuronales de ser aproximadores universales de funciones continuas tal como sostiene Hagan [37] y con apoyo de la demostración propuesta por Redfern [36], la cual establece que una red neuronal binaria de 3 capas es a su vez un aproximador universal de funciones, esto sugiere que se pueden diseñar perceptrones multicapa binarios que aproximan la solución de problemas modelables, estableciendo un número adecuado capas ocultas y neuronas por capa.

Por lo tanto, el objetivo de este trabajo es el siguiente: desarrollar una arquitectura de perceptrón multicapa binario que incorpore capas adicionales y atajos, para conocer si la exactitud obtenida con estos cambios de estructura puede compensar la pérdida de información derivada de la binarización, buscando evaluar si esta precisión es comparable con la de un perceptrón multicapa de punto flotante, pero a un menor costo computacional. El enfoque adoptado en esta investigación es de naturaleza empírica.

1.3.2. Objetivos específicos

Para realizar el desarrollo de esta arquitectura, se consideran los siguientes objetivos específicos:

- Aplicar diferentes modificaciones a la arquitectura de un perceptrón multicapa binario para conocer el comportamiento de precisión y pérdida obtenidos con cada uno de estos cambios.
- Identificar una manera de implementar atajos que mantenga compacto el tamaño del modelo.
- Proponer una arquitectura de perceptrón multicapa binario que pueda ser implementada de manera sencilla por cualquier persona interesada sin requerir conocimientos profundos de redes neuronales binarias.
- Elaborar una lista de recomendaciones de diseño para perceptrones multicapa binarios, basada en los resultados obtenidos durante la experimentación.

1.4. Justificación

Un modelo de perceptrón multicapa binario residual se justifica debido a su capacidad para satisfacer los requisitos de aprendizaje profundo y su idoneidad para su implementación en dispositivos con limitaciones de energía y recursos computacionales, como los dispositivos de cómputo de frontera. Esto se logra mediante la sustitución de cálculos de punto flotante por operaciones elementales de bits y manteniendo en memoria únicamente los pesos y

activaciones binarizados. Estas características permiten un procesamiento eficiente y optimizado en entornos con restricciones de recursos, sin comprometer la capacidad de realizar tareas complejas de aprendizaje profundo.

1.5. Delimitación

Este estudio se centra en el diseño y prueba de perceptrones multicapa binarios de hasta tres capas. Esta profundidad se considera adecuada y suficiente para cualquier tarea de clasificación tanto de textos como de imágenes de acuerdo con De Luca [38] , Heaton [39] y Redfern [36]. Es importante destacar que este trabajo de investigación no aborda el problema de regresión utilizando redes neuronales binarias ya que estas no son la elección típica para dicha tarea, ni se enfoca en el diseño de redes neuronales convolucionales binarias.

Las variables que se utilizan para evaluar las arquitecturas propuestas con la exactitud y el tamaño del modelo. La rapidez final de inferencia no está considerada, debido a que esta depende del ambiente de ejecución.

1.6. Organización del documento

El presente trabajo consta de seis capítulos. El capítulo 1 hace una introducción a las redes neuronales, la compresión de estas y presenta la binarización como el máximo grado de la cuantización. Se mencionan algunos de los avances más significativos en la optimización de redes binarias proporcionando de esta manera un panorama general del tema. También se realiza el planteamiento del problema, enunciando la hipótesis y los objetivos de este estudio. El capítulo 2 se dedica al marco teórico de las redes neuronales binarias, describiendo en mayor detalle los conceptos de redes neuronales, compresión y binarización, así como las herramientas a disposición del investigador que se utilizaron para trabajar con estos modelos. El capítulo 3 resume el estado del arte en la investigación sobre redes neuronales binarias, haciendo énfasis en los trabajos sobre arquitecturas binarias. El capítulo 4 presenta la metodología que se siguió para optimizar perceptrones multicapa binarios en tareas de clasificación binaria y multiclase. El capítulo 5 se enfoca en el proceso de experimentación y los resultados obtenidos con diferentes conjuntos de datos. En el capítulo 6 se elaboran las conclusiones, enumeran aportaciones y posible dirección para trabajos futuros.

2. MARCO TEÓRICO

2.1. APRENDIZAJE AUTOMÁTICO

Los textos de Aggarwal [3] y Munkata [4], describen que el enfoque de los seres humanos al aprendizaje está basado por lo general en la experiencia probatoria, a partir de ejemplos del mundo real. Este enfoque tiene un elemento de generalización que no puede ser justificado en su totalidad por el uso de métodos basados en la lógica. En muchos campos, el aprendizaje a través de la práctica es más eficiente que tratar de aprender las reglas de un sistema formalmente definido. Los humanos tenemos la habilidad natural de crear hipótesis generalizadas a partir de ejemplos, que luego utilizamos para resolver problemas nuevos en situaciones similares.

En opinión de Aggarwal [3], esta habilidad humana de aprender de manera inductiva se refleja en los métodos diseñados para el aprendizaje automático en los cuales, se utilizan observaciones etiquetadas en lugar de alguna hipótesis. Es una forma de aprendizaje dirigido por evidencias o aprendizaje estadístico, que es más poderoso que el aprendizaje deductivo, en especial cuando se dispone de una gran cantidad de ejemplos que puedan utilizarse para el entrenamiento.

2.2. REDES NEURONALES ARTIFICIALES

2.2.1. INSPIRACIÓN BIOLÓGICA

Tanto Shalev-Shwartz & Ben-David [1] como Munkata [4] definen una red neuronal artificial como un modelo matemático-computacional abstracto de aprendizaje automático inspirado en la estructura y funcionamiento del cerebro. Este modelo se compone de un conjunto de unidades de procesamiento o neuronas artificiales que, en palabras de Kasabov [2], están conectadas entre sí de manera similar a las neuronas biológicas, que son células especializadas que responden a la entrada de impulsos eléctricos recibidos por las dendritas y comunican esta respuesta a través de una salida o axón, mediante conexiones llamadas sinapsis como describen Coppin [40], Aggarwal [23] y Wikipedia [41], siguiendo la hipótesis de que la actividad mental es producto de la actividad electroquímica que ocurre en las redes de neuronas que componen el cerebro humano según afirman Russell & Norvig [42]. La ilustración 1 muestra las partes de la neurona y sus interconexiones con las que se conforman estas redes.

Concordando con Hagan [37], las redes neuronales artificiales solo están remotamente relacionadas con sus contrapartes biológicas; para Kasabov [2], el objetivo de estas no es modelar el cerebro sino servir de pauta para construir representaciones que ayuden a resolver problemas para los que no existen algoritmos específicos. En opinión de Russell & Norvig [42],

el aprendizaje profundo actual toma su inspiración de muchos campos, en particular de las matemáticas aplicadas como el álgebra lineal, la probabilidad, la teoría de la información y la optimización numérica. Siguiendo la misma idea, Goodfellow [5] enuncia que el modelado del cerebro le corresponde a la neurociencia computacional, la cual es un campo de estudio separado del aprendizaje automático y los modelos que lo conforman.

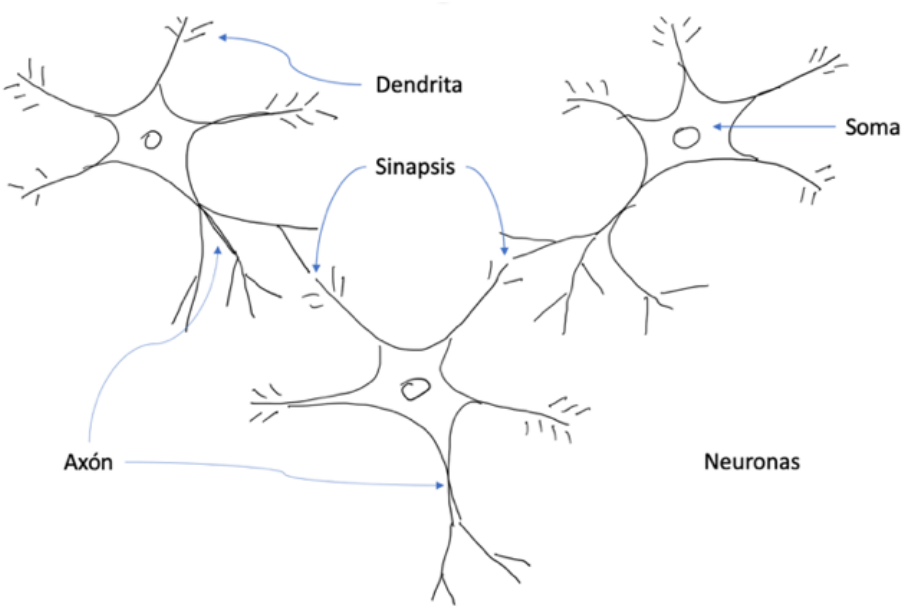


Ilustración 1. Neurona biológica. (Por ASW)

2.2.2. MODELADO DE REDES NEURONALES ARTIFICIALES

Ampliando la definición de gráfica dirigida acíclica de González Moreno [43] con la encontrada en Wikipedia [44], y construyendo sobre la base del concepto de gráfica computacional, tanto Shalev-Shwartz & Ben-David [1] como Aggarwal [23] establecen que una red neuronal artificial puede verse como un tipo especial de gráfica computacional, cuyos vértices representan las unidades de procesamiento que comúnmente llamamos neuronas artificiales, que se conectan entre sí por aristas que representan las conexiones y que unen la salida de una neurona con la entrada de otra.

Hagan [37] considera la neurona artificial como el bloque básico de construcción de la red neuronal; de acuerdo con Shalev-Shwartz & Ben-David [1] cada neurona artificial se modela como una función escalar no lineal, conocida como función de activación la cual, dependiendo del tipo de problema, se puede seleccionar específicamente, contando entre las más utilizadas la función sigmoide, la unidad de rectificación lineal (ReLU), la identidad o la función signo.

De esta manera, siguiendo las exposiciones de Aggarwal [23] y Bettilyon [45] se sabe que cada nodo realiza el cómputo sobre su entrada, la cual se obtiene de todas las neuronas conectadas a este para crear su valor de salida, que es alimentado a los nodos subsecuentes de manera semejante a como ocurre con las neuronas biológicas. Como resultado, la gráfica computacional construida conectando estos bloques básicos aproxima una función de gran complejidad mediante la composición recursiva de funciones como lo muestran Goodfellow [5] y Aggarwal [3].

2.2.3. PERCEPTRONES MULTICAPA.

Munkata [4], Heaton [46] junto con Shalev-Shwartz & Ben-David [1] describen como las unidades de procesamiento de una red neuronal artificial se organizan en capas, las cuales están conectadas en sucesión como explica Chollet [7], de forma que los nodos de una capa solo pueden recibir entradas de los nodos de la capa anterior, y las salidas solo pueden transferirse a los nodos de la capa siguiente.

Como en esta configuración no existen ciclos, Shalev-Shwartz & Ben-David [1] y Aggarwal [3] denominan a este modelo red neuronal prealimentada; adicionalmente Goodfellow [5] lo nombra perceptrón multicapa, por extensión del concepto del trabajo original de Rosenblatt [6] y la red neuronal prealimentada de una capa citada por Russell & Norvig [42].

Con respecto a las capas, Shalev-Shwartz & Ben-David [1] explican que las unidades de procesamiento que conforman la red pueden organizarse en conjuntos disjuntos de forma tal que si V es un perceptrón multicapa entonces:

$$V = \cup_{t=0}^{T} V_t \qquad (1)$$

La capa V_0, es llamada capa de entrada, y está compuesta por n neuronas, donde n se conoce como dimensionalidad del espacio de entrada. Esta capa, como explican Munkata [4] y Aggarwal [3] no realiza ningún procesamiento.

Las capas $V_1, \dots, V_{T-1}$ son las capas ocultas, llamadas así porque como aclaran Heaton [46], Munkata [4] y Goodfellow [5], su operación está oculta para el usuario.

La capa final V_T se denomina capa de salida, ya que como concuerdan Munkata [4], Heaton [46] y Shalev-Shwartz & Ben-David [1], es la que entrega el resultado final calculado de toda la red.

Para cada capa el número de neuronas que contiene se expresa como $|V_t|$ y se conoce como ancho de capa.

Cada neurona de la primera capa oculta se conecta con todas las entradas, y cada neurona de las capas subsecuentes se conecta con la salida de cada neurona de la capa anterior.

Cuando una red neuronal tiene al menos una capa oculta, se le conoce como red profunda. Como explican Shalev-Shwartz & Ben-David [1], la profundidad de la red es igual al número de capas de procesamiento de la red, es decir la capa de salida más el número total de capas ocultas. La ilustración 2 muestra un esquema de perceptrón multicapa con dos capas ocultas.

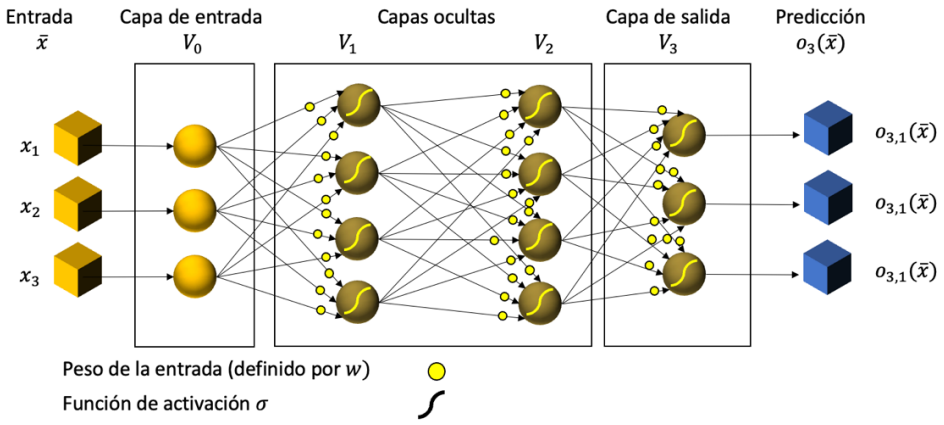


Ilustración 2. Perceptrón multicapa con dos capas ocultas. (Por ASW)

2.2.4. APRENDIZAJE DE LAS REDES NEURONALES

Como puntualiza Call [47], el conjunto de valores calculados en la capa de salida es la predicción o inferencia de la red. Para que las predicciones sean útiles, la red neuronal artificial necesita ser entrenada, lo cual significa que, como describe Munkata [4], se requieren ajustar los valores de los parámetros de entrenamiento para que el valor inferido por la red corresponda con el resultado del problema que se quiere resolver.

Este entrenamiento o aprendizaje se realiza utilizando un algoritmo iterativo de optimización que explican Munkata [4], Shalev-Shwartz & Ban-David [1], y con más detalle Heaton [46], el cual se conoce como “algoritmo de retropropagación” que consta de dos fases.

Durante la primera fase o etapa de propagación hacia adelante, los valores de entrada de la red neuronal se van transformando con la aplicación de las funciones de activación de las neuronas de cada capa utilizando su salida como entrada de la capa siguiente como lo hace ver Goodfellow [5]. Al final se aplica la función de costo y se compara la salida contra el valor esperado para la entrada.

Durante la segunda fase o etapa de retropropagación, los valores de los parámetros de aprendizaje se van ajustando desde la última capa hacia la primera de forma tal que se minimiza la función de costo. Como nos muestran Goodfellow [5], Heaton [46] y Shalev-Shwartz & Ben-David [1], este ajuste se lleva a cabo utilizando un algoritmo adicional llamado descenso estocástico de gradiente que hace el ajuste en dirección contraria al gradiente de la función de costo. Con este proceso se mejora la capacidad de la red para producir salidas precisas a partir de los datos de entrada.

2.3. APRENDIZAJE PROFUNDO

Tanto Heaton [46] como Aggarwal [23] sostienen que el gran éxito actual del aprendizaje profundo, como ha sido rebautizada el área de las redes neuronales de múltiples capas, se debe al continuo incremento en la velocidad de cómputo de los procesadores en los últimos años, el aumento en la disponibilidad de datos, y la mejora en los algoritmos. Estos factores han permitido el desarrollo de modelos más poderosos y grandes, que como afirma Cheng [9], han logrado igualar o superar el desempeño humano en dominios específicos como reconocimiento de imágenes, juegos o conducción autónoma de automóviles [48].

En la actualidad, el aprendizaje profundo proporciona un marco muy poderoso para el aprendizaje supervisado. Goodfellow [5] explica que cuando agregamos más capas y más unidades de procesamiento a cada capa, la red profunda puede representar funciones cuya complejidad se va incrementando. De acuerdo con Aggarwal [3], una gran cantidad de tareas que una persona puede realizar de manera rápida y casi sin pensar, se modelan mapeando un vector de entrada a uno de salida, y pueden ser completadas utilizando aprendizaje profundo si tenemos modelos del tamaño adecuado y se cuenta conjuntos de datos etiquetados suficientemente grandes.

Las redes prealimentadas son de gran importancia para los profesionales del aprendizaje automático y se utilizan extensamente en aplicaciones comerciales. Las redes convolucionales son un tipo particular de redes prealimentadas especializadas en reconocimiento de objetos y son un paso en el camino hacia las redes recurrentes, que son el mecanismo detrás de muchas aplicaciones actuales de procesamiento de lenguaje natural de como hace ver Goodfellow [5].

2.4. COMPRESIÓN DE REDES NEURONALES

2.4.1. MOTIVACIÓN

Los modelos del estado del arte de redes profundas llegan a tener un gran número de parámetros, como la actual GPT-3 que es considerada una de las más grandes desde 2020 con 175 mil millones de parámetros que requieren un almacenamiento mayor a 800Gb como cita O'Neill [14] y se consigna en la Wikipedia [49].

Este aumento en el tamaño de los modelos está acompañado de un incremento dramático en intensidad computacional, necesitando grandes cantidades de memoria y tiempo de procesamiento como afirma Mishra [11], requiriendo en ocasiones, semanas para realizar su entrenamiento, aún con el uso de unidades de procesamiento gráfico capaces realizar cálculos de punto flotante en paralelo.

Los procesadores gráficos consumen grandes cantidades de energía como preocupa a Hao [12], O'Neill [14] y Ganesh [13], quienes están de acuerdo en que el considerable impacto ambiental producido por los modelos del estado del arte, junto con los requerimientos estrictos de latencia y privacidad mencionados por Cheng [9], sumados a la necesidad de procesamiento en tiempo real señalada por Pokhrel [10] en dispositivos que en muchas ocasiones tienen limitaciones de recursos como puntualiza Alqahtani [8], ha despertado el interés en la compresión de modelos como hacen constar Aggarwal [23], Hubara [50] y Novac [19].

2.4.2. DEFINICIÓN

De las definiciones de Nakod [51], Cheng [9] y Abbasi-Asi & Yu [21] podemos establecer la compresión de redes neuronales como el conjunto de técnicas y procesos cuyo objetivo es reducir la complejidad de los modelos de aprendizaje profundo manteniendo al mismo tiempo su precisión y desempeño en la predicción para su implementación en dispositivos de frontera que tienen potencia computacional y memoria reducida como describen Aggarwal [23] y Alqahtani [8].

La idea principal de la compresión de modelos de acuerdo con Goodfellow [5] es reemplazar el modelo original por uno más pequeño que requiera menos memoria y tiempo de ejecución para almacenar sus valores y evaluar sus entradas.

La compresión de redes es aplicable cuando el tamaño del modelo original está obligado por la necesidad de prevenir el sobreajuste. En opinión de Arora [52], estas arquitecturas sobreparametrizadas pueden comprimirse eficientemente mientras mantienen la propiedad de generalización.

2.4.3. MÉTODOS DE COMPRESIÓN

No se ha encontrado aún en la literatura un consenso universal para la forma de clasificar los métodos de compresión de redes neuronales, pero basándonos en sus características podemos hablar de varios grupos principales entre los que se encuentran la poda de redes que se menciona por primera vez con el método de "daño cerebral óptimo" de LeCun [17], y los métodos mencionados en los trabajos de sondeo de Call [47], Mishra [11] y Alqahtani [8]; la destilación del conocimiento a la que se hace referencia desde los trabajos de Bucilli [53] y los de Ba & Caruana [54], que involucran el aprendizaje de una red más pequeña a partir de una de

mayor tamaño utilizando aprendizaje supervisado por la red mayor y minimizando la divergencia entre sus estimaciones probabilísticas según explica O'Neill [14]; la factorización que se basa en descomposición de matrices o tensores, y la cuantización, la cual se discute en la siguiente sección.

2.5. CUANTIZACIÓN

Como explican los trabajos de Gholami [55] y Nagel [56], entre las técnicas de compresión de redes neuronales, la cuantización es una de las categorías por las que se ha demostrado bastante interés en los últimos años, debido a que, siendo el entrenamiento e inferencia de redes neuronales computacionalmente intensivo, una representación numérica eficiente puede ayudar a acelerar el proceso.

En su trabajo sobre aceleración de redes, Call [47] define la cuantización como el proceso de representar valores numéricos utilizando un menor número de bits que el requerido para una representación exacta, con el consecuente error de representación. En su mayoría, las arquitecturas de procesadores están optimizadas para realizar operaciones en 8/16/32/64 bits, por lo que, en primera instancia, una cuantización que utiliza un número diferente de bits no aprovecha esta optimización.

Alqahtani [8] establece que, si se considera que la cuantización se basa en el uso de un número reducido de bits o representaciones binarias para los pesos, activaciones, y cálculos de convergencia como explica O'Neill [14], se disminuyen tanto las necesidades de almacenamiento como de cómputo, logrando soluciones rápidas al problema de compresión como concuerdan Cheng [9],Qin [24] y Gholami [55].

La sobreparametrización, a la que se refieren Gholami [55] y Arora [52], es una característica de los modelos más modernos de redes profundas, que brinda grandes oportunidades para reducir la complejidad sin impactar la exactitud, debido a que como describen Simons & Lee [26], su gran cantidad de parámetros las hace resistentes a la cuantización y discretización agresiva.

La cuantización ha sido ampliamente estudiada en un gran número de trabajos como resume Gholami [55]; desde representaciones de punto fijo, uso de enteros de 32, 16 u 8 bits, hasta estrategias con uso de 4, 2, y siendo de interés especial para este trabajo, la cuantización con representaciones de 1 bit, la cual es comúnmente conocida como binarización.

2.6. REDES NEURONALES BINARIAS

2.6.1. BINARIZACIÓN

Dentro de las técnicas de cuantización, la binarización es su forma más extrema. Yuan & Agaian [25] definen la binarización como el proceso de reducir la precisión de los parámetros de la red neuronal hasta un solo bit. El propósito no es solo ahorrar espacio de almacenamiento, sino también 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 explican Yuan & Agaian [25], Simons & Lee [26] y Qin [24], con lo cual se pueden lograr reducciones de almacenamiento que Rastegari [29] estima hasta por un factor de 32x, al mismo tiempo que se pueden tener ganancias de hasta 58x en la rapidez de entrenamiento e inferencia, en comparación con el uso de punto flotante de 32 bits.

El resultado de la binarización es una red neuronal binaria, que se caracteriza porque las activaciones (características) y los pesos en tiempo de ejecución se representan con valores de 1 bit para todas las capas ocultas, es decir, los parámetros y las operaciones de activación solo pueden tener dos valores: -1 o $+1$ como proponen Courbariaux [20], Hubara [50], Rastegari [29] y registran Yuan & Agaian [25].

Las redes binarias fueron propuestas originalmente en 2015 por Courbariaux [28]. En una red neuronal binaria las capas de entrada y salida se mantienen en punto flotante, y las capas ocultas se convierten en capas binarias reemplazando la función de activación ReLU por la función signo y el estimador directo STE para la fase de retropropagación. En su modelo BinaryConnect [20], los pesos utilizan valores binarios, tanto durante la inferencia como durante el entrenamiento. Este primer modelo emplea valores reales para las activaciones, por lo que requiere adición de punto flotante. En su siguiente trabajo, BinaryNet [20] se incluyen activaciones binarias, también probadas por Hubara [50] como refieren Simons & Lee [26].

La idea de la binarización surge de la siguiente observación: Si se restringen los valores de los pesos y las activaciones de una red neuronal únicamente a $+1$ y -1 durante el entrenamiento, podemos reemplazar las operaciones de multiplicación y acumulación de punto flotante por las operaciones XNOR y POPCOUNT de 1 bit las cuales pueden ejecutarse a mucha mayor velocidad que las de 32 bits [26].

XNOR es una operación lógica que se define como la negación del o exclusivo (XOR); en esta operación, el resultado es 1 cuando las entradas son iguales y cero cuando son diferentes de acuerdo con la tabla 1 tomada de Simons [26] y Yuan & Agaian [25], que muestra la similitud entre tabla de verdad de XNOR y la del producto de -1 y $+1$.

Tabla 1. Tabla de verdad de XNOR y producto de los valores de binarización. (Por ASW)

XNOR de valores codificados		
p	q	p XNOR q
0	0	1
0	1	0
1	0	0
1	1	1

Multiplicación de valores		
a	b	a*b
-1	-1	1
-1	1	-1
1	-1	-1
1	1	1

En la aplicación práctica, el valor -1 se codifica como 0 y $+1$ como 1, de manera que el resultado de la operación es aritméticamente correcto como se muestra en la tabla 1.

Por su lado, POPCOUNT es una instrucción presente en la mayoría de las arquitecturas modernas de procesador, la cual cuenta el número de bits con valor de 1 en una palabra.

Como expone Qin [24], la meta de la binarización es representar los pesos de punto flotante $\boldsymbol{w}$ y las activaciones $\boldsymbol{a}$ utilizando 1 bit. La función de binarización se define como sigue:

$$Q_w(\boldsymbol{w}) = \alpha \boldsymbol{b_w}\,, Q_\alpha(\boldsymbol{a}) = \beta \boldsymbol{b_a}$$

(2)

Donde $\boldsymbol{b_w}$ y $\boldsymbol{b_a}$ son el tensor de pesos binarios y las activaciones binarias con los correspondientes escalares α y β.

Para Q_w y Q_a Courbariaux [28] propone dos funciones de binarización:

Binarización determinista:

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

(3)

Donde x^b es el valor binarizado de la variable x, que es la variable con valor real (peso o activación).

Binarización estocástica:

$$x^b = \begin{cases} +1 \; con \; probabilidad \; p = \sigma(x), \\ -1 \; con \; probabilidad \quad 1 - p, \end{cases}$$

(4)

Donde σ es la función "sigmoide dura". Sin embargo, Courbariaux [20] explica que esta última requiere en el procesador circuitos para generar números aleatorios al momento de cuantizar, por lo que en la práctica la binarización determinista es la más utilizada.

Por lo tanto, siguiendo la explicación de Yuan & Agaian [25] el cálculo de una neurona se realiza como sigue:

$$Y = \sum_{i=1}^{n} b_{\boldsymbol{w}_i} b_{\boldsymbol{a}_i} + b \tag{5}$$

Donde la suma se realiza con POPCOUNT y el producto con XNOR, y la activación se calcula sobre Y:

$$Z = activación(Y) \tag{6}$$

2.6.2. *Retropropagación para redes binarias*

Adicionalmente, Courbariaux [20] discute el problema que se presenta para la aplicación del algoritmo de retropropagación con descenso de gradiente, donde la derivada de la función de binarización (como la función signo en las ecuaciones 3 y 4) puede ser no diferenciable o su derivada ser cero en todos los puntos, haciéndola incompatible con la retropropagación, pues el gradiente de la función de costo con respecto a las cantidades antes de la discretización es cero como explican Qin [24] y Yuan [57]. Normalmente el descenso de gradiente hace pequeñas actualizaciones a los pesos, las cuales no son posibles con valores binarios como establecen Simons & Lee [26].

Para abordar el problema y proporcionar un método para entrenar una red utilizando los pesos binarizados, Simons & Lee [26] explican la propuesta de Courbariaux [20], que utiliza la técnica del estimador directo (STE) presentada por primera vez por Hinton & Teleman [58], para poder entrenar una red binarizada con la función signo como exponen Yuan & Agaian [25].

$$\boldsymbol{b_w} = sign(\boldsymbol{w}) \tag{7}$$

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 [26]:

$$\frac{\partial L}{\partial \boldsymbol{w}} = \frac{\partial L}{\partial \boldsymbol{b_w}}$$

(8)

Donde L es la pérdida en la salida. Esta aproximación del gradiente se utiliza para actualizar los pesos reales.

Para binarizar las activaciones, se 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 activación es muy grande, de acuerdo con Simons & Lee [26], se cancela el gradiente en el recorrido hacia atrás utilizando (9):

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

(9)

Donde $\boldsymbol{a}$ es la entrada con valor real de la función de activación y $\boldsymbol{b_a}$ es la salida binarizada de la misma y $1_{|a|\leq 1}$ se evalúa como 1 si $|\boldsymbol{a}| \leq 1$ y como cero en otro caso.

Esto convierte el gradiente en cero si la entrada es muy grande, y puede lograrse agregando la función tangente dura antes de la función de activación signo. Esto tendrá efecto solo en la pasada hacia atrás.

El estimador directo se define como sigue de acuerdo con Qin [24] y Courbariaux [20]:

$$htanh(x) = clip(x, -1, 1) = \max(-1, \min(1, x))$$

(10)

Durante el entrenamiento, los pesos reales de cada capa se mantienen y se actualizan utilizando el estimador directo. Al concluir el entrenamiento, se almacenan los pesos binarios y los pesos reales se descartan como lo explica Courbariaux [20] y lo confirman Yuan & Agaian [25].

La aplicación de la función de binarización, se puede ver como una capa en sí misma; en la entrada se tienen pesos reales, que pasan a través de la capa de binarización y a la salida se obtienen pesos binarios durante la propagación hacia adelante, y se aplica la función de identidad durante la retropropagación de acuerdo con Simons & Lee [26], como se muestra en la ilustración 3.

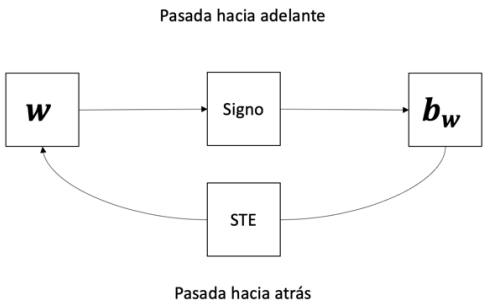


Ilustración 3. Visualización de la binarización como capa y el STE. (Por ASW)

Bethge [27] demuestra de manera empírica que en general, los modelos binarios pueden entrenarse desde cero utilizando técnicas de entrenamiento estándares sin necesidad de recurrir a técnicas más complejas.

2.7. ENTORNOS PARA MODELADO DE REDES BINARIAS.

Como exponen Yuan & Agaian [25], gran cantidad de las implementaciones publicadas de redes binarias no almacenan los parámetros en formato binario y por lo tanto no pueden utilizar las operaciones XNOR y POPCOUNT para realizar las multiplicaciones de matrices binarias tanto en capas totalmente conectadas como en capas convolucionales.

La razón es que los modelos que están implementados directamente en entornos como TensorFlow de acuerdo con Abadi [59] y PyTorch según Paszke [60], están limitados por el hecho de que Python no puede almacenar los datos en forma binaria y por lo tanto realizar operaciones entre datos tipo bit como lo hacen C y C++.

La tabla 2, recopilada también por Yuan & Agaian [25] resume los entornos que permiten modelado de redes con auténtico desempeño binario.

2.8. KERAS

2.8.1. KERAS API

La página oficial de Keras [61] describe que es una biblioteca de aprendizaje profundo que proporciona una API de alto nivel para poder definir y entrenar redes neuronales. Fue desarrollada para investigadores con la meta proporcionar una herramienta rápida de experimentación. Puede ejecutarse sobre TensorFlow, MCTK o Theano.

Tabla 2. Entornos para modelado de redes con desempeño binario. (Por ASW)

Entorno	Licencia	Formato base	Entrenamiento	Fin de mantenimiento
BMXNet	Apache	MXNet	MXNet	18/11/2019
BMXNet2	Apache	MXNet	MXNet	02/07/2020
daBNN	BSD	ONNX	PyTorch	11/11/2019
Riptide	Privada	TensorFlow/TVM	TensorFlow	13/05/2020
FINN	BSD-3	ONNX	Brevitas (PyTorch)	Actual
Larq	Apache 2.0	TensorFlow	TensorFlow	Actual

Keras y TensorFlow sirven como base para Larq [62], lo que permite utilizar muchas de las características de Keras durante el desarrollo de redes neuronales binarias.

Chollet [7], aclara que Keras se distribuye bajo permiso de la licencia MIT que permite que sea utilizada libremente en proyectos comerciales.

2.8.2. TIPOS DE CAPAS MÁS FRECUENTES DE KERAS

La capa es la estructura fundamental de las redes neuronales en Keras como explica Chollet [7]. Una capa es un módulo de procesamiento que toma como entrada uno o más tensores y genera como salida también uno o más tensores. Algunas no tienen un estado, pero más frecuentemente las capas almacenan los pesos que se aprenden con el algoritmo de descenso estocástico de gradiente. Existen distintos tipos de capas, con funciones definidas que se pueden utilizar para estructurar modelos de aprendizaje profundo:

- Densa: es la capa más básica y comúnmente utilizada en las redes neuronales, también conocida como capa completamente conectada. Conecta cada entrada a cada salida con un conjunto de pesos ajustables. Se utiliza para tareas como la clasificación de imágenes y el modelado del lenguaje.
- Convolucional: se utiliza en tareas de procesamiento de imágenes. Aplica un conjunto de filtros a la imagen de entrada para extraer características como bordes, formas y texturas. Las capas convolucionales se utilizan en tareas como el reconocimiento de objetos y la segmentación de imágenes.
- Recurrente: se emplea en tareas como el modelado del lenguaje y el reconocimiento de voz. Procesa datos secuenciales como texto, audio o datos de series temporales, manteniendo una memoria de las entradas pasadas.
- Agrupamiento: utilizada en conjunto con las capas convolucionales, reduce el tamaño de la salida de una capa convolucional mediante un muestreo descendente de las características para preservar sólo la información más importante.

- Abandono: sirve para prevenir el sobreajuste en las redes neuronales. Elimina aleatoriamente un porcentaje de las unidades de entrada durante el entrenamiento, lo que obliga a la red a aprender características más robustas.
- Normalización por lotes: esta capa mejora la estabilidad y velocidad del entrenamiento en las redes neuronales. Normaliza la entrada a cada neurona en el lote, lo que reduce el impacto de pequeños cambios en la distribución de la entrada.

2.9. LARQ

Larq es una biblioteca abierta de Python que proporciona una API construida sobre la API de TensorFlow *tf.keras* y que en palabras de Geiger [62] está diseñada para modelar y entrenar redes neuronales binarias y cuantizadas de manera sencilla. Larq se menciona formalmente en el trabajo de Bannik [63] y está enfocada tanto hacia investigadores como profesionales con deseos de explorar las redes neuronales binarias como se explica en la página oficial de Larq [64].

De manera similar en que la capa es la estructura fundamental de Keras, Larq define capas cuantizadas que son compatibles con la API de Keras y se pueden utilizar de manera conjunta para el diseño y entrenamiento de modelos neuronales, pero con la particularidad de que entre sus parámetros se cuenta con funciones de cuantización que se utilizan para cuantizar y binarizar tanto las entradas como las matrices de pesos. Larq también incluye versiones de la función signo y el estimador directo STE.

3. ESTADO DEL ARTE DE LAS REDES NEURONALES BINARIAS

3.1. DESARROLLO DEL MODELO

En la encuesta de Qin [24], se hace notar que el problema de las redes neuronales binarias es que sufren de notable degradación y pérdida de exactitud, pues como agregan Bethge [27] y Yuan & Agaian [25], la binarización de pesos y activaciones produce una gran desviación de los valores de precisión real y el algoritmo de retropropagación puede no encontrar una solución satisfactoria aún con el empleo del estimador directo.

Muchos trabajos han sido desarrollados buscando mejorar la precisión y exactitud de las redes binarias, varias ideas se han construido sobre la metodología original de estas y han realizado contribuciones a la teoría. Desde su introducción en 2015, se han realizado avances en todos los aspectos de estos modelos.

BinaryConnect de Courbariaux [28] es el modelo original de red neuronal binaria, en donde se binarizan los pesos, pero no las activaciones; este autor presenta al año siguiente BinaryNet que ya utiliza activaciones binarias [20].

El modelo XNOR-NET presentado por Rastegari [29], además de incluir todos los métodos propuestos en la BinaryConnect original, agrega un valor de ganancia para compensar la pérdida de información durante la binarización y cambia el orden de algunas capas para mejorar el entrenamiento. Sin embargo, aunque los términos de ganancia mejoran la precisión de la red, su cálculo es costoso.

Otro trabajo del mismo año expuesto por Zhou [30] introduce la DoReFa-Net, que trata de mejorar las ineficiencias de XNOR-NET, y en la cual se utilizan diferente precisión y longitud para los pesos, las activaciones y los cálculos inversos durante el entrenamiento. Sin embargo, su método es muy complejo y no exhibe mejora sobre las operaciones de bits.

Con Tang [65] , se aplican muchas ideas que ya se habían recomendado en otros trabajos; demuestra la binarización de la red AlexNet, a la que agrega una capa para utilizar factores de escala.

Lin [31], basándose en las ideas de Do-Re-Fa, desarrolla ABC-Net con activaciones binarizadas y pesos en múltiples bases, aunque su costo computacional es más alto.

La GroupNet de Zhuang [66] se enfoca en la descomposición de la red en grupos de precisión real y con la BNN+ dada a conocer por Darabi [67], se extienden los principios fundamentales de las redes binarias originales buscando alternativas a las funciones de cálculo utilizadas en las redes propuestas anteriormente.

Con la Binary DenseNet de Bethge [33], se recomienda un modelo de arquitectura para redes binarias, y se utilizan conexiones de atajo para aumentar el flujo de información entre capas al mismo tiempo que aconseja utilizar capas de reducción de precisión completa para preservar la información.

Posteriormente Helwegen [68] propone el "Optimizador Binario" BOP, específicamente diseñado para redes binarias, que elimina la necesidad de almacenar los pesos reales para el cálculo durante la retropropagación.

El trabajo de Kim [69], expone el uso de la distribución de activaciones no balanceado y capas de consolidación máxima.

El estudio "$BNN - BN = ?$", de Chen [70], demuestra el reemplazo de las capas de normalización por capas convolucionales con estandarización escalada de pesos, método que es aplicable para redes convolucionales.

Redfern introduce BCNN [36], que explora una red convolucional binaria en la que todas las operaciones de matrices están cuantizadas a 1 bit.

Con AdaBin presentada por Tu [71], se recomienda utilizar conjuntos binarios adaptativos para cada capa.

BoolNet se enfoca en el uso de una convolución binaria, e incrementar la propagación de información mediante el uso mapas de binarios de características [72].

Finalmente, merece mención el trabajo de Shi, que con RepBNN [73], introduce un mapa de características mejorado mediante repetición.

3.2. OPTIMIZACIÓN DEL MODELO

Todos los trabajos sobre redes neuronales binarias buscan soluciones para mejorar las técnicas originales. Los trabajos de Qin [24] y Yuan & Agaian [25] proponen la siguiente clasificación para los métodos de optimización de redes binarias. En particular el trabajo de Yuan & Agaian [25] incluye tablas detalladas de todos estos enfoques.

3.2.1. MINIMIZACIÓN DEL ERROR DE CUANTIZACIÓN

Reducir el error de cuantización de pesos y activaciones como explican Qin [24], y Yuan & Agaian [25], es la solución más directa similar a los mecanismos de cuantización donde el parámetro cuantizado debe aproximar al de precisión real tanto como sea posible con la expectativa de que el desempeño de modelo binario se acerque al de precisión real. Entre los enfoques investigados utilizados para lograr esta reducción se pueden mencionar la utilización

de factores de escala como demuestran Rastegari [29] y el uso de métodos alternativos a la función signo para la binarización de parámetros como lo hacen Zhou [30] y Xu [74].

3.2.2. MEJORA DE LA FUNCIÓN DE PÉRDIDA

Varios trabajos que desarrollan este enfoque reducen la brecha con las redes de valores reales; por ejemplo, Tang [65] y Darabi [67] mediante el uso funciones de pérdida para la distribución o regularización especial sobre la función de costo. Algunos proponen versiones especiales de las funciones objetivo o la regularización con hiperparámetros como lo hace Martínez [75].

3.2.3. REDUCCIÓN DEL ERROR DEL GRADIENTE

Las redes binarias se continúan entrenando con el algoritmo de retropropagación. Dado que la derivada de la función signo es cero, el problema es la actualización de pesos durante la fase de retropropagación. Para lidiar con los gradientes de la función de binarización no diferenciable, se utiliza el estimador directo de Hinton & Teleman [58] para estimar los gradientes en la retropropagación. Sin embargo, existe un desajuste obvio entre el gradiente de la función de binarización y el STE; adicionalmente se tiene el problema de que los parámetros fuera del rango y en los bordes de $[-1, +1]$ no cambian, dañando la capacidad de actualización de la retropropagación, lo cual conduce a redes suboptimizadas con problemas severos de degradación.

Entre las soluciones a este problema, se utiliza el aprendizaje basado en gradientes reales como lo hace Darabi [67] que emplea la función sigmoide en la fase de retorno, aproximación de pesos polinomiales como propone Z. Liu [32] o con aproximación del gradiente con funciones gaussianas como manifiesta C. Liu [76].

3.2.4. ARQUITECTURAS OPTIMIZADAS PARA REDES BINARIAS.

Considerando los mayores atractivos de las redes binarias, tales como la rapidez de cálculo, el reducido consumo de energía y el ahorro de memoria que permiten implementarlas en hardware de recursos limitados, un buen número de trabajos se han dado a la tarea de modificar arquitecturas tradicionales para obtener buen desempeño al binarizar.

ABC-Net [31], CBCN [76], Bi-Real-net [32], WPRN [77], GroupNet [66], BBG-Net [78] y Real-to-bin [79] proponen la alteración de redes clásicas como ResNet para mejorar su desempeño y precisión. BENN [80] toma ventaja de un ensamble de redes binarias para mejorar el desempeño en la predicción.

3.2.5. ESTRATEGIAS DE ENTRENAMIENTO

Otra vertiente de investigación desde los primeros trabajos sobre redes binarias ha sido la búsqueda de diferentes trucos y esquemas de entrenamiento que pueden afectar positivamente la exactitud final de la red. Ejemplos de este enfoque se pueden encontrar en el trabajo de Tang [65] y en el estudio de Alizadeh [81].

3.3. RESUMEN

Simons & Lee [26] concluyen que los modelos de redes neuronales binarias han mejorado a través del tiempo mediante el uso de diversas técnicas, tales como:

- Términos de ganancia, para dar a los productos un sentido de magnitud, el cual se pierde como efecto de la binarización.
- Bases múltiples, utilizando binarizaciones múltiples sobre un conjunto de entradas para compensar la pérdida de información durante la binarización.
- Binarización parcial, donde solo se binarizan las partes de la red que se benefician más de la compresión, y dejando las capas más esenciales en precisión máxima.
- Índice de aprendizaje, incrementando el ritmo para acelerar el entrenamiento.
- Relleno. Aplicando los valores $+1$ o -1 como relleno para las operaciones de convolución.
- Incremento de la binarización. Binarizar la capa de entrada y la de salida.
- Capas de normalización y activaciones como umbral.
- Cambio del orden de las capas, como utilizar una de agrupamiento antes que una capa de normalización.
- Optimizaciones enfocadas a minimizar los errores de cuantización, de cálculo del gradiente o mejorar la función de pérdida.
- Diversas estrategias de entrenamiento.
- Diseño de arquitecturas optimizadas para operaciones binarias

4. MARCO METODOLÓGICO

4.1. DEGRADACIÓN DE LAS REDES NEURONALES BINARIAS

Como se describe en el marco teórico, el problema de las redes neuronales binarias es que sufren deterioro notable en su exactitud comparadas con sus equivalentes de punto flotante.

Para el caso de un perceptrón multicapa binario, esta degradación es aún más notoria, por lo que el enfoque experimental incluye la realización de modificaciones que han sido sugeridas en la literatura del estado del arte - pero no aplicadas a perceptrones binarios - para mejorar la exactitud del modelo binario equivalente que obtenemos al binarizar un perceptrón de punto flotante. Tomando como meta la exactitud del perceptrón original, se realizan cambios buscando mejorar el desempeño.

4.2. CONSIDERACIONES PARA EL DESARROLLO

Como expone Alizadeh [81], se han hecho varios intentos no empíricos para formalizar las redes neuronales binarias y el estimador directo. En su trabajo se mencionan el punto de vista geométrico de Anderson & Berg [82] para justificar la existencia de soluciones binarias independientes del proceso de optimización y el de Li [83], que garantiza la exactitud para el entrenamiento de redes binarias asumiendo convexidad. Sin embargo, puntualiza que no se ha demostrado que el STE pueda hallar la solución de una función de pérdida particular, a pesar de que los modelos binarios han ganado niveles aceptables de exactitud en la práctica.

Tomando en cuenta la afirmación anterior, se considera que la experimentación es el enfoque más adecuado para el presente trabajo, pues hasta el momento no se tiene conocimiento de un marco conceptual teórico que explique en detalle el funcionamiento de las redes neuronales binarias.

Las ideas principales que se utilizan en este desarrollo experimental de una arquitectura de perceptrón multicapa binario residual han sido tomadas de los trabajos aplicables al perceptrón de entre los presentados en el capítulo sobre estado del arte.

Aunque Chen [70] sugiere que pueden dejar de utilizarse las capas de normalización, el método propuesto en su lugar solo es aplicable para redes convolucionales. Sin embargo, reconoce que la normalización es una técnica bien conocida para estabilizar y acelerar el entrenamiento de los modelos. Para el caso de redes binarias, las 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.

De los trabajos de Bethge [27] y He [84], se experimenta con atajos de punto flotante. De acuerdo con su experiencia el uso de esta técnica ayuda a restaurar en la red el flujo de información que se desvanece durante las operaciones de binarización. Los atajos normalmente se implementan con capas convolucionales con filtros de 1×1, pero para nuestro caso se toma un enfoque diferente, y se experimenta adicionalmente con atajos cuantizados de 8 bits.

Otra recomendación sugerida por Bethge [33] es la de evitar los cuellos de botella. Un cuello de botella en una red neuronal es la reducción del ancho de una capa a la capa siguiente como explica Szegedy [85]. Esto se logra manteniendo o aumentando el número de neuronas en cada capa con respecto a la capa anterior, lo cual permite mayor flujo de información que de otra manera se pierde al reducir el ancho de las capas. Esta recomendación se puso a prueba con modelos en los que la anchura de la red se mantiene o incrementa hasta la capa de salida.

Finalmente, Alizadeh [81] recomienda un entrenamiento más largo y tamaños de lote más pequeño para los modelos binarios, puesto que el uso de la activación con la función signo y el estimador directo requieren un número mayor de eras.

Paralelamente con estas recomendaciones, se tiene que considerar el problema del sobreajuste, es decir, la red no debe quedar tan sobreparametrizada que termine por aprender todos los ejemplos de entrenamiento y no pueda generalizar al evaluar el conjunto de prueba como advierte Chollet [7]. Esto tiene gran probabilidad de suceder cuando el número de neuronas excede el producto de características por el número de instancias de la secuencia de entrenamiento. Una de las maneras más sencillas de tratar este problema en las redes binarias es agregar capas de abandono, las cuales no agregan costo computacional.

4.3. AMBIENTE DE EXPERIMENTACIÓN

Para la realización de los experimentos de este trabajo las tablas 3 y 4 resumen la configuración utilizada.

Tabla 3. Configuración de hardware utilizada. (Por ASW)

Hardware	Descripción	Nota
Procesador:	Apple M1	Arquitectura ARM
Memoria	16Gb Unificada	128 bits LPDDR4X SDRAM
GPU	8 núcleos	Integrada al M1

Tabla 4. Configuración de software utilizada. (Por ASW)

Software		Versión
Lenguaje	Python	3.9
Bibliotecas	TensorFlow	2.5
	Larq	0.12.2
Ambiente de desarrollo	Jupyter notebooks	6.4.11

4.4. SELECCIÓN DE LA TAREA

Las series experimentales se realizan sobre tareas de clasificación, tanto multiclase como binaria. La clasificación es la tarea más comúnmente abordada en artículos de investigación sobre redes binarias, como se establece en la sección “Delimitación” del capítulo 1.

4.5. MÉTRICAS

Existen varias métricas proporcionadas por el marco de desarrollo, siendo las más representativas la exactitud, que calcula la frecuencia con la que la predicción es igual a la etiqueta; y la pérdida, que calcula la diferencia entre el valor de la etiqueta y la predicción de la red.

En este trabajo se utilizan estas dos métricas sobre los conjuntos de entrenamiento, validación y prueba, siendo la exactitud sobre la secuencia de prueba la base de comparación. Se prefirió la exactitud sobre el F1, debido a que tanto MNIST, IMDB y CIFAR10 son conjuntos de datos balanceados por lo que de acuerdo con Korstanje [86] F1 no ofrece ventajas, y Keras implementa la exactitud de manera directa. El conjunto Reuters en particular es un conjunto de datos extremadamente desbalanceado, por lo que durante el entrenamiento se procesan muchos lotes con ejemplos que pertenecen únicamente a la clase mayoritaria, lo cual produce un sesgo en el valor calculado de F1.

4.6. CONJUNTOS DE DATOS

Con la finalidad de dedicar mayor tiempo a la experimentación que al preprocesamiento de los datos, se toma ventaja de los conjuntos de datos incluidos en la biblioteca Keras. En particular, para este trabajo se utilizaron cuatro:

1. MNIST: esta es la base de datos de dígitos escritos a mano por aproximadamente 250 personas, dividida en 60,000 ejemplos de entrenamiento y 10,000 ejemplos de prueba. Es un subconjunto de las Special Database 3 y la Special Database 1 del National Institute of Standards and Technology (NIST). Se utiliza para el entrenamiento en técnicas de aprendizaje automático y reconocimiento de patrones por requerir mínimo esfuerzo para su preprocesamiento y formato. El conjunto MNIST ha sido normalizado y

las imágenes ha sido centradas mediante el cálculo del centro de masa de los pixeles, transfiriendo la imagen al centro del campo de 28x28 [87]. Esta colección ha sido probada con numerosos métodos, por lo que es un referente del área. Fue mencionada por primera vez por LeCun [88].

2. CIFAR10: está formado por 60000 imágenes en color de 32x32 pixeles divididos en 10 clases con 6000 imágenes por clase. Se divide en 50000 imágenes de entrenamiento y 10000 imágenes de prueba. Es un subconjunto etiquetado de la colección de "80 millones de pequeñas imágenes". Las clases son mutuamente exclusivas. Cada imagen se representa por un arreglo de 3072 enteros, el cual está dividido en 3 secciones de 1024 elementos cada uno que representan los canales de colores primarios rojo, verde y azul. Este conjunto es referido por Alex Krizhevsky de la Universidad de Toronto, de cuyo sitio [89] se puede descargar.
3. Large Movie Review Dataset 1.0: conocido como el conjunto IMDB, contiene 50,000 reseñas de películas clasificadas como marcadamente positivas o negativas, seleccionadas para realizar análisis binario de sentimientos. Contiene 25000 reseñas para entrenamiento y 25000 para pruebas; se puede descargar de la página de Maas [90]. La versión que está disponible en Keras se encuentra tokenizada. Este conjunto de datos se menciona por primera vez en el trabajo de Maas [91].
4. Reuters-21578, Distribution-1: su nombre completo es "conjunto de datos para categorización de texto Reuters-21578". Está compuesto por los textos de las noticias que se publicaron en "Reuters Newswire" en 1987. Los documentos fueron indexados por categorías, por personal de Reuters Ltd. y Carnegie Group Ltd. Se puso a disposición de los investigadores en 1990, y se encuentran en el repositorio Irvine Machine Learning Repository de la Universidad de California [92]. Contiene 8982 ejemplos de entrenamiento y 2246 de prueba, divididos en 46 categorías con un vocabulario de 29930 palabras [93]. Similar a la agrupación de las reseñas de IMDB, este conjunto de datos se encuentra disponible en formato tokenizado en la biblioteca Keras.

MNIST y CIFAR10 son muy utilizados en la literatura sobre compresión de redes por el hecho de que la gran mayoría de trabajos se centran en reconocimiento de imágenes con redes convolucionales.

4.7. Preprocesamiento de los conjuntos de datos

Las imágenes de MNIST están organizadas como 60000 instancias de 28x28 y las de CIFAR10 como 50000 de 32x32x3. Ambos conjuntos solo requieren un cambio de topología que se realiza mediante el método *reshape* de la biblioteca *numpy*, para que puedan funcionar como entrada de un perceptrón. Después de esta operación únicamente se normalizan los valores a punto flotante entre 0 y 1, dividiendo cada valor entre 255.

En el caso de los conjuntos de IMDB y Reuters, estos se encuentran disponibles en formato tokenizado, de manera que únicamente se realiza una operación de codificación uno a uno para

utilizarlos. Para IMDB se manejan las 10000 palabras más comunes y para Reuters las 15000 más comunes.

Por su lado las etiquetas se convierten en arreglos utilizando la utilidad de Keras *to_categorical*.

De cada conjunto de entrenamiento se separan 10,000 ejemplos para validación excepto del conjunto de Reuters, del que solo se utilizan 1000 ejemplos para validación, por ser el conjunto más pequeño.

4.8. PARÁMETROS

Al ser nuestro objetivo la búsqueda de una topología, mantenemos constantes la mayoría de los hiperparámetros para todas las pruebas.

4.8.1. PARÁMETROS DE LA RED

- Función de activación para las capas ocultas: en los modelos de punto flotante se utilizó ReLU y para las variantes binarias se empleó el estimador signo-STE con recorte de peso que proporciona Larq.
- Función de activación para la capa de salida: Para clasificación multicapa se hace uso softmax y para clasificación binaria sigmoide.

4.8.2. PARÁMETROS DE ENTRENAMIENTO

- Tamaño de lote: De acuerdo con resultados experimentales de Alizadeh [81], se recomienda un tamaño de lote de menor a 50 para reducir el error de entrenamiento en redes binarias. En los experimentos realizados se aprecia que mejora el ajuste, aunque requiere mayor número de operaciones. Para nuestro caso se utiliza un tamaño de lote de 32 que en las pruebas preliminares ha mostrado buenos resultados.
- Optimizador: se utiliza en todos los casos el RMSprop (propagación de la raíz media cuadrática) sin modificación de parámetros, tanto en los modelos de punto flotante como binarios. Este optimizador es una extensión del algoritmo de descenso estocástico de gradiente estándar que toma en consideración la historia de los gradientes anteriores al actualizar los pesos. La idea es que ajusta de manera adaptativa la tasa de aprendizaje de cada peso de la red.
- Función de pérdida: se emplea entropía cruzada categórica para la clasificación. categórica, y entropía cruzada binaria para clasificación binaria. La entropía cruzada mide la discrepancia entre la distribución de probabilidad sobre las clases que predice un modelo y la distribución real de estas. La minimización del valor mejora la precisión.
- Ritmo de entrenamiento: se utiliza el valor por defecto de Keras 0.001.

- Eras: el número de eras que se entrena un modelo se fija en 10, los experimentos realizados muestran sobreajuste con un número mayor de eras.

4.9. Modelos de perceptrón de punto flotante

Para estudiar la precisión de un perceptrón multicapa binario residual, se realiza una comparación entre el modelo de perceptrón de punto flotante que se toma como base, y el perceptrón binarizado equivalente al cual se le aplican las modificaciones propuestas: capas de normalización, de abandono y conexiones residuales. Se calculan la exactitud y la pérdida con los conjuntos de entrenamiento, validación y prueba.

4.10. Organización de las pruebas

Originalmente, el enfoque planteado consistía en aplicar el siguiente procedimiento:

1. Seleccionar tres perceptrones para cada conjunto de datos: el primero de una capa, otro de dos capas y un último de tres capas.
2. Entrenar y evaluar cada uno de los perceptrones seleccionados.
3. Construir, entrenar y evaluar la versión binaria de cada perceptrón.
4. Registrar los valores de exactitud obtenidos en los pasos anteriores.
5. Realizar modificaciones en la arquitectura de cada perceptrón binario con el objetivo de mejorar la exactitud. Se adicionan capas y se observa el cambio en el resultado
6. Identificar la mejor variante obtenida después de las modificaciones.
7. A partir de la mejor variante de cada perceptrón, crear tres modelos adicionales, cada uno utilizando un tipo diferente de atajo.
8. Evaluar el desempeño de cada uno de los modelos adicionales.
9. Seleccionar el perceptrón que obtenga el mejor desempeño entre los modelos adicionales, considerando la exactitud como métrica de evaluación.

Posteriormente se detectó la necesidad de realizar pruebas exhaustivas como se describe en el siguiente capítulo.

4.11. Consolidación y análisis

Los resultados de cada prueba se reúnen en una hoja de cálculo que agrupa los resultados de todas las combinaciones para un número determinado de capas, y finalmente se realizan tanto un análisis de sensibilidad de la tabla para identificar los componentes relevantes de acuerdo con la exactitud obtenida, así como un cálculo de exactitud promedio por componente, que se compara con los resultados de un análisis de regresión lineal multivariable, lo cual ayuda a identificar los componentes adicionales que tienen una mayor contribución a los resultados, lo que nos brinda mayor información para llegar a conclusiones.

5. EXPERIMENTACIÓN Y RESULTADOS

5.1. ARQUITECTURA PROPUESTA

Para el desarrollo del perceptrón multicapa binario residual, se consideraron las recomendaciones presentadas particularmente por Bethge [33], Li [83] y Kim [69], así como los componentes descritos en la sección “Consideraciones para el desarrollo” del capítulo 4, 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 como establece Courbariaux [28], al tiempo 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 capa de normalización por lotes como se muestra en la ilustración 4. Este bloque se repite según el número de capas requeridas por el modelo. Si el perceptrón tiene al menos una capa oculta se agrega una conexión de atajo, pero ajustando su anchura a la de la última capa oculta.

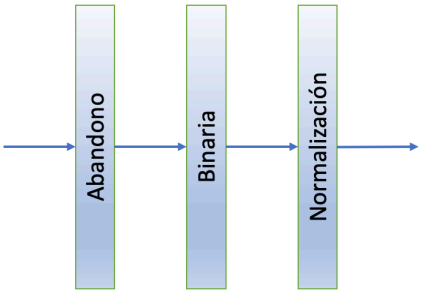


Ilustración 4. Bloque de construcción de un perceptrón multicapa binario. (Por ASW)

5.2. IMPLEMENTACIÓN DE ATAJOS

La función del atajo es pasar información desde la capa de entrada hasta el final de la red como propone Liu [32], donde se combina con la salida de la última capa oculta para contribuir al valor resultante en la capa de salida.

Por lo general las conexiones residuales se introducen utilizando capas de convolución con filtros de 1×1 cuando se trata de redes convolucionales. Para el caso del perceptrón multicapa se toma un enfoque diferente mediante capas densas sin activación y también mediante capas de agrupamiento, que se instrumentan como se describe a continuación:

- Punto flotante empleando una capa totalmente conectada *Dense* sin función de activación. Esta capa genera una salida del ancho de la última capa oculta con la que se suma antes de pasar a la capa de salida.
- Punto flotante utilizando una capa de agrupamiento máximo de una dimensión *MaxPooling1D*. Esta implementación requiere otra capa adicional totalmente conectada sin activación *Dense* para ajustar su tamaño con el de la última capa oculta.
- Cuantizada mediante *QDense, una* capa totalmente conectada cuantizada sin función de activación. Para los experimentos, se utilizó cuantización de 8 bits, lo cual reduce la memoria requerida a la cuarta parte de la que requiere la conexión residual con *Dense*.

El atajo se suma con la salida de la última capa oculta mediante una operación de suma de capas utilizando el método de Keras *layers.add(),* antes pasar a la capa de salida.

5.3. Modelo propuesto

Utilizando el bloque de construcción y la conexión residual sumada con la salida del último bloque de capa oculta, se obtiene un perceptrón binario multicapa residual que se puede ver en la ilustración 5.

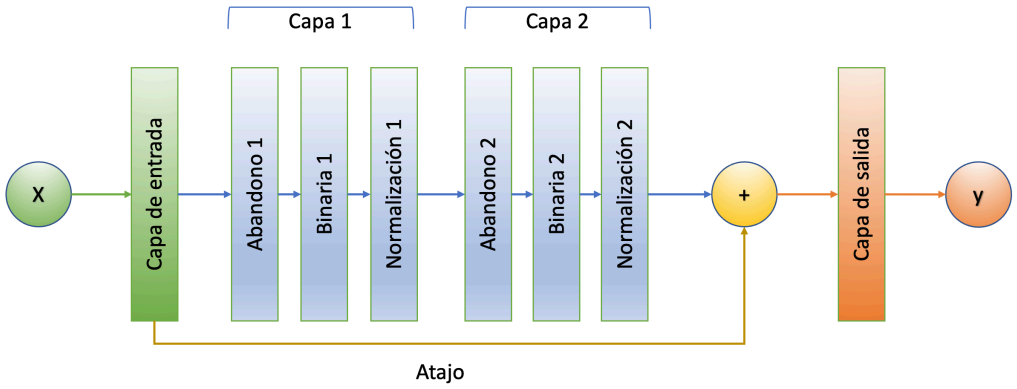


Ilustración 5. Perceptrón multicapa binario residual de dos bloques ocultos. (Por ASW)

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

5.4. TAMAÑO DE LOS MODELOS

5.4.1. TAMAÑO DEL PERCEPTRÓN DE PUNTO FLOTANTE

Para un perceptrón multicapa compuesto de $T-1$ capas ocultas, en las que cada neurona perteneciente a una capa dada está conectada con todas las salidas de las neuronas de la capa anterior; utilizando la notación de la sección "Perceptrones multicapa" del capítulo 2, podemos representar el número de parámetros para una capa $t \in [1, T]$ como:

$$P_t = (|V_{t-1}| + 1)|V_t|$$

(11)

Si cada parámetro se representa mediante un valor de m bits entonces, utilizando la ecuación 11, el número de bits S requerido por el perceptrón se calcula mediante la suma:

$$S = \sum_{t=1}^{T} m(|V_{t-1}| + 1)|V_t|$$

(12)

5.4.2. TAMAÑO DE LOS MODELOS BINARIOS.

Cuando se implementa una capa cuantizada, se utilizan k bits para representar cada valor. En los modelos binarios, las entradas a cada capa están binarizadas mediante el uso de la función signo como se establece en la sección 2.6.1 "Binarización", por lo que al tener una longitud de $k = 1$ bit, la anchura en la entrada es igual al número de parámetros; sin embargo, como la salida de la capa se calcula en punto flotante con $m = 32$ bits y la capa de salida se calcula también en punto flotante, el tamaño del perceptrón binario en bits se expresa:

$$S = \sum_{t=1}^{T-1} (k|V_{t-1}||V_t| + m|V_t|) + m(|V_T| + 1)|V_{t-1}|$$

(13)

5.4.3. TAMAÑO DE LAS CAPAS ADICIONALES

Las capas de normalización se caracterizan por utilizar punto flotante y el mismo ancho tanto de entrada como de salida, por lo que si m es el número de bits utilizado para representar un valor de punto flotante, una capa de normalización tendrá un requerimiento de memoria que queda establecido por:

$$S_N = 2m|V_t|.$$

(14)

Las capas de abandono no almacenan ningún parámetro por lo que su tamaño es cero.

5.4.4. TAMAÑO DE LOS ATAJOS

La memoria requerida por una conexión residual varía dependiendo del tipo de su tipo.

Atajos de punto flotante: Este atajo se representa mediante una capa totalmente conectada, con un número de parámetros de entrada $|V_0|$, y una salida de tamaño $|V_{T-1}|$, por lo que utilizando un número de m bits para representar un valor de punto flotante, el tamaño se puede expresar mediante:

$$S_F = m(|V_0| + 1)|V_{T-1}|$$

(15)

Atajos cuantizados: Esta conexión residual se implementa con una capa totalmente conectada cuantizada, y de manera similar a como sucede con las capas binarias, la entrada de la capa esta cuantizada utilizando k bits, y la salida en punto flotante con m bits y con $k < m$, por lo que el tamaño del atajo cuantizado será menor que el de punto flotante y se calcula como:

$$S_Q = k|V_0||V_{T-1}| + m|V_{T-1}|$$

(16)

Si k toma el valor de 1, el tamaño es similar al de una capa binaria, y si $k = m$ entonces el tamaño de la capa es similar al de una capa de punto flotante.

Atajos basados en capas de agrupamiento: Este atajo está diseñado mediante una serie de capas de Keras: Primero se utiliza una capa *Reshape* para que la entrada sea compatible con la capa de agrupamiento *MaxPooling1D*. En seguida de esta se utiliza una capa de aplanado *Flatten* para ajustar la entrada a una *Dense* cuya salida tiene el ancho de la última capa oculta. Con excepción de la capa *Dense*, ninguna de estas capas almacena valores. La capa *MaxPooling1D* requiere un parámetro que especifica el tamaño del agrupamiento, que expresamos mediante p, por lo que si $p > 1$, entonces el atajo tendrá un tamaño menor que el de punto flotante, en cambio si $p = 1$ entonces la conexión residual requiere una cantidad de memoria similar a la del atajo de punto flotante, por lo que el cálculo del tamaño para este se escribe:

$$S_P = m\left(\frac{|V_0|}{p} + 1\right)|V_{T-1}|$$

(17)

Finalmente, si consolidamos las tres ecuaciones 15, 16 y 17, el tamaño de un atajo se calcula de la siguiente forma:

$$S_A = k\frac{|V_0|}{p}|V_{T-1}| + m|V_{T-1}|$$

(18)

Donde:

k - es el número de bits utilizados para cuantización de la capa. Por ejemplo, si k=32, entonces tenemos un atajo de punto flotante; si $k = 1$ entonces estamos en el caso de una capa binaria y si $k = 8$ obtenemos una capa cuantizada de 8 bits.

m - es el número de bits utilizado para representar un valor de punto flotante, el valor por omisión es 32.

p - es el tamaño de grupo para atajos basados en capas de agrupamiento. Si $p = 1$, entonces se tiene el caso de no tener este tipo de capa.

5.4.5. TAMAÑO DEL MODELO PROPUESTO

Finalmente, el tamaño máximo de un modelo binario con T capas de normalización, utilizando la ecuación 18 y si incluimos también la ecuación 14 queda establecido por:

$$S = \sum_{t=1}^{T}(k|V_{t-1}||V_t| + m|V_t|) + m(|V_T| + 1)|V_{t-1}| + S_A + \sum_{t=1}^{T} 2m|V_t|$$

(19)

La selección de los valores k, m y p, permite ajustar el tamaño del modelo de acuerdo con las necesidades y restricciones de almacenamiento.

5.5. ORGANIZACIÓN DE LAS PRUEBAS

5.5.1. CONJUNTOS DE DATOS E HIPERPARÁMETROS

Como se describió en el capítulo 4, 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 5 muestra un resumen de las características de cada grupo de datos.

Tabla 5. Resumen de los conjuntos de datos utilizados. (Por ASW)

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 en 32x32 color 24 bits	3072	10	50000	10000

Como se explicó en el marco metodológico, para mantener el enfoque en el cambio de la exactitud al modificar la arquitectura de la red, para todos los casos de estudio se utilizó un conjunto de valores similar para los hiperparámetros. Estos valores se resumen en la Tabla 6.

Tabla 6. Parámetros de prueba para los experimentos. (Por ASW)

Parámetro	Valor
Eras	10
Tamaño de lote	32
Ritmo de entrenamiento	0.001
Ritmo de abandono	0.05
Cuantización	8 bits
Tamaño de agrupamiento	8

5.5.1. CONSTRUCCIÓN DE LOS MODELOS

En un principio la codificación de cada modelo binario se realizó manualmente, pues se esperaba que no fueran requeridas gran cantidad de pruebas una vez determinada la arquitectura básica y la influencia de las diferentes capas en el modelo; sin embargo, después del análisis inicial de las primeras pruebas se llegó a la conclusión de que eran necesarias pruebas experimentales exhaustivas para cada modelo, por lo que se tomó la decisión de desarrollar un generador de redes con la capacidad de construir, entrenar y evaluar cualquier modelo de perceptrón multicapa binario residual de una a cuatro capas ocultas, lo que permitió probar todas las combinaciones posibles de una arquitectura.

El generador de modelos utiliza una representación de la arquitectura del perceptrón consistente en dos listas:

Ancho de capas. Esta lista tiene seis elementos ordenados:

$$[\text{entrada}, \text{uno}, \text{dos}, \text{tres}, \text{cuatro}, \text{salida}]$$

Donde cada elemento representa el ancho de la capa indicada en el nombre de parámetro.

Para representar un perceptrón con un número específico de capas, en la lista se coloca el número de unidades de procesamiento de cada capa y el resto de las posiciones toman valor cero, y en la última posición se coloca el número de unidades de la capa de salida. No se permiten ceros intermedios entre capas, por ejemplo: $[128,16,8,0,0,10]$ representa una red con una entrada de 128, salida de 10 y dos capas ocultas, una de 16 y otra de 8 unidades.

Configuración de capas. Esta lista puede tener hasta trece elementos dependiendo del número de capas indicado en la lista de ancho de capas. En esta lista cada elemento puede tomar únicamente los valores de 0 o 1, y se codifica como sigue:

$$[\text{N}, \text{P}, \text{F}, \text{Q}, \text{D1}, \text{N1}, \text{D2}, \text{N2}, \text{D3}, \text{N3} , \text{D4}, \text{N4}, \text{D}]$$

En donde cada elemento toma el valor de 1 si:

N – se normaliza el atajo antes de sumarlo a la última capa oculta.

P – se utiliza un atajo de punto flotante implementado con capa de agrupamiento.

F – se emplea un atajo de punto flotante codificado con una capa totalmente conectada.

Q –se aplica un atajo cuantizado de 8 bits.

Dx –se emplea una capa de abandono antes de la capa oculta $x \in \{1,4\}$.

Nx –se utiliza una capa de normalización después de la capa oculta $x \in \{1,4\}$.

D –se coloca una capa de abandono antes de la capa de salida.

Estas arquitecturas se generan secuencialmente obteniendo una representación binaria del número de modelo, y utilizando este arreglo binario como variante a generar. Esta convención nos da un máximo de 2^{14} modelos posibles, de los cuales se omiten las combinaciones no válidas, como tener más de un tipo de atajo en una variante o capa de normalización N cuando no hay atajo.

5.5.2. *IDENTIFICACIÓN DE VARIANTES*

Los modelos se identifican mediante su descripción, para la que se utiliza una notación de la forma $(\mathrm{M} - \mathrm{DxN}, \mathrm{DxN}, \mathrm{DxN}, \mathrm{A}, \mathrm{N}, \mathrm{D_s})$ en donde:

M - designa a un modelo de punto flotante (F) o binario (B).

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 oculta.

A - esta posición señala el uso de una conexión residual. Los valores posibles pueden ser:

- F: denota un atajo de puto flotante construido mediante capa *Dense*.
- P : refiere un atajo de punto flotante implementado mediante una capa *MaxPooling1D*, con tamaño de grupo = 8.
- Q - indica un atajo instrumentado mediante una capa *Dense* cuantizada en 8 bits.

N - señala que se utiliza una capa de normalización después de una capa totalmente conectada o un atajo.

D_s – especifica que se utiliza una capa de abandono antes de la capa de salida

5.5.3. *NÚMERO DE VARIANTES*

De acuerdo con el número de capas ocultas, se determinó el número de combinaciones válidas de capas adicionales y conexiones residuales que se presenta en la tabla 7.

Tabla 7. Número de variantes válidas por número de capas ocultas. (por ASW)

Capas ocultas	Variantes válidas
1	56
2	224
3	896

5.6. DESCRIPCIÓN Y ANÁLISIS DE LOS RESULTADOS

Para evaluar la efectividad de una arquitectura particular de un modelo de perceptrón multicapa binario residual, se realizaron pruebas en las que se generaron todas las combinaciones de capas de abandono, normalización y atajos sobre un diseño base definido por el ancho y número de sus capas ocultas. Los modelos resultantes se entrenaron y evaluaron en términos de su exactitud y su tamaño. Los resultados consolidados de estos experimentos se presentan mediante las tablas y gráficas en esta sección.

Nótese que en atajos cuantizados y de punto flotante el número de parámetros es igual al producto del ancho de la capa de entrada por el ancho de la última capa oculta, aunque el atajo cuantizado requiere la cuarta parte de memoria que el de punto flotante si utiliza representación de 8 bits, por lo que el este tipo de conexión residual siempre es más económico que el de precisión real, que incrementa el tamaño del perceptrón en proporción al ancho de la última capa oculta, pudiendo en ocasiones superar el tamaño del modelo original cuando la primera y última capa tienen un número de unidades similar.

Por otro lado, para los atajos basados en capas de agrupamiento, el número de parámetros se reduce en proporción inversa al tamaño de grupo seleccionado.

Estas consideraciones posibilitan conocer a priori si una arquitectura tiene un tamaño conveniente aún antes de ser construida como se discute en la sección “Tamaño de los modelos” de este capítulo. Con respecto a las capas de abandono, el hecho de que no incrementan el tamaño, pero si mejoran la exactitud cuando se utilizan en combinación con las capas de normalización, sugiere centrar el enfoque en estas últimas durante el análisis de los resultados.

Las tablas y las gráficas presentadas en las secciones siguientes muestran la exactitud promedio que se alcanza con el uso de cada capa en particular. Estos valores están respaldados por un análisis de regresión multivariable realizado para cada serie de experimentos. Adicionalmente se intentó identificar relaciones mediante matrices de correlación midiendo la relación entre capas, pero sin resultados válidos. Para proporcionar mayor referencia, las tres primeras líneas de cada tabla muestran la exactitud alcanzada por los modelos de punto flotante, los modelos binarios equivalentes y la variante con mejor desempeño del modelo propuesto. Las gráficas permiten apreciar de manera visual los valores concentrados en las tablas.

También para cada conjunto de datos se incluye una tabla que muestra el tamaño resultante para cada arquitectura particular, y su tamaño relativo con respecto al del modelo base de punto flotante. En general y siguiendo los razonamientos de la sección “Tamaño de los modelos”, se establece que las conexiones de atajo son los componentes que mayor impacto tienen en la cantidad de memoria requerida por de un perceptrón binario residual, debido principalmente a que utilizan parámetros de longitud mayor a un bit, pero al mismo tiempo son las que tienen mayor contribución a la mejora del resultado de la red.

5.6.1. Resultados con el conjunto MNIST

Para MNIST, se utilizaron como base los modelos de punto flotante que se resumen en la tabla 8. Las arquitecturas binarias probadas correspondieron con el número y ancho de las capas de los perceptrones de punto flotante. La capa de entrada tiene un ancho de 784, que

corresponde con el número de pixeles de la imagen de 28x28 y la de salida 10, que se equipara con el número de clases. Para este conjunto de datos se realizaron pruebas exhaustivas de todas las combinaciones, incluyendo capas de abandono, normalización de capas binarias y normalización de atajos de cada uno de los modelos base.

Tabla 8. Modelos base de punto flotante base para pruebas con MNIST. (Por ASW)

Capas ocultas	Descripción	Unidades/capa			Variantes probadas
		1	2	3	
1	(F-128)	128			56
2	(F-128,88)	128	88		224
3	(F-128,128,88)	128	88	88	896

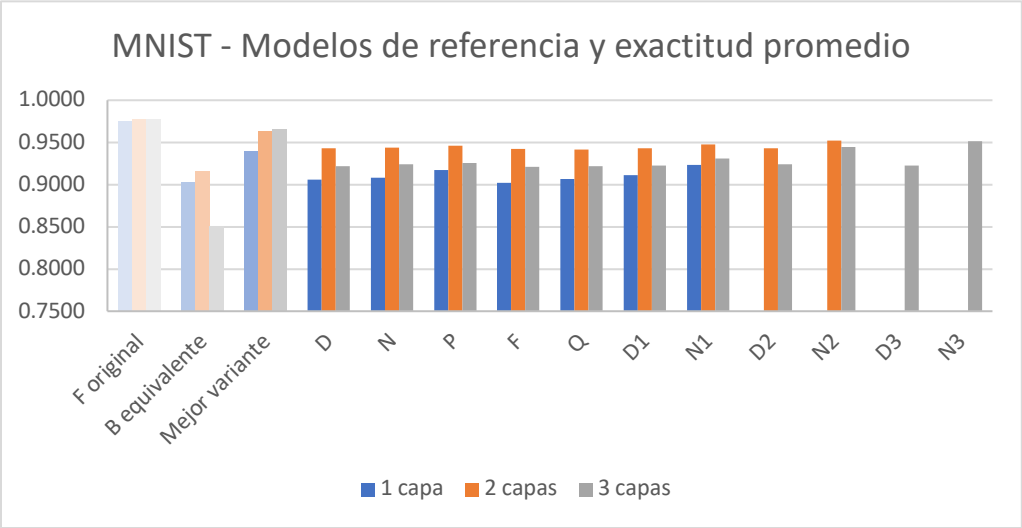
En resumen, para esta serie de pruebas de perceptrón multicapa binario residual sobre el conjunto de datos MNIST, se observa en la tabla 9 que los perceptrones originales de punto flotante obtienen la mayor exactitud entre todas las variantes, y que las tres arquitecturas binarias equivalentes presentan la menor exactitud.

De entre las combinaciones probadas, las que utilizan capas de normalización después de cada capa totalmente conectada tuvieron mejor desempeño, el cual mejora con la utilización de atajos basados en capas de agrupamiento.

Tabla 9. Resultados de exactitud promedio consolidada con MNIST. (Por ASW)

Capa	*1 capa*	*2 capas*	*3 capas*
F original	0.9749	0.9771	0.9772
B equivalente	0.9024	0.9162	0.8510
Mejor variante	0.9401	0.9630	0.9652
D	0.9061	0.9429	0.9221
N	0.9080	0.9437	0.9241
P	0.9174	0.9464	0.9256
F	0.9022	0.9420	0.9214
Q	0.9069	0.9419	0.9218
D1	0.9110	0.9433	0.9223
N1	0.9238	0.9479	0.9309
D2		0.9428	0.9238
N2		0.9520	0.9448
D3			0.9225
N3			0.9514

En la gráfica 1 se ilustra la exactitud promedio de cada componente para las tres capas. Se puede apreciar que las capas de normalización son las que tienen mayor impacto en el modelo, seguidas de los atajos. La capa de normalización más cercana a la capa de salida es la que tiene mayor influencia en la exactitud del perceptrón.



Gráfica 1. Exactitud promedio consolidada para modelos sobre MNIST. (Por ASW)

Estos resultados sugieren que, para la tarea de clasificación de imágenes, los diseños con capas de normalización y atajos proporcionan mejores resultados que el perceptrón binario equivalente y con un tamaño adecuado para una implementación en ambientes de recursos limitados. Como se aprecia en la tabla 10, la mejor variante propuesta de una capa oculta tiene un tamaño del 0.16 del tamaño de la arquitectura de punto flotante original, La mejor combinación de dos capas requiere un 0.11 mediante una capa de agrupamiento al igual que en modelos de 3 capas. Esto confirma el ahorro de memoria cuando se utilizan atajos basados en capas de agrupamiento. Por el contrario, las variantes que emplean atajos de punto flotante llegan a requerir un espacio de hasta 0.66 del tamaño original en la versión de dos capas y hasta 1.04 en el modelo de una capa oculta

5.6.1. Resultados con el conjunto IMDB

La colección IMDB se preparó con una entrada de 10000 características y al tratarse de clasificación binaria, solo una unidad de salida. La tabla 11 resume las tres arquitecturas base utilizadas para las pruebas. Para este conjunto se realizaron pruebas exhaustivas de todas las combinaciones posibles para una y dos capas, incluyendo capas de abandono, normalización de capas binarias y normalización de atajos. Para el caso de tres capas se tomó una muestra aleatoria de 63 variantes.

Tabla 10. MNIST. Comparación entre tamaño y exactitud para 1 a 3 capas ocultas. (Por ASW)

Ocultas	Descripción	Modelo	Exactitud	Tamaño	Tam. rel.
1	Flotante	F-128	0.9749	397.00	1.00
	Atajo P	B-D128N,P	0.9340	62.79	0.16
	Atajo Q	B-D128N,Q	0.9206	117.29	0.30
	Atajo F	B-D128N,F	0.9243	411.29	1.04
	Mejor	B-128N,PN,D	0.9401	63.79	0.16
	Binario	B-128	0.9024	17.79	0.04
2	Flotante	F-128,88	0.9771	440.32	1.00
	Atajo P	B-D128N,D88N,P	0.9577	49.88	0.11
	Atajo Q	B-D128N,D88N,Q	0.9562	87.35	0.20
	Atajo F	B-D128,D88N,F	0.9556	290.16	0.66
	Mejor	B-D128N,D88N,PN	0.9630	50.57	0.11
	Binario	B-128,88	0.9162	17.95	0.04
3	Flotante	F-128,88,88	0.9772	470.91	1.00
	Atajo P	B-D128N,D88N,D88N,P	0.9591	51.86	0.11
	Atajo Q	B-D128N,D88N,D88N,Q	0.9520	89.33	0.19
	Atajo F	B-D128N,D88N,D88N,F	0.9576	292.14	0.62
	Mejor	B-D128N,D88N,D88N,PN,D	0.9652	52.55	0.11
	Binario	B-128,88,88	0.8510	19.23	0.04

En esta serie de pruebas sobre el conjunto de datos IMDB, se puede apreciar en la tabla 12 que, con una capa oculta, el perceptrón de punto flotante original obtiene la mayor exactitud, mientras que el modelo binario equivalente muestra la menor exactitud.

Tabla 11. Modelos base de punto flotante para las pruebas con IMDB. (Por ASW)

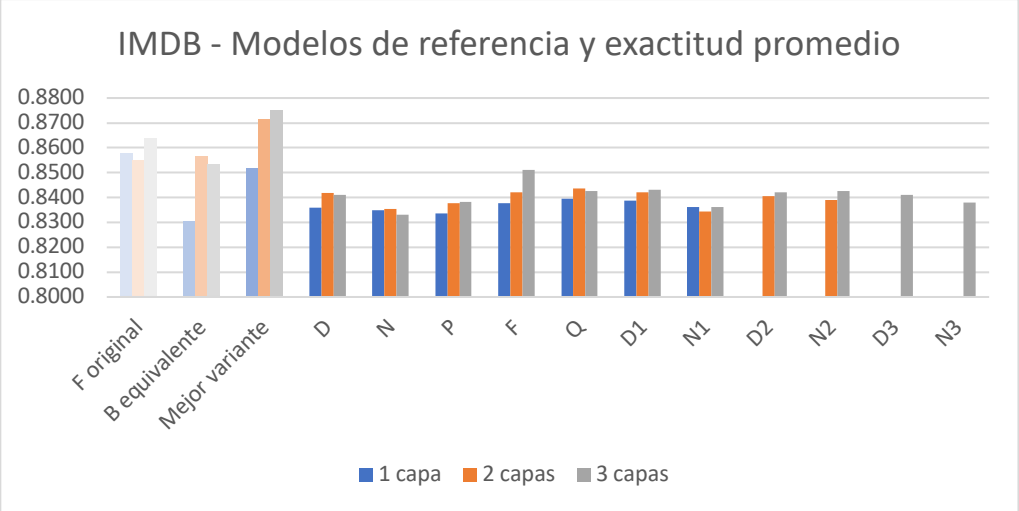
Capas ocultas	Descripción	Unidades / capa			Variantes probadas
		1	2	3	
1	(F-250)	250			56
2	(F-32,32)	32	32		224
3	(F-32,16,8)	32	16	8	63

En cambio, para arquitecturas de dos capas, el perceptrón binario equivalente supera al modelo original, y las variantes modificadas obtienen también resultados superiores al de punto flotante.

Tabla 12. Resultados de exactitud promedio consolidada con IMDB. (Por ASW)

Modelo / Capa	1 capa	2 capas	3 capas
F original	0.8577	0.8549	0.8637
B equivalente	0.8303	0.8565	0.8533
Mejor variante	0.8515	0.8713	0.8748
D	0.8359	0.8417	0.8411
N	0.8348	0.8355	0.8330
P	0.8336	0.8378	0.8383
F	0.8377	0.8421	0.8511
Q	0.8396	0.8437	0.8425
D1	0.8386	0.8421	0.8431
N1	0.8361	0.8345	0.8363
D2		0.8404	0.8421
N2		0.8390	0.8425
D3			0.8410
N3			0.8380

En las redes de tres capas basadas en (F-32, 16, 8) se supera también al perceptrón de punto flotante original. Esto puede percibirse en la gráfica 2.



Gráfica 2. Exactitud promedio consolidada para modelos sobre IMDB. (Por ASW)

Sin embargo, el uso de atajos de punto flotante incrementa el tamaño del perceptrón residual sobre el tamaño del diseño de punto flotante. Las combinaciones con atajos cuantizados

obtienen buen resultado con significativamente menor tamaño para ambos escenarios de una y dos capas.

Tabla 13. IMDB. Comparación entre tamaño y exactitud para 1 a 3 capas ocultas. (por ASW)

Ocultas	Descripción	Modelo	Exactitud	Tamaño	Tam. rel.
1	Flotante	F-250	0.8577	9767.00	1.00
	Atajo P	B-D250N,P	0.8320	1395.03	0.14
	Atajo Q	B-D250N,Q	0.8500	2751.47	0.28
	Atajo F	B-D250N,F	0.8463	10075.69	1.03
	Mejor	B-D250N,Q,D	0.8514	2751.27	0.28
	Binario	B-250	0.8303	307.13	0.03
2	Flotante	F-32,32	0.8549	1250.38	1.00
	Atajo P	B-D32N,D32N,P	0.8266	179.07	0.14
	Atajo Q	B-D32N,D32N,Q	0.8432	352.69	0.28
	Atajo F	B-D32N,D32N,F	0.8474	1290.19	1.03
	Mejor	B-32,32,Q	0.8698	352.19	0.28
	Binario	B-32,32	0.8565	39.57	0.03
3	Flotante	F-32,16,8	0.8637	1252.75	1.00
	Atajo P	B-D32N,D16N,D8N,P	0.8338	74.58	0.06
	Atajo Q	B-D32N,D16N,D8N,Q	0.8468	117.99	0.09
	Atajo F	B-D32N,D16N,D8N,F	0.8362	352.36	0.28
	Mejor	B-32N,D16N,D8,Q,D	0.8652	117.93	0.05
	Binario	B-32,16,8	0.8533	39.39	0.02

En la tabla 13 se observa que los modelos binarios equivalentes pueden representarse con 0.03 de la memoria del perceptrón original. Cuando agregamos capas de normalización y conexiones residuales para mejorar la exactitud, llegamos hasta 0.28 del tamaño del original. Como resultado notable, la mejor variante de 3 capas obtiene una exactitud mayor a la de la arquitectura original de punto flotante, con solo un 0.05 del tamaño del original utilizando atajos basados en cuantización.

5.6.2. Resultados con el conjunto Reuters

Para abordar este caso, la capa de entrada se limitó a las 15000 palabras más comunes que aparecen en las noticias, que corresponde con el ancho de la capa de entrada. El número de clases es 46, que determina el ancho de la capa de salida. Como en los casos anteriores se realizaron pruebas exhaustivas de todas las combinaciones válidas de modelos de una y dos

capas. Para el caso de tres capas se evaluó una muestra aleatoria de 62 variantes como se resume en la tabla 14, donde se muestran también las arquitecturas base.

Tabla 14. Modelos base de punto flotante para las pruebas con Reuters. (Por ASW)

Capas ocultas	Descripción	Unidades / capa			Variantes probadas
		1	2	3	
1	(F-256)	256			65
2	(F-128,64)	128	64		224
3	(F-128,64,32)	128	64	32	62

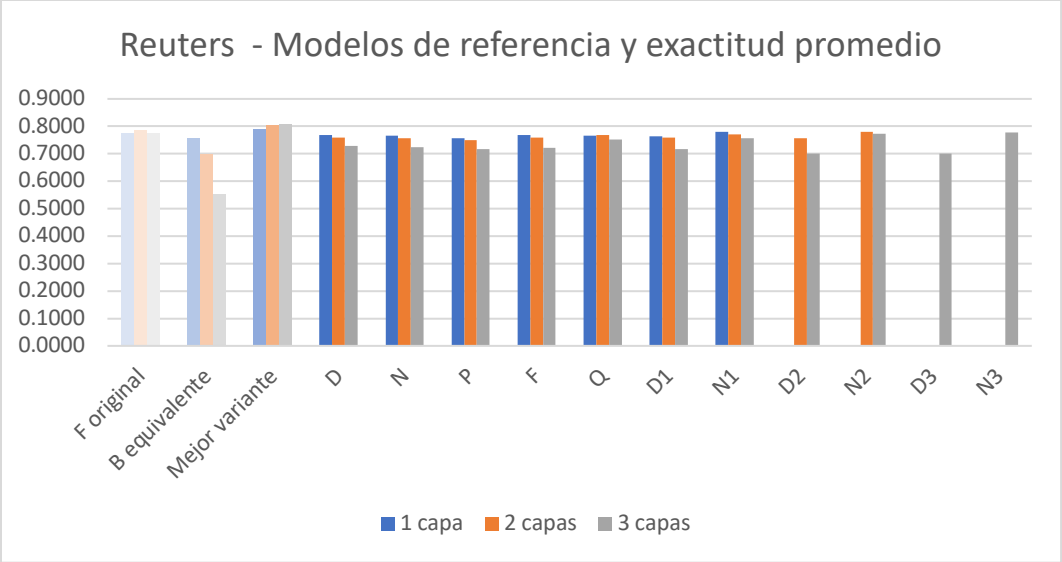
Interesantemente, para este conjunto de prueba todos los modelos binarios con capas de normalización y atajos superan la exactitud del perceptrón original, como consta en la tabla 15.

Tabla 15. Resultados de exactitud promedio consolidada con Reuters. (Por ASW)

Modelo / Capa	1 capa	2 capas	3 capas
F original	0.7734	0.7850	0.7752
B equivalente	0.7565	0.6963	0.5505
Mejor variante	0.7907	0.8037	0.8072
D	0.7672	0.7601	0.7280
N	0.7668	0.7578	0.7236
P	0.7577	0.7505	0.7167
F	0.7674	0.7592	0.7216
Q	0.7669	0.7673	0.7524
D1	0.7635	0.7579	0.7163
N1	0.7792	0.7704	0.7557
D2		0.7576	0.7018
N2		0.7789	0.7724
D3			0.7001
N3			0.7767

Los resultados de las variantes de perceptrón multicapa binario residual sobre el conjunto de datos Reuters, permiten observar que estos pueden superar a los modelos originales de punto flotante, mejorando con mucho a los diseños binarios equivalentes.

De los modelos modificados, los que utilizan capas de normalización tuvieron mejor desempeño, con las mejoras que brindan los atajos cuantizados y los atajos de punto flotante, aunque estos últimos no llegan al tamaño del perceptrón base, la opción cuantizada es más adecuada.



Gráfica 3. Exactitud promedio consolidada para modelos sobre Reuters. (Por ASW)

La gráfica 3 confirma la contribución de las capas de normalización N1 a N3 y los atajos Q.

Tabla 16. Reuters. Comparación entre tamaño y exactitud para 1 a 3 capas. (Por ASW)

Ocultas	Descripción	Modelo	Exactitud	Tamaño	Tam. Rel.
1	Flotante	F-256	0.7734	15047.00	1.00
	Atajo P	B-D256N,P	0.7640	2184.93	0.15
	Atajo Q	B-D256N,Q	0.7827	4268.93	0.28
	Atajo F	B-D256N,F	0.7836	15518.93	1.03
	Mejor	B-256N,QN	0.7867	4270.93	0.28
	Binario	B-256	0.7564	515.93	0.03
2	Flotante	F-128,64	0.7850	7544.43	1.00
	Atajo P	B-D128N,D64N,P	0.7738	666.05	0.09
	Atajo Q	B-D128N,D64N,Q	0.7818	1187.05	0.16
	Atajo F	B-D128N,D64N,F	0.8018	3998.55	0.53
	Mejor	B-D128,D64N,QN,D	0.8018	1186.35	0.16
	Binario	B-128,64	0.6963	247.80	0.03
3	Flotante	F-128,64,32	0.7752	7546.00	1.00
	Atajo P	B-D128N,D64N,D32N,P	0.7627	452.55	0.06
	Atajo Q	B-D128N,D64N,D32N,Q	0.7809	713.05	0.09
	Atajo F	B-D128N,D64N,D32N,F	0.7853	2119.30	0.28
	Mejor	B-D128,D64N,D32N,Q,D	0.8072	713.05	0.09
	Binario	B-128,64,32	0.5505	242.43	0.03

La tabla 16 muestra que los requerimientos de memoria de los mejores modelos van de 0.28 del tamaño original para una capa, hasta un mínimo de 0.09 en una arquitectura de tres capas, con mejor exactitud que los modelos originales.

Los resultados obtenidos con los conjuntos IMDB y Reuters sugieren que los atajos cuantizados funcionan mejor para entradas preprocesadas utilizando codificación uno a uno como es el caso de textos.

5.6.3. RESULTADOS CON EL CONJUNTO CIFAR10

El conjunto de datos CIFAR-10 está conformado por imágenes de color de 32 x 32 pixeles y tres canales de color, utilizando un byte para cada color primario, lo que resulta en una entrada al modelo de 3072 características. Para la salida se tienen diez categorías. Se realizaron pruebas exhaustivas de todas las combinaciones válidas para una y dos capas. Para modelos de tres capas se probó una muestra aleatoria de 52 modelos. El resumen de arquitecturas base de punto flotante y número de variantes probadas se muestra en la tabla 17.

Tabla 17. Modelos base de punto flotante para las pruebas con CIFAR10. (Por ASW)

Capas ocultas	Descripción	Unidades / capa			Variantes probadas
		1	2	3	
1	(F-256)	256			65
2	(F-256,256)	128	64		224
3	(F-256,256,256)	128	64	32	52

Aunque 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 para este siempre y cuando se haga un preprocesamiento más extenso de la entrada, incluyendo aumento de las características y recorte de la imagen [94]. Sin embargo, se presenta este caso sin utilizar preprocesamiento para mostrar las bondades de la arquitectura propuesta.

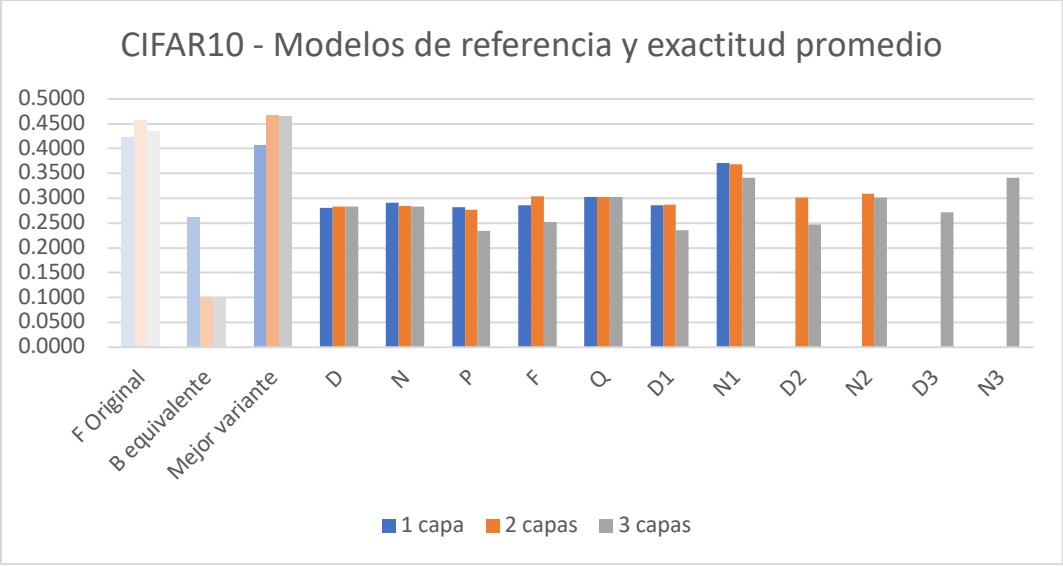
En esta serie de pruebas, la tabla 18 muestra que las capas de normalización N1 y N2 proporcionan mayor contribución al resultado, seguidas de los atajos F y Q.

En la gráfica 4 se aprecia también que la exactitud alcanzada por estos modelos no es consecuencia de un solo componente, pues en promedio ninguno alcanza la exactitud de la mejor variante.

Tabla 18. Resultados de exactitud promedio consolidada con CIFAR10. (Por ASW)

Modelo / Capa	1 capa	2 capas	3 capas
F Original	0.4224	0.4567	0.4350
B equivalente	0.2612	0.1000	0.1000
Mejor variante	0.4061	0.4679	0.4661
D	0.2803	0.2831	0.2835
N	0.2907	0.2850	0.2836
P	0.2824	0.2762	0.2340
F	0.2861	0.3040	0.2530
Q	0.3025	0.3025	0.3025
D1	0.2857	0.2875	0.2362
N1	0.3708	0.3677	0.3407
D2		0.3008	0.2478
N2		0.3096	0.3016
D3			0.2712
N3			0.3418

A pesar de la dificultad el conjunto CIFAR10 para el perceptrón multicapa, este ejemplo nos muestra como la contribución de los diferentes elementos ayuda a mejorar sustancialmente el desempeño de un perceptrón binario.



Gráfica 4. Exactitud promedio consolidada para modelos sobre CIFAR10. (Por ASW)

La tabla 19 permite apreciar que los atajos de punto flotante incrementan los requerimientos de memoria hasta casi el tamaño del perceptrón original. También vemos que las mejores variantes se desempeñan bien con un tamaño de 0.15 del modelo base.

Tabla 19. CIFAR10. Comparación entre tamaño y exactitud para 1 a 3 capas. (Por ASW

Ocultas	Descripción	Modelo	Exactitud	Tamaño	Tam. rel
1	Flotante	F-256	0.4224	3083.04	1.00
	Atajo P	B-D256N,P	0.3922	451.04	0.15
	Atajo Q	B-D256N,Q	0.3617	878.04	0.28
	Atajo F	B-D256N,F	0.3670	3184.04	1.03
	Mejor	B-D256N,P,D	0.4061	451.04	0.15
	Binario	B-256	0.2612	107.04	0.03
2	Flotante	F-256,256	0.4567	3340.04	1.00
	Atajo P	B-D256N,D256N,P	0.4679	462.04	0.14
	Atajo Q	B-D256N,D256N,Q	0.4091	889.04	0.27
	Atajo F	B-D256N,D256N,F	0.4477	3195.04	0.96
	Mejor	B-D256N,D256N,P	0.4679	462.04	0.14
	Binario	B-256,256	0.1	116.04	0.03
3	Flotante	F-256,256,256	0.4350	3597.04	1.00
	Atajo P	B-D256N,D256N,D256N,P	0.4423	473.04	0.13
	Atajo Q	B-D256N,D256N,D256N,Q	0.4049	900.04	0.25
	Atajo F	B-D256N,D256N,D256N,F	0.4314	3204.04	0.89
	Mejor	B-D256N,D256,D256N,PN,D	0.4661	475.04	0.13
	Binario	B-256,256,256	0.1000	125.04	0.03

6. CONCLUSIONES

6.1. CONCLUSIONES Y DISCUSIÓN

Dentro de la búsqueda por mejorar la exactitud de los modelos de redes neuronales binarias, la búsqueda de arquitecturas específicas binarias juega un papel relevante al proponer modelos que mejoran la exactitud de las redes binarizadas mediante la implementación de estrategias basadas en cambios a la estructura original. La ilustración 6 representa la disposición original del perceptrón multicapa binario.

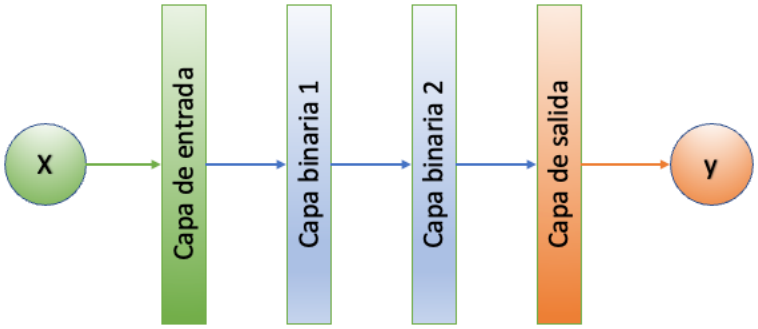


Ilustración 6. Perceptrón multicapa binario de dos capas ocultas. (Por ASW)

Para cumplir con los objetivos propuestos, se ha introducido un bloque de construcción de perceptrón multicapa binario que reemplaza cada capa totalmente conectada del perceptrón de punto flotante original mediante un grupo, el cual se muestra en la ilustración 7 y consiste en una capa de abandono antes de la capa totalmente conectada, después de la cual se utiliza una capa de normalización por lotes.

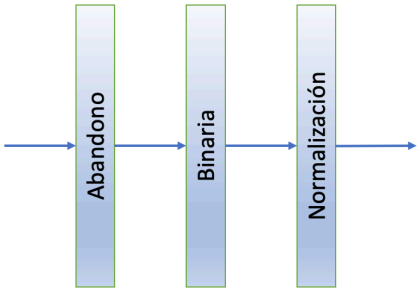


Ilustración 7. Bloque de construcción para perceptrón multicapa binario residual. (Por ASW)

Como se aprecia en la ilustración 8, el diseño se completa mediante una conexión residual que toma la información de la capa de entrada y la suma con la salida de la última capa oculta, antes de entrar a la capa de salida.

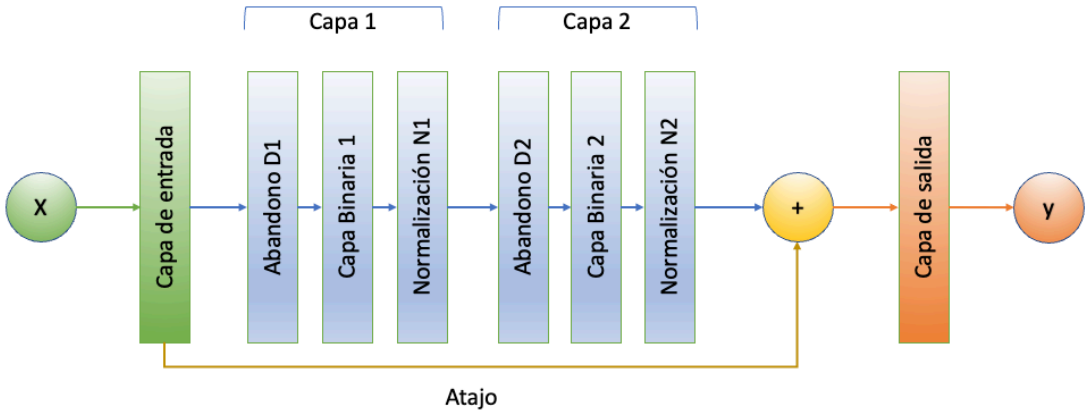


Ilustración 8. Perceptrón multicapa binario residual de dos capas binarias. (Por ASW)

Para conocer si la exactitud obtenida compensa la pérdida de información derivada de la binarización, se procedió con un enfoque experimental en el que se probó el modelo propuesto con todas sus combinaciones válidas basadas en la estructura del bloque de construcción mediante un generador de perceptrones que construye cada variante posible recopilando los resultados del entrenamiento y la evaluación.

Se desarrollaron tres métodos diferentes de implementar los atajos, los cuales fueron incluidos en el generador de modelos para conocer cuál de estos proporcionaba mejores resultados, determinándose que aunque las tres implementaciones contribuyen a mejorar la precisión, los atajos de punto flotante incrementan mucho el tamaño de la variante binaria, llegando en ocasiones a ser mayor que el modelo de punto flotante original, resultando en una arquitectura que no es viable para dispositivos de recursos limitados.

Se realizó un análisis sobre el tamaño de los modelos, llegando a la conclusión de que este puede ser determinado a priori, dado que está determinado por el ancho de las capas y el número de bits utilizado para la representación de cada una de estas como se muestra en la ilustración 9, por lo que no es necesario entrenar un modelo que no se ajuste a las necesidades de memoria y almacenamiento del entorno de implementación.

La ecuación (20) reproduce la ecuación final (19) de la sección “Tamaño del modelo propuesto”, que nos permite calcular el tamaño de un perceptrón multicapa binario residual.

$$S = \sum_{t=1}^{T} (k|V_{t-1}||V_t| + m|V_t|) + m(|V_T| + 1)|V_{t-1}| + S_A + \sum_{t=1}^{T} 2m|V_t|$$

(20)

La tabla 20, consolida los valores obtenidos para el tamaño relativo de cada tipo de modelo de acuerdo con la conexión residual utilizada, y se percibe en la columna B que un perceptrón binario sin modificaciones tiene un tamaño promedio de 0.04 de los modelos base de punto flotante. Para las variantes residuales, los diseños con atajos basados en capa de agrupamiento ocupan en promedio 0.12 del tamaño original y en el extremo opuesto los que utilizan conexiones residuales de punto flotante no muestran mucho ahorro, encontrándose un promedio de 0.78 del tamaño del perceptrón base, que representa más de 3.5 veces el tamaño de las combinaciones con atajos cuantizados.

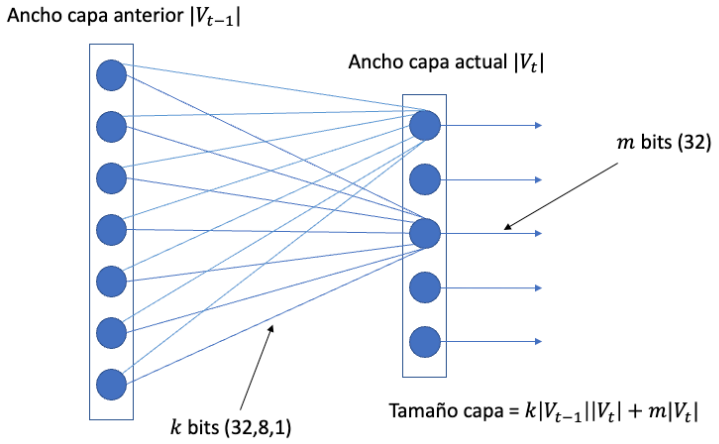


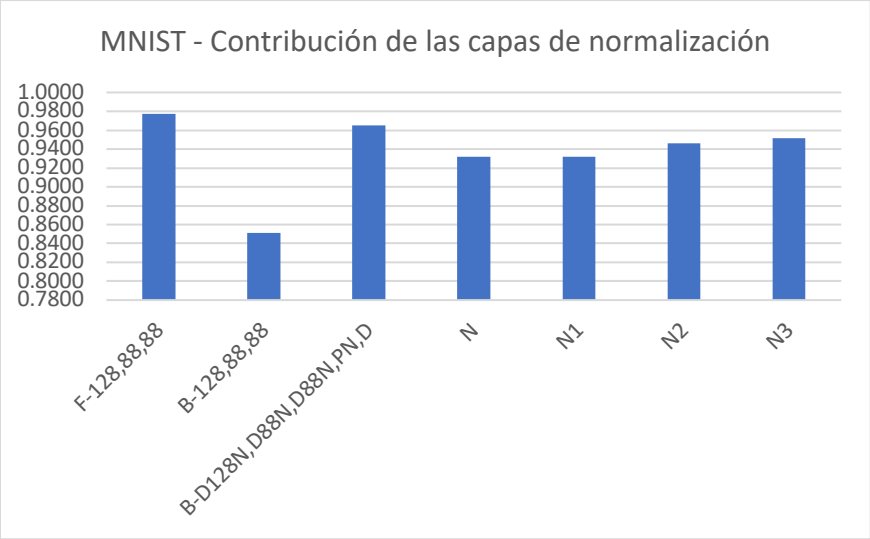
Ilustración 9. Cálculo del tamaño de la capa V. (Por ASW)

El desarrollo experimental de la arquitectura de perceptrón multicapa binario residual que se plantea en este estudio, demuestra que los modelos que siguen el bloque de construcción propuesto tienen mejor desempeño que los diseños binarios equivalentes, y algunas variantes pueden incluso superar la exactitud de los de punto flotante. Esto se debe a que las capas de normalización ayudan a mejorar la exactitud de sus equivalentes binarios, la gráfica 5 ilustra la contribución de las capas de normalización N1, N2 y N3 en el modelo, y las conexiones de atajo también contribuyen al resultado como se observa en la gráfica 6.

Tabla 20. Tamaño relativo del perceptrón binario de acuerdo con el tipo de atajo. (Por ASW)

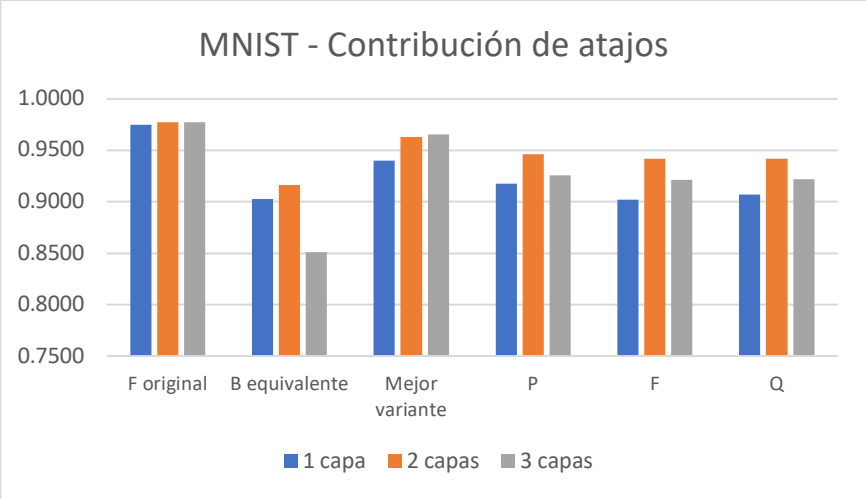
Tamaño relativo promedio				
Capas	B	P	Q	F
1	0.04	0.15	0.29	1.03
2	0.03	0.12	0.23	0.79
3	0.03	0.09	0.16	0.52
Promedio	0.04	0.12	0.22	0.78

Los resultados obtenidos a través de la experimentación sugieren que la inclusión de capas de normalización y conexiones residuales en un perceptrón multicapa binario, conforma una buena estrategia para mejorar su exactitud. Además, un atajo implementado mediante la combinación de una capa de agrupamiento máximo seguida de una capa totalmente conectada de punto flotante es la mejor opción para los modelos que utilizan valores reales como entrada, como en el caso de imágenes, mientras que los atajos cuantizados son mejores para los conjuntos de datos de texto con codificación uno a uno.



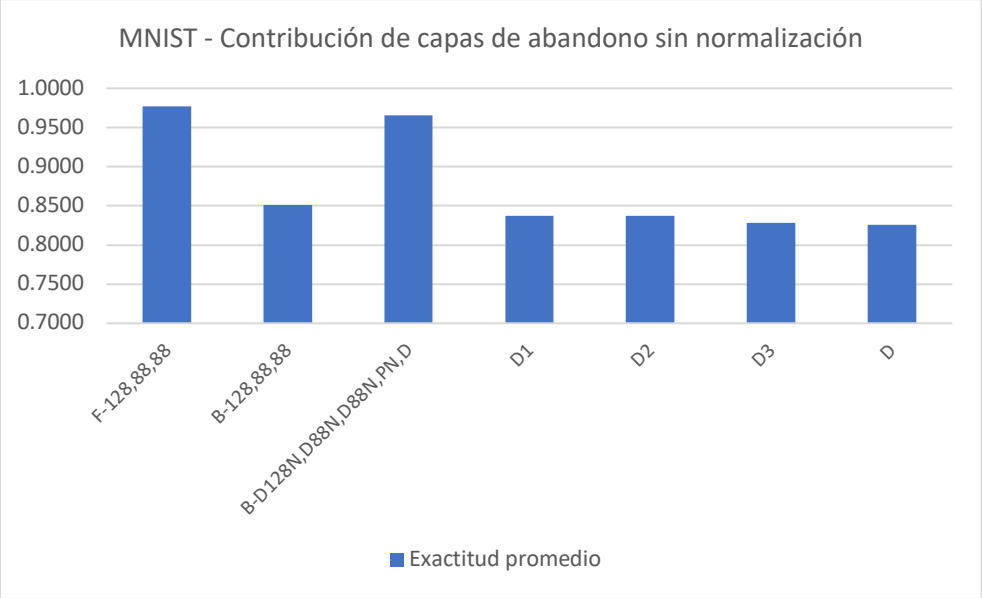
Gráfica 5. Contribución de las capas de normalización N a N3 al resultado. (Por ASW)

Los resultados también indican que el uso de capas de abandono cuando no están en combinación con las de normalización la exactitud no mejora como se muestra en la gráfica 7, en la que se aprecia que la exactitud es menor al perceptrón binario equivalente si únicamente se usan capas de abandono, lo cual sugiere que las capas de normalización son las que más contribuyen a la mejora de la exactitud en el modelo propuesto.



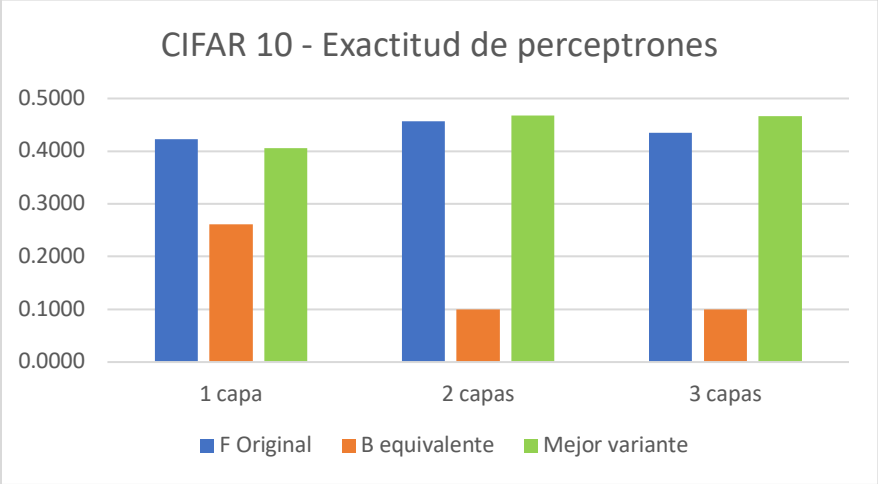
Gráfica 6. Contribución del uso de atajos P, F y Q al resultado. (Por ASW)

La gráfica 8 muestra como el desempeño de la mejor variante del perceptrón desarrollado supera al perceptrón de punto flotante original en modelos de dos y tres capas, y en todos los casos es mejor que el perceptrón binario equivalente.



Gráfica 7. Contribución de las capas de abandono sin normalización. (Por ASW)

Por lo tanto, se recomienda que esta técnica se implemente en futuros desarrollos de sistemas de aprendizaje automático enfocados a dispositivos de recursos restringidos basados en perceptrones multicapa binarios



Gráfica 8. Exactitud del modelo original, el equivalente y el propuesto. (Por ASW)

6.2. APORTACIONES

La principal aportación de este trabajo es mostrar que las redes neuronales binarias representan un área de oportunidad para todos los interesados en implementar modelos de aprendizaje profundo en escenarios de recursos computacionales limitados.

En segundo lugar, se propone un modelo residual de perceptrón multicapa binario, y se muestra mediante la experimentación y el análisis de los resultados que estas modificaciones contribuyen a mejorar la exactitud, derivándose las siguientes recomendaciones:

- Las capas de normalización siempre ayudan a mejorar la exactitud del modelo, las pruebas muestran que la capa de normalización enseguida de la última capa binaria es la que tiene mayor aportación.
- Las capas de abandono por sí mismas no producen una mejora notable, por lo que se recomienda utilizarlas solo en combinación con las capas de normalización.
- El uso de conexiones residuales mejora la exactitud de la red con impacto moderado en al tamaño de esta, el cual se puede calcular a priori.
- El tamaño de un modelo binario está predeterminado por el ancho de sus capas, su tipo y precisión, por lo que puede diseñarse el modelo considerando la cantidad de memoria en que se desea implementar.

Finalmente, se muestra la implementación de conexiones de atajo entre bloques de un modelo de aprendizaje profundo utilizando técnicas diferentes a las documentadas en la literatura sobre redes neuronales residuales.

6.3. Trabajo futuro

Las redes neuronales binarias tienen un gran potencial de aplicación cuando se logra reducir la pérdida de exactitud derivada de la binarización, y aunque el perceptrón multicapa es un modelo flexible, existen aplicaciones para las cuales se requieren modelos más complejos que pueden beneficiarse del enfoque binario para su implementación en ambientes de recursos limitados. Esta investigación que hoy se concluye se utilizará como base para profundizar en el campo de diseño de otros modelos binarios de aprendizaje profundo, ahondando en técnicas que permitan la sustitución de las capas de normalización por otras que requieran menor número cálculos y el uso de optimizadores binarios, de manera que reduzcan más las operaciones de punto flotante requeridas tanto para entrenar los modelos como realizar la inferencia, mejorando el desempeño en ambientes restringidos.

6.4. Publicaciones Derivadas

Solís Winkler Agustín, Tapia Fabela José Luis, Osnaya Baltierra Santiago. Diseño empírico de una arquitectura de perceptrón multicapa binario residual. Aceptado como poster para exponerse en COMIA 2023 y ser publicado en número especial de la revista Research in Computing Science del IPN.

REFERENCIAS

[1] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: from theory to algorithms*, First. New York: Cambridge University Press, 2014. [Online]. Available: http://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning

[2] N. K. Kasabov, *Foundations of neural networks, fuzzy systems, and knowledge engineering*, Second Printing. London: MIT Press, 1996.

[3] C. Aggarwal, *Artificial Intelligence. A textbook*, 1st ed. Swtzerland: Springer Nature Switzerland AG, 2021.

[4] T. Munkata, *Fundamentals of the New Artificial Intelligence*, 2nd ed. London: Springer-Verlag, 2008.

[5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT Press, 2016.

[6] F. Rosenblatt, "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain," *Psychological Review*, vol. 65, no. 6, pp. 19–27, 1958.

[7] F. Chollet, *Deep Learning with Python*. Manning Publications Co., 2018.

[8] A. Alqahtani, X. Xie, and M. W. Jones, "Literature review of deep network compression," *Informatics*, vol. 8, no. 4, Dec. 2021, doi: 10.3390/informatics8040077.

[9] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "A survey of model compression and acceleration for deep neural networks," Oct. 2017, [Online]. Available: http://arxiv.org/abs/1710.09282

[10] S. Pokhrel, "4 popular model compression techniques explained," *Xailient*, Jan. 19, 2022. https://xailient.com/blog/4-popular-model-compression-techniques-explained/ (accessed Mar. 01, 2023).

[11] R. Mishra, H. P. Gupta, and T. Dutta, "A Survey on Deep Neural Network Compression: Challenges, Overview, and Solutions," Oct. 2020, [Online]. Available: http://arxiv.org/abs/2010.03954

[12] K. Hao, "Training a single AI model can emit as much carbon as five cars in their lifetimes," *MIT technology Review*, Jun. 06, 2019. https://www.technologyreview.com/2019/06/06/239031/training-a-single-ai-model-can-emit-as-much-carbon-as-five-cars-in-their-lifetimes/ (accessed Mar. 01, 2023).

[13] P. Ganesh, “Deep Learning — Model Optimization and Compression: Simplified,” *Towards Data Science*, Jun. 24, 2019. https://towardsdatascience.com/machine-learning-models-compression-and-quantization-simplified-a302ddf326f2 (accessed Mar. 01, 2023).

[14] J. O’Neill, “A Survey of Neural Network Compression,” Jun. 2020, [Online]. Available: http://arxiv.org/abs/2006.03669

[15] Wikipedia - Embedded System, “Embedded system,” *Wikipedia*, Feb. 15, 2023. https://en.wikipedia.org/wiki/Embedded_system (accessed Mar. 01, 2023).

[16] Wikipedia - Edge Computing, “Edge computing,” *Wikipedia*, Feb. 09, 2023. https://en.wikipedia.org/wiki/Edge_computing (accessed Feb. 11, 2023).

[17] Y. LeCun, J. S. Denker, and S. A. Solla, “Optimal Brain Damage,” in *Advances in neural information processing systems*, D.S. Touretzky, 1990, pp. 598–605.

[18] S. Han, J. Pool, J. Tran, and W. J. Dally, “Learning both Weights and Connections for Efficient Neural Networks,” 2015.

[19] P. E. Novac, G. B. Hacene, A. Pegatoquet, B. Miramond, and V. Gripon, “Quantization and deployment of deep neural networks on microcontrollers,” *Sensors*, vol. 21, no. 9, May 2021, doi: 10.3390/s21092984.

[20] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, “Binarized neural networks: training deep neural networks with weights and activations constrained to +1 or -1,” Feb. 2016, [Online]. Available: http://arxiv.org/abs/1602.02830

[21] R. Abbasi-Asl and B. Yu, “Structural Compression of Convolutional Neural Networks with Applications in Interpretability,” *Front Big Data*, vol. 4, p. 73, Aug. 2021, doi: 10.3389/FDATA.2021.704182/BIBTEX.

[22] G. C. Marinó, A. Petrini, D. Malchiodi, and M. Frasca, “Deep neural networks compression: A comparative survey and choice recommendations,” *Neurocomputing*, vol. 520, pp. 152–170, Feb. 2023, doi: 10.1016/J.NEUCOM.2022.11.072.

[23] C. Aggarwal, *Neural Networks and Deep Learning. A textbook*. Springer International Publishing AG, 2018.

[24] H. Qin, R. Gong, X. Liu, X. Bai, J. Song, and N. Sebe, “Binary neural networks: a survey,” Mar. 2020, doi: 10.1016/j.patcog.2020.107281.

[25] C. Yuan and S. S. Agaian, “A comprehensive review of binary neural network,” Feb. 2022, [Online]. Available: http://arxiv.org/abs/2110.06804

[26] T. Simons and D. J. Lee, “A review of binarized neural networks,” *Electronics (Switzerland)*, vol. 8, no. 6. MDPI AG, Jun. 01, 2019. doi: 10.3390/electronics8060661.

[27] J. Bethge, H. Yang, M. Bornstein, and C. Meinel, “Back to simplicity: how to train accurate BNNs from scratch?,” Jun. 2019, [Online]. Available: http://arxiv.org/abs/1906.08637

[28] M. Courbariaux, Y. Bengio, and J.-P. David, “BinaryConnect: training deep neural Networks with binary weights during propagations,” Nov. 2015.

[29] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, “XNOR-Net: ImageNet classification using binary convolutional networks,” 2016.

[30] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, “DoReFa-Net: training low bitwidth convolutional neural networks with low bitwidth gradients,” Jun. 2016, [Online]. Available: http://arxiv.org/abs/1606.06160

[31] X. Lin, C. Zhao, and W. Pan, “Towards Accurate Binary Convolutional Neural Network,” Nov. 2017.

[32] Z. Liu, B. Wu, W. Luo, X. Yang, W. Liu, and K.-T. Cheng, “Bi-real net: enhancing the performance of 1-bit CNNs with improved representational capability and advanced training algorithm.,” in *Proceedings of the European conference on computer vision (ECCV)*, 2018.

[33] J. Bethge, H. Yang, M. Bornstein, and C. Meinel, “BinaryDenseNet: developing an architecture for binary neural networks,” 2019. [Online]. Available: https://github.com/hpi-xnor/BMXNet-v2

[34] H. Phan, Y. He, M. Savvides, and Z. Shen, “Mobinet: A mobile binary network for image classification,” in *Proceedings of the IEEE/CVF Winter Conference onApplications of Computer Vision*, 2020.

[35] Z. Liu, Z. Shen, M. Sarvides, and K.-T. Cheng, “Reactnnet. Towards precise binary neural network with generalized activation functions.,” in *European Conference on Computer Vision*, 2020.

[36] A. J. Redfern, L. Zhu, and M. K. Newquist, “BCNN: A binary CNN with all matrix ops quantized to 1 bit precision,” 2021.

[37] M. T. Hagan, H. B. Demuth, M. H. Beale, and O. De Jesus, *Neural network design*, 2nd ed. 1996.

[38] G. De Luca, “Neural Network Architecture: Criteria for Choosing the Number and Size of Hidden Layers,” *Baeldung on Computer Science*, Nov. 11, 2022. https://www.baeldung.com/cs/neural-networks-hidden-layers-criteria (accessed Feb. 19, 2023).

[39] J. Heaton, *Introduction to Neural Networks with Java*, Second. St. Louis: Heaton Research, 2008.

[40] B. Coppin, *Artificial Intelligence Illuminated*. Jones and Bartlett Publishers, Inc, 2004.

[41] Wikipedia - Neuron, “Neuron,” *Wikipedia*, Feb. 03, 2023. https://en.wikipedia.org/wiki/Neuron (accessed Apr. 24, 2022).

[42] S. J. Russell and P. Norvig, *Artificial Intelligence A Modern Approach Third Edition*. Upper Saddle River, New Jersey, 2010.

[43] D. A. González-Moreno, *Introducción a la Teoría de las Gráficas*. UAM, 2017.

[44] Wikipedia - Graph Theory, “Graph Theory,” *Wikipedia*, Sep. 2022. https://en.wikipedia.org/wiki/Graph_theory (accessed Sep. 14, 2022).

[45] T. E. Bettilyon, “Deep Neural Networks As Computational Graphs,” *Ted´s Lab*, 2018. https://medium.com/tebs-lab/deep-neural-networks-as-computational-graphs-867fcaa56c9 (accessed Sep. 14, 2022).

[46] J. Heaton, *Artificial Intelligence for Humans. Volume 3: Deep Learning and Neural Networks*, 1.0. Heaton Research, Inc., 2015.

[47] E. Call, “Faster Convolutional Networks,” Master of Science in Artificial Intelligence, Radboud University, Nijmegen, 2017.

[48] Wikipedia - Deep Learning, “Deep Learning,” *Wikipedia*, Feb. 05, 2023. https://en.wikipedia.org/wiki/Deep_learning#Deep_neural_networks (accessed Apr. 24, 2022).

[49] Wikipedia - GPT3, “GPT-3,” *Wikipedia*, Dec. 15, 2022. https://en.wikipedia.org/wiki/GPT-3 (accessed Dec. 14, 2022).

[50] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, “Quantized neural networks: training neural networks with low precision weights and activations,” 2018.

[51] R. R. Nakod, “Model Compression Techniques for Edge AI,” *Embedded Computing Design*, Sep. 06, 2022. https://embeddedcomputing.com/technology/software-and-os/simulation-modeling-tools/model-compression-techniques-for-edge-ai (accessed Mar. 14, 2023).

[52] S. Arora, R. Ge, B. Neyshabur, and Y. Zhang, “Stronger generalization bounds for deep nets via a compression approach,” in *Proceedings of the International Conference on Machine Learning* , Stockholm, Sweden, 2018, pp. 254–263.

[53] C. Bucili, R. Caruana, and A. Niculescu-Mizil, “Model compression,” in *ACM KDD Conference*, 2006, pp. 535–541.

[54] J. Ba and R. Caruana, “Do deep nets really need to be deep?,” in *NIPS Conference*, 2014, pp. 2564–2662.

[55] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, “A Survey of Quantization Methods for Efficient Neural Network Inference,” Mar. 2021, [Online]. Available: http://arxiv.org/abs/2103.13630

[56] M. Nagel, M. Fournarakis, R. A. Amjad, Y. Bondarenko, M. van Baalen, and T. Blankevoort, “A White Paper on Neural Network Quantization,” Jun. 2021.

[57] C. Yuan and S. S. Agaian, “A comprehensive review of binary neural network,” *Artificial Intelligence Review 2023*, pp. 1–65, Oct. 2021, doi: 10.1007/S10462-023-10464-W/METRICS.

[58] G. Hinton and T. Teleman, “Divide the gradient by a running average of its recent magnitude,” *Coursera: neural networks for machine learning*, 2012.

[59] M. Abadi *et al.*, “Tensorflow: A system for large-scale machine learning,” in *12th symposium on operating systems design and implementation*, 2016, p. 265.

[60] A. Paszke *et al.*, “PyTorch: An Imperative Style, High-Performance Deep Learning Library,” Dec. 2019.

[61] Google, “Keras: the Python deep learning API,” *keras.io*, 2023. https://keras.io/ (accessed Feb. 19, 2023).

[62] L. Geiger and team Plumerai, “Larq: An Open-Source Library for Training Binarized Neural Networks,” *J Open Source Softw*, vol. 5, no. 45, p. 1746, Jan. 2020, doi: 10.21105/joss.01746.

[63] T. Bannink *et al.*, “Larq Compute Engine: Design, Benchmark, and Deploy State-of-the-Art Binarized Neural Networks,” 2021.

[64] larq.dev, “Getting Started - Larq,” *Larq*. https://docs.larq.dev/larq/ (accessed May 14, 2023).

[65] W. Tang, G. Hua, and L. Wang, “How to train a compact binary neural network with high accuracy,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2017.

[66] B. Zhuang, C. Shen, M. Tan, L. Liu, and I. Reid, “Structured Binary Neural Networks for Accurate Image Classification and Semantic Segmentation,” Nov. 2018, [Online]. Available: http://arxiv.org/abs/1811.10413

[67] S. Darabi, M. Belbahri, M. Courbariaux, and V. P. Nia, “Regularized Binary Network Training,” Dec. 2018, [Online]. Available: http://arxiv.org/abs/1812.11800

[68] K. Helwegen, J. Widdicombe, L. Geiger, Z. Liu, K.-T. Cheng, and R. Nusselder, “Latent Weights Do Not Exist: Rethinking Binarized Neural Network Optimization,” Jun. 2019.

[69] H. Kim, J. Park, C. Lee, and J.-J. Kim, “Improving Accuracy of Binary Neural Networks using Unbalanced Activation Distribution,” Dec. 2020.

[70] T. Chen, Z. Zhang, X. Ouyang, Z. Liu, Z. Shen, and Z. Wang, “Training binary neural networks without batch normalization,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

[71] Z. Tu, C. Xinghao, R. Pengju, and W. Yunhe, “AdaBin: Improving Binary Neural Networks with adaptive Binary Sets,” 2021.

[72] Under double blind review, “BoolNet: Streamlining bynary neural networks using binary feature maps,” *ICLR*, 2022.

[73] X. Shi *et al.*, “RepBNN: towards a precise Binary Neural Network with Enhanced Feature Map via Repeating,” Jul. 2022.

[74] Z. Xu *et al.*, “ReCU: Reviving the Dead Weights in Binary Neural Networks,” Mar. 2021.

[75] B. Martinez, J. Yang, A. Bulat, and G. Tzimiropoulos, “Training Binary Neural Networks with Real-to-Binary Convolutions,” Mar. 2020.

[76] C. Liu *et al.*, “Circulant binary convolutional networks: Enhancing the performance of 1-bit dcnns with circular back propagation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Regognition*, 2019.

[77] A. Mishra, E. Nurvitadhi, J. J. Cook, and D. Marr, “WRPN: Wide Reduced-Precision Networks,” Sep. 2017.

[78] M. Shen, X. Liu, R. Gong, and K. Han, “Balanced binary neural networks with gated residual,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

[79] A. Bulat, B. Martinez, and G. Tzimiropoulos, “BATS: Binary ArchitecTure Search,” Mar. 2020.

[80] S. Zhu, X. Doug, and H. Su, “Binary ensemble neural networks: More bits per network or more networks per bit?,” in *Proceedi gs of the IEEE/CVF Conference in Computer Vision and Pattern Recognition*, 2019.

[81] M. Alizadeh, J. Fernández-Marqués, N. D. Lane, and Y. Gal, “An empirical study of binary neural networks’ optimization.,” in *International Conference on Learning Representations*, 2018.

[82] A. G. Anderson and C. P. Berg, “The high-dimensional geometry of binary neural net-works.,” in *International Conference on Learning Representations*, 2018.

[83] H. Li, S. De, Z. Xu, C. Studer, H. Samet, and T. Goldstein, “Training quantized nets: A deeper understanding.,” *Adv Neural Inf Process Syst*, pp. 5811–5821, 2017.

[84] K. He, X. Zhang, S. Ren, and jiang Sun, “Deep Residual Learning for Image Recognition,” Dec. 2015.

[85] C. Szegedy *et al.*, “Going deeper with convolutions,” in *Cvpr*, 2015.

[86] J. Korstanje, “The F1 score,” *Towards Data Science*, Aug. 31, 2021. https://towardsdatascience.com/the-f1-score-bec2bbc38aa6 (accessed Jul. 07, 2023).

[87] Y. LeCun, C. Cortes, and C. Burges, “MNIST handwritten digit database.,” *yann.lecun.com*. http://yann.lecun.com/exdb/mnist/ (accessed Mar. 26, 2023).

[88] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998, doi: 10.1109/5.726791.

[89] A. Krizhevsky, “CIFAR-10 and CIFAR-100 datasets.” https://www.cs.toronto.edu/~kriz/cifar.html (accessed Mar. 26, 2023).

[90] A. Maas, “Sentiment Analysis,” *Stanford AI Labs*. https://ai.stanford.edu/~amaas/data/sentiment/ (accessed Mar. 27, 2023).

[91] A. L. Maas, R. E. Daly, P. T. Pham, dan Huang, A. Y. Ng, and C. Potts, “Learning Word Vectors for Sentiment Analysis,” in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, Association fro Computational Linguistics, Jun. 2011, pp. 142–150. Accessed: Mar. 26, 2023. [Online]. Available: https://ai.stanford.edu/~amaas/data/sentiment/

[92] D. D. Lewis, “Reuters-21578 Text Categorization Collection Data Set,” *UCI Machine Learning Repository*, Sep. 26, 1997. https://archive.ics.uci.edu/ml/datasets/reuters-21578+text+categorization+collection (accessed Mar. 26, 2023).

[93] Keras.io, “Reuters newswire classification dataset,” *Keras.io*. https://keras.io/api/datasets/reuters/ (accessed Mar. 26, 2023).

[94] C. Parvathi, “Classification of Cifar-10 DATASET using a Multi-Layer Perceptron model and a Convolutional Neural Network model.,” *GitHub*, 2019. https://github.com/parvathic/ml-cifar10 (accessed Apr. 19, 2023).