Instituto Politécnico Nacional

Centro de Investigación en Computación

Deep Learning

Prof.: Dr. Ricardo Menchaca Méndez

CNNs vs MLPs en Clasificación de Imágenes*

Alan Rodrigo López López
alopez1820@alumno.ipn.mx
alanrodrigolopez1017@gmail.com
https://github.com/AlanLopez1017/Project-MLPvsCNN
03 Noviembre 2025

Abstract

This work presents a comparison of the performance, complexity, and efficiency of two deep learning architectures: the Multilayer Perceptron (MLP) and Convolutional Neural Networks (CNN), applied to the task of image classification using the CIFAR-10 dataset. Both architectures were implemented from scratch using PyTorch, developing two main configurations for each model: a base version and another with approximately 820,000 trainable parameters, with and without dropout to evaluate the impact of regularization. The models were trained for 50 epochs under controlled conditions using the Adam optimizer and data augmentation techniques. Experimental results show that CNNs significantly outperform MLPs across all performance metrics, achieving an average accuracy of 82%, compared to 49% for the MLP. In both experiments — those differing in the number of parameters and those with architectures matched in parameter count — CNNs consistently achieved superior results. From a computational perspective, CNNs involve a higher training and inference cost, yet they converged faster due to their ability to learn hierarchical representations of spatial features.

Key Words: Multilayer Perceptron, Convolutional Neural Network, Image Classification, CIFAR-10, Data Augmentation

1. Introducción y motivación

Los métodos de clasificación de imágenes desempeñan un papel importante en una amplia variedad de campos y aplicaciones. La clasificación de imágenes es el proceso de extraer clases a partir de un mapa de bits. En este trabajo se presenta una comparación en el rendimiento realizado por dos tipos de redes neuronales: perceptrón multicapa (MLP) y redes convolucionaes (CNN) al realizar la clasificación de imágenes.

El Perceptrón Multicapa (MLP) es el tipo más popular de redes neuronales artificiales. Es una clase de red neuronal de propagación hacia adelante. Suele constar de una capa de entrada, varias capas ocultas y una capa de salida. Cada nodo en un MLP es una neurona con una función de activación. La arquitectura basada en este perceptrón multicapa fue utilizada comúnmente en visión artificial, pero ha sido reemplazada por las redes neuronales convolucionales. [1] [2]

Las redes neuronales convolucionales han demostrado ser prometedoras en aplicaciones de reconocimientos de patrones. Han sido utilizadas obteniendo un gran éxito en diferentes campos, reduciendo la necesidad de extracción manual de características y logrando un rendimiento de vanguardia en campos como reconocimiento de voz y visión por computadora. [3] [4]

Llamó la atención cuando las redes neuronales convolucionales a gran escala para la clasificación de imágenes superó a todas las demás técnicas en la competición ImageNet 2012 [5]. Con ello, varios investigadores desarrollaron diversos modelos CNN que compitieron con humanos en el reconocimiento de imágenes. [4] [6] [7]
La principal ventaja de las CNN radica en su capacidad para extraer de forma fiable y automática características relevantes sustituyendo el uso de técnicas de extracción manual a partir de conjunto de datos de alta dimensionalidad que son susceptibles a errores humanos.

Las CNN constan de 3 tipos principales de capas: convolución, pooling y totalmente conectadas, siendo esta última la de perceptrón multicapa (MLP). Tanto las convolucionales como las de pooling son dos tipos de capas ocultas, análogas a las capas ocultas en MLP.

La operación de convolución permite reducir simultáneamente la cantidad de información almacenada en memoria, lo que mejora su manejo con imágenes de alta resolución, y resaltar las características de referencia de la imagen, como bordes, contornos o contornos.
En la siguiente etapa de procesamiento, a partir de estos bordes y contornos, se pueden reconocer fragmentos repetibles de texturas, que luego se pueden integrar en fragmentos de la

imagen [8] [9].

La capa de pooling se dedica a reducir el tamaño de la resolución espacial en el mapa de características, lo que conlleva una disminución drástica en el número de parámetros, reduciendo así el riesgo de sobreajuste del modelo y por consiguiente, una mayor generalización. Reduce la dimensionalidad conservando la información más importante. Otra ventaja importante de esta capa es que hace que la CNN sea invariante a pequeñas transformaciones, distorsiones y traslaciones.

Se puede añadir tantas capas de convolución y pooling como se desee siempre que se consiga un alto rendimiento. Las capas de pooling no tiene por qué seguir a cada capa de convolución.

La principal diferencia entre MLP y CNN es que en el MLP cada elemento de entrada está conectado a cada neurona de la capa oculta, es por ello que se le denomina como red totalmente conectada, mientras que en la CNN solamente una vecindad predeterminada (número limitado de elementos de entrada) denominado campo receptivo, está conectado a solo una parte de la capa oculta.

En el proceso de aprendizaje se toma en cuenta la preparación apropiada de los datos, así como los parámetros óptimos para que el modelo se entrene, incrementando el impacto de los datos de entrenamiento [10] [11].

El número de imágenes en el conjunto de datos, su tamaño y el número de imágenes en cada una de las categorías afectan la eficiencia del entrenamiento. Una selección incorrecta de parámetros puede provocar sobreajuste o subajuste [12].

A diferencia de las MLP's, las arquitecturas de CNN son profundas y requieren operaciones computacionalmente costosas, implicando que tendrán un tiempo de entrenamiento más prolongado así como una alta potencia de cómputo. Este trabajo pretende entender porqué a pesar de estas desventajas de las CNN, siguen siendo la primera y mejor opción para este tipo de tareas como la clasificación de imágenes, teniendo en mente de que logran un mejor rendimiento tanto en menos épocas como con menos parámetros.

1.1. Objetivos

1. Implementar arquitecturas CNN y MLP utilizando PyTorch.
2. Comparar los modelos según tres métricas clave: **precisión/exactitud**, **número de parámetros** y **tiempo de entrenamiento**.
3. Comprender por qué se prefieren las CNN para datos de imagen, a pesar de que las MLP son aproximadores de funciones universales.

2. Metodología y detalles de la arquitectura

2.1. Dataset

El conjunto de datos a utilizar para los experimentos es el CIFAR-10 que consta de 60,000 imágenes a color de 32x32 píxeles, divididas en 10 clases (6000 imágenes por clase). Desde la página oficial los datos ya se encuentran divididos en 50,000 imágenes para entrenamiento y 10,000 para pruebas.

El conjunto de datos se divide en cinco lotes de entrenamiento y uno de prueba, cada uno con 10,000 imágenes. El lote de prueba contiene 1000 imágenes seleccionadas aleatoriamente de cada clase. En total, los lotes de entrenamiento contiene 5000 imágenes de cada clase. Las 10 clases de este dataset son: avión, automóvil, pájaro, ciervo, perro, rana, caballo, barco y camión. En la Figura 1 se muestran dichas clases del conjunto de datos junto con 10 imágenes aleatorias de cada una.

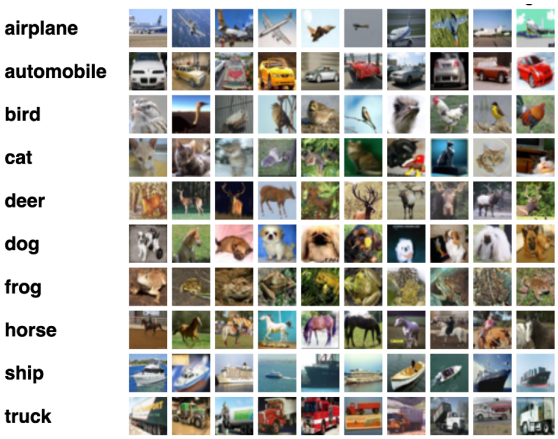

Fig. 1.: Clases del conjunto de datos con 10 imágenes por cada una

Para los experimentos se creó una partición de los datos de entrenamiento para crear los datos de validación. Por lo que, de los 50,000 datos de entrenamiento se tomaron 10,000 para datos de validación, quedando 40,000 para los datos de entrenamiento.

2.2. Arquitecturas propuestas

Para los experimentos se desarrollaron principalmente dos arquitecturas para cada modelo. La primer arquitectura corresponde a una configuración base, convencional para el modelo en cuestión. La segunda arquitectura está configurada de modo que cada modelo tenga aproximadamente 820,000 parámetros entrenables.

2.2.1. Arquitectura MLP

Configuración base:

- **Capa de entrada:** *Flatten* 32×32×3 = 3,072 características de entrada
- **Capas ocultas:** 3 capas totalmente conectadas:
  - *Capa 1:* 3,072 → 512 neuronas
  - *Capa 2:* 512 → 256 neuronas
  - *Capa 3:* 256 → 128 neuronas
- **Capa de salida:** 10 clases con softmax
- **Función de activación:** ReLU
- **Regularización:** Dropout (0.0 o 0.3 según el experimento)

Con esta configuración para el modelo de MLP se tienen $1,738,890$ parámetros entrenables.

Configuración para 820K parámetros:

- **Capa de entrada:** *Flatten* 32×32×3 = 3,072 características de entrada
- **Capas ocultas:** 2 capas totalmente conectadas:
  - *Capa 1:* 3,072 → 256 neuronas
  - *Capa 2:* 256 → 128 neuronas
- **Capa de salida:** *fc3:* 128 → 10 clases
- **Funciones de activación:** ReLU en las capas ocultas
- **Regularización:** Dropout (0.0 o 0.3 según el experimento)

Para ser exactos, el valor de parámetros entrenables en este modelo de MLP teniendo esta configuración es de 820, 874.

2.2.2. Arquitectura CNN

Configuración base:

- **Capa de entrada:** Imágenes de entrada 32×32×3
- **Bloques convolucionales:**
  - **Bloque 1:** 32 filtros, kernel 3×3, Batch Normalization, ReLU y MaxPooling (2×2)
  - **Bloque 2:** 64 filtros, kernel 3×3, Batch Normalization, ReLU y MaxPooling (2×2)
  - **Bloque 3:** 128 filtros, kernel 3×3, Batch Normalization, ReLU y MaxPooling (2×2)
- **Capa de aplanado:** Flatten de 4×4×128 = 2048 características
- **Capas totalmente conectadas:**
  - *Capa 1:* 2048 → 128 neuronas
  - *Capa 2:* 128 → 10 neuronas
- **Funciones de activación:** ReLU
- **Capa de salida:** 10 clases con softmax

Con esta configuración para el modelo de CNN se tienen 357, 258 parámetros entrenables.

Configuración para 820K parámetros:

- **Bloques convolucionales:** 3 bloques
  - **Bloque 1:** 32 filtros, kernel 3×3, *Batch Normalization*, *ReLU*, y *MaxPooling 2×2*
  - **Bloque 2:** 64 filtros, kernel 3×3, *Batch Normalization*, *ReLU*, y *MaxPooling 2×2*
  - **Bloque 3:** 128 filtros, kernel 3×3, *Batch Normalization*, *ReLU*, y *MaxPooling 2×2*
- **Capa de aplanamiento:** *Flatten* (128×4×4 = 2048 características)
- **Capas totalmente conectadas:**
  - *Capa 1:* 2048 → 355 neuronas, activación ReLU
  - *Capa 2:* 355 → 10 clases
- **Regularización:** *Dropout* (0.0 o 0.3 según el experimento)
- **Normalización:** *Batch Normalization* después de cada capa convolucional

El valor exacto de parámetros entrenables de este modelo de CNN teniendo esta configuración es de 824, 651.

2.3. Diseño experimental

2.3.1. Configuración básica

- Optimizador: Adam con tasa de aprendizaje $\alpha = 0.001$
- Batch size: 128
- Número de épocas: 50
- Data augmentation: Random horizontal flips, crops, rotación de 15 y cambio en propiedades de colores de la imagen como brightness=0.2, contrast=0.2 y saturation=0.2

2.3.2. Experimentos específicos

Se realizaron 4 experimentos con el objetivo de capturar diversos comportamientos dado la arquitectura de los modelos en cuestión. De manera básica, los experimentos sin especificación de dropout se da por sentado que se considera como 0. Los experimentos se muestran a continuación:

1. Configuración base.
2. Configuración base y dropout = 0.3.
3. Configuración para los 820K parámetros.
4. Configuración para los 820K parámetros con dropout = 0.3.

2.3.3. Métricas

Al llevar a cabo cada experimento, se considerará el rendimiento de las medidas de desempeñol como *accuracy*, *precision*, *recall* y *f1-score*; así como el tiempo en el entrenamiento que engloba: el tiempo por época, el tiempo total de entrenamiento y el tiempo de inferencia.

3. Resultados

3.1. Experimento 1: Configuración base

En la Figura 2 se observa la matriz de confusión obtenida.

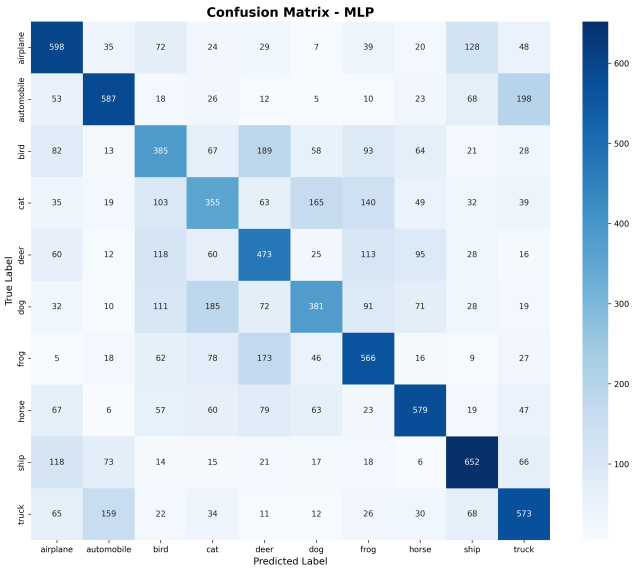

Fig. 2.: Confusion matrix - MLP - Augmentation

Utilizando el modelo de MLP con la arquitectura y configuración base, usando data augmentation con Random Horizontal Flips se obtuvo un test accuracy de 0.5149, con valores de 0.5142, 0.5149 y 0.5131 en las medidas de desempeño macro de *precision*, *recall* y *f1-score*, respectivamente. El tiempo de inferencia fue de 0.6638 segundos. En promedio cada época se realizaba en 11.0383 segundos y el tiempo total de entrenamiento fue de 552.22 segundos.

La matriz de confusión obtenida con el modelo de CNN se muestra en la Figura 3.

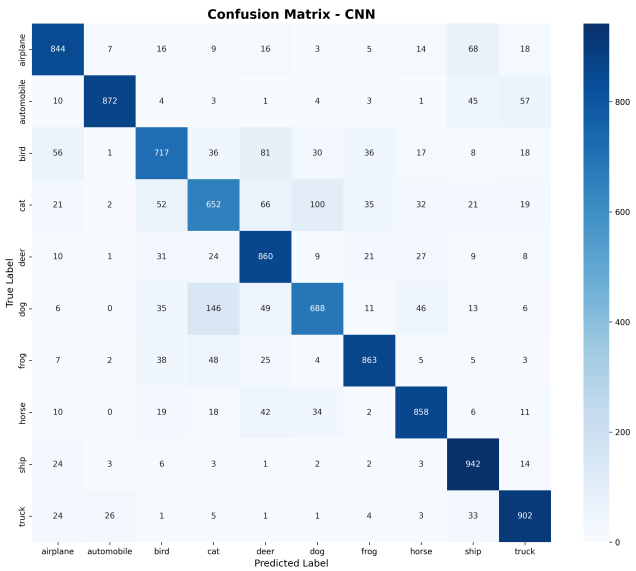

Fig. 3.: Confusion matrix - CNN - Augmentation

Con este modelo de CNN con la arquitectura y configuración base, usando data augmentation con Random Horizontal Flips se obtuvo un test accuracy de 0.8198, con valores de 0.8203, 0.8198 y 0.8185 en las medidas de desempeño macro de *precision*, *recall* y *f1-score*, respectivamente. El tiempo de inferencia fue de 0.8352 segundos, mayor que el modelo MLP. En promedio cada época se realizaba en 13.4065 segundos y el tiempo total de entrenamiento fue de 670.51 segundos.

En la tabla 2 se aprecian los valores de *precision*, *recall* y *f1-score* por clase obtenidos por ambos modelos en este experimento.

En la Figura 4 se muestra el accuracy obtenido tanto en entrenamiento como en validación en ambos modelos a través de las épocas de entrenamiento. Se observa claramente la tendencia que existe hacia un mejor rendimiento por parte de la CNN en entrenamiento. Desde las primeras épocas se van teniendo valores mayores por parte de este modelo CNN sobre el MLP. Se observa que en los valores de validación para CNN se mantuvo oscilando los valores de entrenamiento, no obstante, en el de MLP se tuvo durante todo el entrenamiento un valor mayor en validación que de entrenamiento, no obstante, con un desempeño mucho peor.

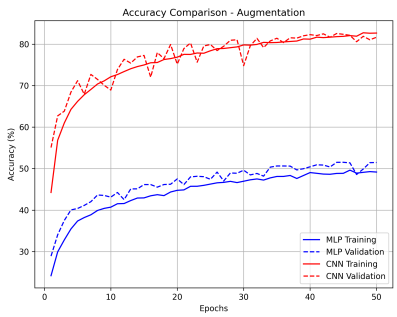

Fig. 4.: Comparación de las curvas de accuracy en entrenamiento y validación del modelo MLP y CNN - Configuración base

3.2. Experimento 2: Configuracion base y dropout = 0.3

En la Figura 5 se observa la matriz de confusión obtenida.

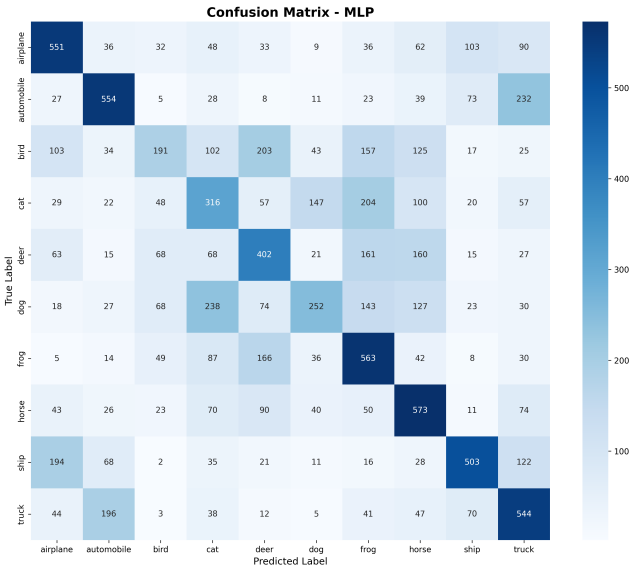

Fig. 5.: Confusion matrix - MLP - Augmentation y Dropout = 0.3

Teniendo la misma arquitectura y configuración del experimento 1, solamente cambiando el valor del dropout de 0 a 0.3 para este experimento se obtuvo un perror rendimiento en general. El test accuracy fue de 0.4449, con valores de 0.4465, 0.4449 y 0.4365 en las medidas de desempeño macro de *precision*, *recall* y *f1-score*, respectivamente. El tiempo de inferencia fue de 0.6623 segundos. En promedio cada época se realizaba en 11.47 segundos y el tiempo total de entrenamiento fue de 573.83 segundos.

La matriz de confusión obtenida con el modelo de CNN se muestra en la Figura 6.

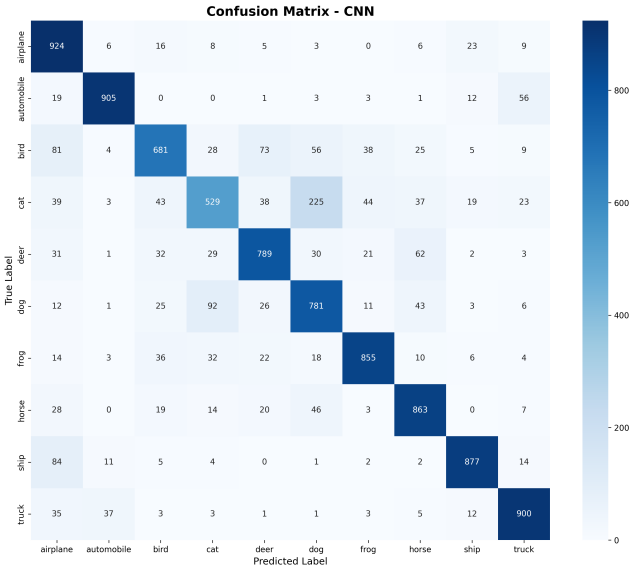

Fig. 6.: Confusion matrix - CNN - Augmentation y Dropout = 0.3

Ahora para el modelo CNN se obtuvieron valores un poco más bajos pero cercanos al experimento 1. En el test accuracy

de 0.8104, con valores de 0.8128, 0.8104 y 0.8081 en las medidas de desempeño macro de *precision*, *recall* y *f1-score*, respectivamente. El tiempo de inferencia fue de 0.9101 segundos. En promedio cada época se realizaba en 13.8559 segundos y el tiempo total de entrenamiento fue de 692.98 segundos.

En la tabla 3 se aprecian los valores de *precision*, *recall* y *f1-score* por clase obtenidos por ambos modelos en este experimento.

Con este experimento trató de observar el comportamiento de cuando se aplica una forma de regularización a los modelos. En las gráficas de la Figura 7 se puede apreciar ahora su comportamiento con dicha regularización de dropout de 0.3. Se observa que los valores de validación tanto para MLP como CNN tienen valores que sobrepasan casi todo el tiempo al de entrenamiento, por lo que se obtuvo realmente una mejor generalización en ambos modelos. El rendimiento de la CNN es un ligeramente menor que en el anterior experimento, pero se ganó en una mejor generalización. Además, se sigue observando el mejor rendimiento por para de la CNN sobre la MLP, incluso desde las primero épocas hasta las 50 realizadas.

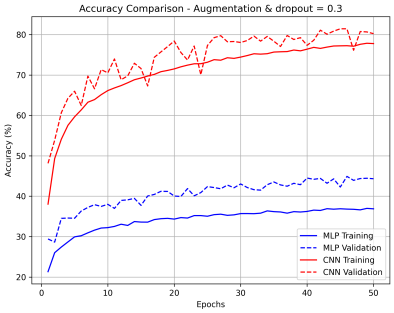

Fig. 7.: Comparación de las curvas de accuracy en entrenamiento y validación del modelo MLP y CNN - Configuración base y Dropout = 0.3

3.3. Experimento 3: Misma cantidad de parámetros - 820K

En la Figura 8 se observa la matriz de confusión obtenida.

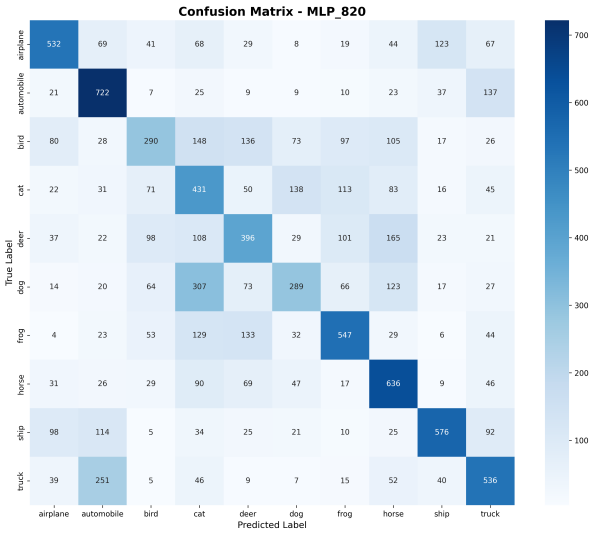

Fig. 8.: Confusion matrix - MLP ≈ 820K parámetros

El principal objetivo de este experimento era observar el rendimiento de ambos modelos siendo más justos al darle el mismo número de parámetros a ambos. Para el modelo de MLP se sigue manteniendo la tendencia en valores cercanos a 0.5 para las diferentes medidas de desempeño. El test accuracy fue de 0.4955, con valores de 0.5002, 0.4955 y 0.4911 en las medidas de desempeño macro de *precision*, *recall* y *f1-score*, respectivamente. El tiempo de inferencia fue de 0.7470 segundos. En promedio cada época se realizaba en 11.1615 segundos y el tiempo total de entrenamiento fue de 558.24 segundos.

Se observa que en relación al tiempo de entrenamiento tanto por época como total, se realizó en menor tiempo, siendo lógico al tener un modelo con un número de parámetros menor que aquel con la arquitectura base.

En la Figura 9 se observa la matriz de confusión obtenida.

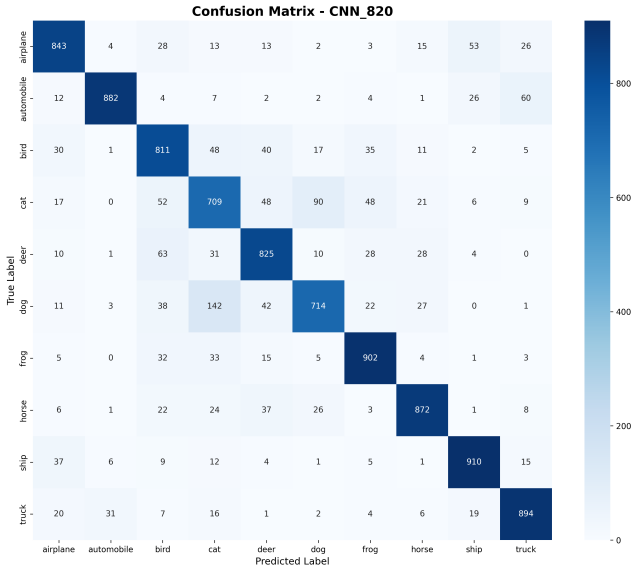

Fig. 9.: Confusion matrix - CNN ≈ 820K parámetros

Aunque no es una garantía, al tener ahora una arquitectura con más parámetros que la base, es decir, es más robusta que antes, se obtuvieron valores más favorables en relación al rendimiento que en los experimentos anteriores.

En el test accuracy de 0.8362, con valores de 0.8377, 0.8362 y 0.8362 en las medidas de desempeño macro de *precision*, *recall* y *f1-score*, respectivamente. El tiempo de inferencia fue de 0.8627 segundos. En promedio cada época se realizaba en 13.5995 segundos y el tiempo total de entrenamiento fue de 680.13 segundos. Aquí se observa que a pesar de tener más parámetros que en la arquitectura anterior, en relación al tiempo por época y total de entrenamiento no cambia demasiado, incluso es un poco menor.

En la tabla 4 se aprecian los valores de *precision*, *recall* y *f1-score* por clase obtenidos por ambos modelos en este experimento.

En la Figura 10 se observa la comparación en el accuracy durante entrenamiento y validación de ambos modelos. Se ve que incluso con ambos modelos teniendo prácticamente la misma cantidad de parámetros, el rendimiento de la CNN

sigue siendo mejor que MLP desde el inicio. Para este caso en específico incluso se obtuvo el mejor valor de accuracy en CNN respecto a los experimentos anteriores.

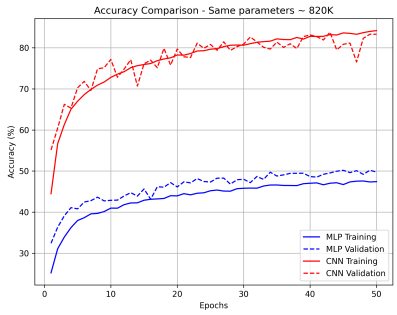

Fig. 10.: Comparación de las curvas de accuracy en entrenamiento y validación del modelo MLP y CNN ≈ 820K parámetros

3.4. Experimento 4: Misma cantidad de parámetros - 820K con dropout = 0.3

En la Figura 11 se observa la matriz de confusión obtenida.

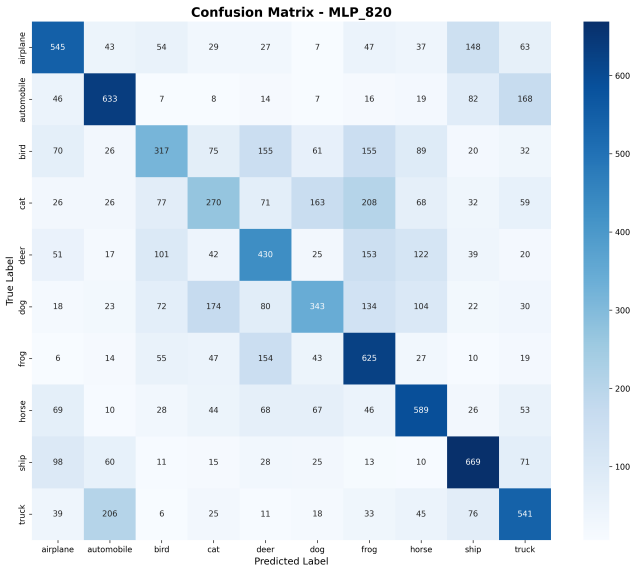

Fig. 11.: Confusion matrix - MLP y Dropout = 0.3 ≈ 820K parámetros

Se realizó un segundo experimento con los modelos con el mismo número de parámetros pero agregando dropout de 0.3 como forma de regularización. Como resultado se obtuvo el prácticamente el mismo comportamiento que antes hablando sobre el valor de accuracy y otras medidas de desempeño.

El test accuracy fue de 0.4962, con valores de 0.4908, 0.4962 y 0.4889 en las medidas de desempeño macro de *precision*, *recall* y *f1-score*, respectivamente. El tiempo de inferencia fue de 0.6702 segundos. En promedio cada época se realizaba en 10.86 segundos y el tiempo total de entrenamiento fue de 542.12 segundos.

Realmente, el comportamiento en relación al rendimiento del modelo siguió siendo el mismo, solamente cambia levemente en los valores de validación.

En la Figura 12 se observa la matriz de confusión obtenida.

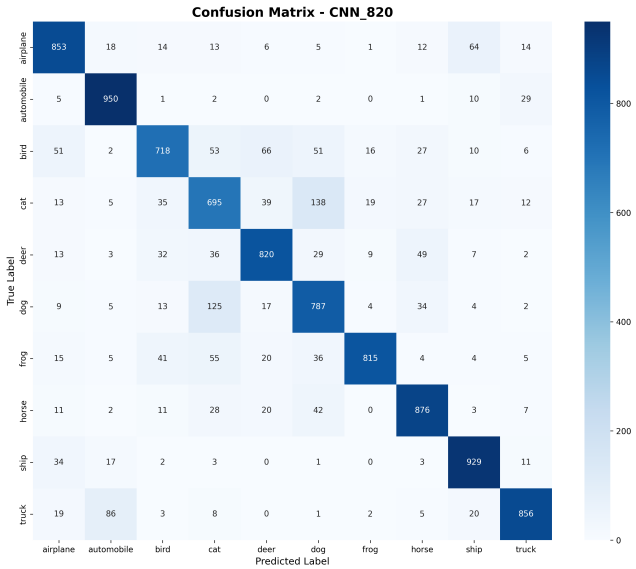

Fig. 12.: Confusion matrix - CNN y dropout = 0.3 ≈ 820K parámetros

Así como sucedió en el experimento 2 respecto al 1, aquí nuevamente se redujo el rendimiento del modelo con dropout ligeramente: el test accuracy fue de 0.8299, con valores de 0.8323, 0.8299 y 0.8297 en las medidas de desempeño macro de *precision*, *recall* y *f1-score*, respectivamente. El tiempo de inferencia fue de 1.0257 segundos. En promedio cada época se realizaba en 13.92 segundos y el tiempo total de entrenamiento fue de 696.09 segundos.

En relación al tiempo tomado durante el entrenamiento no cambia en realidad respecto al experimento anterior. Lo importante de este experimento es observar el comportamiento de los valores en la validación, asegurando que los modelos no se sobreajusten a los datos.

En la tabla 5 se aprecian los valores de *precision*, *recall* y *f1-score* por clase obtenidos por ambos modelos en este experimento.

En la Figura 13 se puede observar un comportamiento similar que en el experimento anterior, aunque con algunos picos en el accuracy de validación, no obstante, no se observó una mejora significativa para ambos modelos. El rendimiento sigue siendo el mismo, destancando la CNN sobre el MLP.

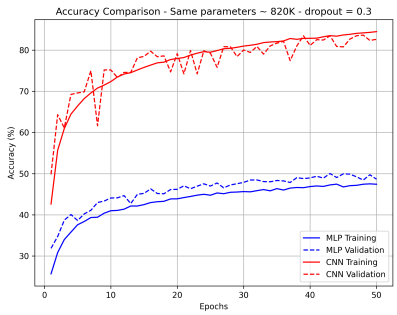

Fig. 13.: Comparación de las curvas de accuracy en entrenamiento y validación del modelo MLP y CNN y dropout = 0.3 ≈ 820K parámetros

En la tabla 1 se muestra el resumen del rendimiento de los modelos en todos los experimentos realizados.

Experimento	Model	Test Accuracy	Test Loss	Precision	Recall	F1 Score	Trainable Params	Inference Time (s)	Inference Speed (samples/s)	Final Epoch	Mean Epoch Time (s)	Total Training Time (s)
1	MLP	0.5149	1.3625	0.5142	0.5149	0.5131	1,738,890	0.6638	15,064.91	50	11.0383	552.22
1	CNN	0.8198	0.5397	0.8203	0.8198	0.8185	357,258	0.8352	11,973.77	50	13.4065	670.51
2	MLP	0.4449	1.5725	0.4465	0.4449	0.4365	1,738,890	0.6623	15,098.04	50	11.4707	573.83
2	CNN	0.8104	0.5481	0.8128	0.8104	0.8081	357,258	0.9101	10,987.62	50	13.8559	692.98
3	MLP	0.4955	1.3944	0.5002	0.4955	0.4911	820,874	0.7470	13,386.51	50	11.1615	558.24
3	CNN	0.8362	0.4966	0.8377	0.8362	0.8362	824,651	0.8627	11,591.17	50	13.5990	680.13
4	MLP	0.4962	1.4092	0.4908	0.4962	0.4889	820,874	0.6702	14,921.95	50	10.8592	543.12
4	CNN	0.8299	0.5111	0.8323	0.8299	0.8297	824,651	1.0257	9,749.34	50	13.9169	696.09

Table 1.: Rendimiento de los modelos en los experimentos realizados

4. Discusión de los trade-offs

En todos los experimentos realizados la arquitectura de la CNN se mantuvo siempre por encima en rendimiento de la MLP de inicio a fin. En promedio el valor del accuracy para el MLP fue de aproximadamente un 49%, mientras que para la CNN fue de un 82%. En los valores de las medidas de desempeño como *precision*, recall y *f1-score* se mantuvieron por esos valores. Eso se pudo observar en experimentos cuando la CNN tenía 357,258 parámetros y el MLP 1,738,890 y cuando se hacía de manera más justa teniendo 824,651 la CNN y 820,874 el MLP.

Con esto se puede destacar que la arquitectura CNN aprovecha de mejor manera la estructura espacial de las imágenes del dataset CIFAR-10. Los filtros de la CNN se aplican en cada región de la imagen, mientras que la MLP tiene pesos distintos para cada píxel, lo que incrementa el número de parámetros. Otro punto importante es que la CNN aprovecha mejor el data augmentation ya que los filtros son capaces de detectar bordes, o texturas incluso si se encuentran desplazados o rotados. Además, las capas de pooling también ayudan a reducir la sensibilidad a pequeñas traslaciones o distorsiones de las imágenes. En cambio, el MLP no puede sacar provecho de esto; si un patrón se mueve de lugar, cambia la combinación de entradas y el modelo debe volver a aprenderlo de nuevo.

Otra ventaja de las CNN es que al conectar las neuronas a pequeñas regiones locales de la imagen produce que las primeras capas aprendan características locales simples, como bordes, esquinas o texturas, y conforme se va profundizando en las capas se combinan dichas características locales para formar patrones más complejos como partes de objetos, atributos particulares de lo que se quiere reconocer. En los experimentos de las CNN, se tuvo mejor rendimiento en la arquitectura con un mayor número de parámetros y donde se obtuvo el incremento de parámetros fue en las capas totalmente conectadas, donde posiblemente aprovecharon de mejor forma toda la información proveniente de las capas de convolución y así tener mejor precisión en su predicción. La diferencia fue aproxidamente de 1.6% de mejora en accuracy en el experimento 1 y 3.

Finalmente, se pudo entonces verificar que la CNN tiene un mejor rendimiento que el MLP en todos los experimentos al ser capaz de aprender representaciones jerárquicas del dataset, mientras que el MLP sólo es capaz de memorizar relaciones directas entre los píxeles.

Conceptualmente, el MLP es más simple debido a que no tiene la restricción en las conexiones de las neuronas de una capa a la siguiente, generando un mayor número de parámetros. Sin embargo, esta simpleza tanto conceptual como operacional (multiplicación matricial es más simple que todo el proceso de convolución y max-pooling) hace que las operaciones sean menos costosas. Como resultado de esto, en todos los experimentos el MLP tuvo un menor tiempo en: inferencia (promedio de 0.685 s contra 0.901 s de la CNN), el tiempo promedio por época (promedio de 11.13 s contra 13.69 s de la CNN) y el tiempo total de entrenamiento (promedio 556.85 s contra 684.93 s de la CNN). Aún así, la convergencia en general hacia un buen valor de accuracy fue mucho más lento, es decir, requeriría más épocas para alcanzar el mismo rendimiento que la CNN, eso si no se considera que se pueda sobreajustar a los datos.

Al utilizar la librería de PyTorch, las convoluciones se encuentran implementadas usando kernels de CUDA, o bien, MPS en Mac, que son muy eficientes, aprovechando las operaciones vectorizadas y memoria compartida. Lo mismo pasa para la arquitectura del MLP, se tienen capas vectorizadas que permite realizar las multiplicaciones matriciales en lugar de bucles.

La convergencia más rápida de las CNN se da por su aprendizaje de características útiles desde el inicio del proceso, que se van reutilizando en las capas posteriores. Las MLPs no tienen esa estructura y por ende, deberían descubrir dichas relaciones desde cero, haciendo su aprendizaje más lento e ineficiente.

5. Conclusiones y reflexiones

En este trabajo se pudo contrastar de manera sistemática el desempeño de dos arquitecturas representativas en el campo del aprendizaje profundo: una red neuronal multicapa (MLP) y una red convoluciones (CNN), tanto con diferentes como con los mismos parámetros. A través de múltiples experimentos bien definidios se observó, de forma consistente, que la CNN supera a la MLP en todas las medidas de desempeño, mostrando un *accuracy*, *precision*, *recall* y *f1-score* signi-

ficativamente mayor. Para ambas arquitecturas el sobreajuste no se logró ver con claridad debido a las pocas épocas tomadas para el MLP, no obstante, se observa la mayor capacidad de generalización por parte de la CNN, dada la tendencia de la curva de accuracy de la MLP, destinada a necesitar más épocas y una mayor memorización de los datos para, de alguna forma, intentar lograr valores más altos.

Este comportamiento confirma la importancia de la estructura espacial en el procesamiento de imágenes. Mientras el MLP trata de aprender directamente las relaciones entre píxeles de manera independiente, la CNN explota las características locales de las imágenes y construye representaciones jerárquicas que capturan border, formas y patrones más complejos. En consecuencia, la CNN no solamente obtiene mejores resultados, sino que lo hace con un menor número de parámetros y una convergencia más rápida durante el entrenamiento. Incluso, igualando el número de parámetros entre ambas arquitecturas, el rendimiento sigue siendo mayor por parte de la CNN. En ese experimento, se incrementó el número de parámetro de la CNN por más del doble y no se obtuvo una diferencia abismal.

Sin embargo, el trabajo también da evidencia que el mayor rendimiento de la CNN implica un costo computacional más alto, tanto en tiempo de entrenamiento como en tiempo de inferencia. Este hecho refleja un *trade-off* en este campo de aprendizaje profundo: estos modelos más expresivos y potentes requieren mayor cantidad de recursos y optimizaciones de hardware específicas para exprimir todo su potencial.

El contraste entre ambas arquitecturas resalta la relevancia de comprender la naturaleza de los datos antes de seleccionar un modelo. Un MLP podría ser suficiente en tareas donde las dependencias espaciales no son tan determinantes, mientras que usar una CNN resulta útil e indispensable cuando la estructura de los datos contiene información de la posición o local relevante, como lo es en el caso de las imágenes.

Así mismo, el proceso experimental permitió observar que la forma de regularización de dropout ayuda de cierto modo en la parte de sobreajuste, haciendo que los datos de validación tengan un poco de mayor relevancia contra los de entrenamiento.

Finalmente, se puede concluir que la efectividad de un modelo de aprendizaje profundo no depende solamente de su complejidad, sino de la forma en cómo se adecua a la estructura del problema abordado. Los resultados obtenidos pueden mostrar la importancia sobre el diseño inductivo, en donde posiblemente hayan mejoras de rendimiento cuando se incorporan supuestos dada la naturaleza de los datos.

References

1) S. B. Driss, M. Soua, R. Kachouri, and M. Akil, "A comparison study between mlp and convolutional neural network models for character recognition," in *Proceedings of SPIE Conference on Real-Time Image and Video Processing*, (Anaheim, United States), p. 1022306, April 2017.
2) N. Sharma, V. Jain, and A. Mishra, "An analysis of convolutional neural networks for image classification," *Procedia Computer Science*, vol. 132, pp. 377–384, 2018.
3) G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, pp. 82–97, Nov 2012.
4) K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
5) A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.
6) K. Simonyan and A. Zisserman, "Very deep convolutional networks for large scale image recognition," *ArXiv preprint arXiv:1409.1556*, 2014.
7) C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.
8) A. F. Agarap, "An architecture combining convolutional neural network (cnn) and support vector machine (svm) for image classification," *arXiv preprint arXiv:1712.03541v2*, 2019.
9) C. L. Chowdhary, M. Mittal, P. A. Pattanaik, and Z. Marszalek, "An efficient segmentation and classification system in medical images using intuitionist possibilistic fuzzy c–mean clustering and fuzzy svm algorithm," *Sensors*, vol. 20, no. 14, p. 3903, 2020.
10) O. Sbai, C. Couprie, and M. Aubry, "Impact of base data set design on few–shot image classification," in *Computer Vision–ECCV 2020: 16th European Conference*, (Glasgow, United Kingdom), pp. 597–613, 2020.
11) J. G. A. Barbedo, "Impact of data set size and variety on the effectiveness of deep learning and transfer learning for plant disease classification," *Computers and Electronics in Agriculture*, vol. 153, pp. 46–53, 2018.
12) D. Bashir, G. D. Montanez, S. Sehra, P. S. Segura, and J. Lauw, "An information–theoretic perspective on overfitting and underfitting," in *AI 2020: Advances in Artificial Intelligence: 33rd Australasian Joint Conference*, (Canberra, Australia), pp. 347–358, 2020.

Experimento	Clase	MLP			CNN		
		Precisión	**Recall**	**F1**	**Precisión**	**Recall**	**F1**
1	airplane	0.5363	0.5980	0.5655	0.8340	0.8440	0.8390
1	automobile	0.6298	0.5870	0.6077	0.9540	0.8720	0.9112
1	bird	0.4002	0.3850	0.3925	0.7802	0.7170	0.7473
1	cat	0.3927	0.3550	0.3729	0.6907	0.6520	0.6708
1	deer	0.4216	0.4730	0.4458	0.7531	0.8600	0.8030
1	dog	0.4891	0.3810	0.4283	0.7863	0.6880	0.7339
1	frog	0.5058	0.5660	0.5342	0.8788	0.8630	0.8708
1	horse	0.6076	0.5790	0.5929	0.8529	0.8580	0.8554
1	ship	0.6192	0.6520	0.6352	0.8191	0.9420	0.8763
1	truck	0.5401	0.5730	0.5560	0.8542	0.9020	0.8774

Table 2.: Precisión, Recall y F1 Score por clase para el experimento *Augmentation*.

Experimento	Clase	MLP			CNN		
		Precisión	**Recall**	**F1**	**Precisión**	**Recall**	**F1**
2	airplane	0.5116	0.5510	0.5306	0.5116	0.5510	0.5306
2	automobile	0.5585	0.5540	0.5562	0.5585	0.5540	0.5562
2	bird	0.3906	0.1910	0.2565	0.3906	0.1910	0.2565
2	cat	0.3068	0.3160	0.3113	0.3068	0.3160	0.3113
2	deer	0.3771	0.4020	0.3892	0.3771	0.4020	0.3892
2	dog	0.4383	0.2520	0.3200	0.4383	0.2520	0.3200
2	frog	0.4039	0.5630	0.4703	0.4039	0.5630	0.4703
2	horse	0.4398	0.5730	0.4976	0.4398	0.5730	0.4976
2	ship	0.5967	0.5030	0.5458	0.5967	0.5030	0.5458
2	truck	0.4419	0.5440	0.4877	0.4419	0.5440	0.4877

Table 3.: Precisión, Recall y F1 Score por clase para el experimento *Augmentation & dropout = 0.3*.

Experimento	Clase	MLP			CNN		
		Precisión	**Recall**	**F1**	**Precisión**	**Recall**	**F1**
3	airplane	0.6059	0.5320	0.5666	0.8507	0.8430	0.8468
3	automobile	0.5528	0.7220	0.6262	0.9494	0.8820	0.9145
3	bird	0.4374	0.2900	0.3488	0.7608	0.8110	0.7851
3	cat	0.3110	0.4310	0.3613	0.6850	0.7090	0.6968
3	deer	0.4263	0.3960	0.4106	0.8033	0.8250	0.8140
3	dog	0.4426	0.2890	0.3497	0.8216	0.7140	0.7640
3	frog	0.5497	0.5470	0.5484	0.8558	0.9020	0.8783
3	horse	0.4949	0.6360	0.5567	0.8844	0.8720	0.8781
3	ship	0.6667	0.5760	0.6180	0.8904	0.9100	0.9001
3	truck	0.5149	0.5360	0.5252	0.8756	0.8940	0.8847

Table 4.: Precisión, Recall y F1 Score por clase para el experimento *Mismos parámetros* $\sim$ *820K*

Experimento	Clase	MLP			CNN		
		Precisión	**Recall**	**F1**	**Precisión**	**Recall**	**F1**
4	airplane	0.5630	0.5450	0.5539	0.8338	0.8530	0.8433
4	automobile	0.5983	0.6330	0.6152	0.8692	0.9500	0.9078
4	bird	0.4354	0.3170	0.3669	0.8253	0.7180	0.7679
4	cat	0.3704	0.2700	0.3123	0.6827	0.6950	0.6888
4	deer	0.4143	0.4300	0.4220	0.8300	0.8200	0.8249
4	dog	0.4519	0.3430	0.3900	0.7207	0.7870	0.7524
4	frog	0.4371	0.6250	0.5144	0.9411	0.8150	0.8735
4	horse	0.5306	0.5890	0.5583	0.8439	0.8760	0.8597
4	ship	0.5952	0.6690	0.6299	0.8699	0.9290	0.8985
4	truck	0.5123	0.5410	0.5263	0.9068	0.8560	0.8807

Table 5.: Precisión, Recall y F1 Score por clase para el experimento *Mismos parámetros ~ 820K & dropout = 0.3*.