Escuela de Ingeniería y Ciencias

Diseño de redes neuronales y aprendizaje profundo (TC2035.201)

Arquitecturas de deep learning para la clasificación de moda
Análisis comparativo entre CNN y ViT

Raúl Correa Ocañas	A01722401
Alfredo André Durán Treviño	A01286222
José Manuel Guerrero Arellano	A01747623
Eliani González Laguna	A00836712

19 de octubre de 2024

Docente:

Dr. Santiago Enrique Conant Pablos

Contenido

Introducción

El reconocimiento de imágenes por medio de redes neuronales es un tema muy estudiado y relevante en la época actual, habiendo diferentes modelos para resolver todo tipo de problemas. En este proyecto de investigación se realizó con la finalidad de realizar una exploración de dos modelos de redes neuronales profundas para crear un análisis de clasificación con respecto a la base de datos aplicada.

En este proyecto nos enfocamos en el análisis comparativo de dos modelos diferentes de redes neuronales con el mismo set de imágenes, para tener una buena investigación de su funcionamiento distinto y ver sus fortalezas y dificultades en su ejecución.

Nuestro primer modelo es una red neuronal convolucional (CNN) y el segundo es un Visual Transformer. Nuestra base de datos es disponible para descargar y usar con las librerías de Keras, se llama Fashion MNIST, que contiene 70000 imágenes de escala de grises de artículos de moda, cada imagen tiene un tamaño de 28x28 píxeles en 10 categorías diferentes de prenda, la siguiente figura contiene un ejemplo de cada categoría de imagen.

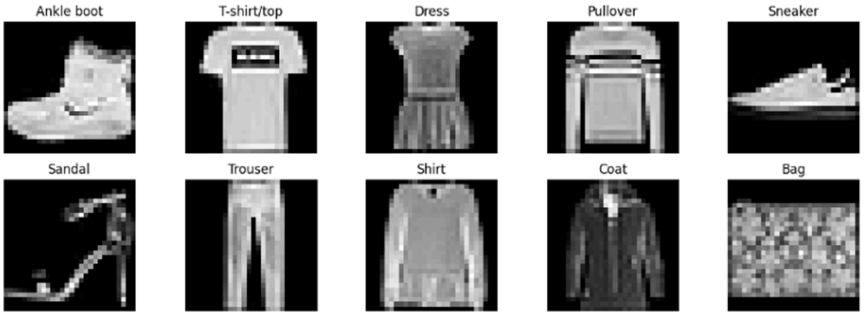

Imagen 1. Demostración de las categorías de la base de datos.

Este estudio es crucial, primero porque el uso de esta base de datos nos proporciona un escenario realista y muy desafiante para estos modelos en el reconocimiento de imágenes, además de que el usar dos modelos distintos y compararlos nos hace notar más fácilmente las fortalezas y debilidades de cada uno y podemos obtener buenos insights de estos. Además de ver cuál de estos puede

tener una mayor precisión en este tipo de problemas y si hay alguna diferencia significativa en su capacidad de clasificación.

Los hallazgos en este estudio pueden guiar profesionales que requieran de soluciones de problemas de este tipo para que puedan usar estas herramientas adecuadamente de acuerdo a sus situaciones.

Marco teórico

La aplicación de aprendizaje profundo para la clasificación de imágenes es un tema bastante explorado en los años recientes. En el año 2012, Alex Krizhevsky, Ilya Sutskever y Geoffrey Hinton investigaron el uso de redes neuronales profundas para el dataset de ImageNet. Su exploración destaca el uso de filtros para aprender características significativas en las imágenes para con estas hacer mejores predicciones de la categoría a la que pertenece (Krizhevsky et al., 2012).

Esta técnica resultó ser efectiva, desarrollando el comienzo de las redes neuronales convolucionales. Muchos años más adelante, en 2017, se publicó el paper *Attention is all you need*, en donde se propuso utilizar un mecanismo de atención que permite a la arquitectura del modelo centrarse en diferentes partes del input secuencial en cada capa, originalmente aplicado para problemas de grandes volúmenes de secuencias (como lo es texto) al particionarse en divisiones llamadas tokens (Vaswani et al., 2024).

En 2020, se adaptó un transformer que solo funcionaba con codificador para la visión artificial, lo que dio lugar al ViT. El autocodificador enmascarado amplió el ViT para que funcionara con entrenamiento no supervisado. El transformador de visión y el auto codificador enmascarado, a su vez, estimularon nuevos desarrollos en las redes neuronales convolucionales.

En la práctica de problemas de clasificación para aprendizaje automático, existen múltiples métricas que apoyan a la cuantificación del desempeño de los modelos. La función de pérdida en una red neuronal es una métrica que se utiliza para medir qué tan bien el modelo predice los valores correctos comparados con los

valores reales. Existen dos funciones comunes para cuantificar la pérdida, la entropía cruzada categórica y la entropía cruzada esparza. La principal diferencia entre estas es la forma en que manejan los datos de entrada, la entropía cruzada categórica se usa cuando los datos están codificados one-hot, mientras que la entropía cruzada esparza se usa cuando las etiquetas de las clases son enteros. Para fines prácticos, este reporte busca profundizar y explorar redes que estén diseñadas tal que utilicen la entropía cruzada categórica. Esta es calculada con la siguiente fórmula:

$$\mathcal{L} = -\frac{1}{N}\sum_{j=1}^{N}\sum_{i \in class_i} y_{ij} \log \hat{y}_{ij}$$

Donde la letra j denota el índice de la muestra y la letra i denota el índice correspondiente a cada categoría. Si la red tiene C clases, entonces habrá C neuronas de salida, cada una representando una probabilidad de predicción. Por otro lado, se asume que solo una de las predicciones es la correcta. Esta tendría un valor de 1 para su respectiva clase correcta, y 0 en cualquier otro caso. De esta forma, se hace un promedio sobre todas las muestras del logaritmo natural de la probabilidad de selección de su verdadera clase.

En cuanto a métricas de clasificación fuera de la función de pérdida, la exactitud es una métrica fácil de interpretar y comúnmente utilizada. Esta se calcula con la siguiente fórmula:

$$Exactitud = \frac{Número\ de\ predicciones\ correctas}{Número\ total\ de\ predicciones}$$

Aún y siendo sencilla de comprender, la métrica de exactitud tiene desventajas. Si bien explica la proporción de aciertos a predicciones, esta tasa no explica nada sobre el rendimiento sobre cada categoría. Una buena métrica que complementa a la interpretación de la exactitud es la puntuación F1, siendo la media armónica de la precisión y la recuperación. Una forma de resumir esta métrica es con su promedio ponderado según la cantidad de muestras para cada categoría.

Metodología

Como se mencionó previamente, se utilizaron los datos proporcionados por la librería Keras, específicamente el conjunto Fashion MNIST desarrollado por Zalando Research. Esta base de datos contiene 70,000 imágenes, divididas en 60,000 para entrenamiento y 10,000 para pruebas. Cada imagen tiene una resolución de 28x28 píxeles y está clasificada en una de 10 categorías de ropa. Aunque existen diversos ejemplos de arquitecturas de redes neuronales para la clasificación de imágenes, en este análisis comparativo se emplearon una red neuronal convolucional (CNN) y un visual transformer.

El primer modelo, la CNN, tiene sus orígenes en los trabajos de Yann LeCun y colaboradores a finales de los 80 y principios de los 90. Su funcionamiento se basa en capas convolucionales que aplican filtros a la imagen de entrada para detectar características locales, como bordes, texturas y esquinas.

Además, utiliza capas de pooling o subsampling, que reducen la dimensionalidad de los mapas de características conservando las más relevantes. En este caso, se emplearon operaciones de MaxPooling, que seleccionan el valor máximo en una región específica del mapa de características. Posteriormente, tras varias capas de convolución y pooling, los mapas de características se aplanan y pasan por una o más capas completamente conectadas, similares a las de una red neuronal tradicional. Estas capas combinan las características aprendidas para realizar la clasificación final.

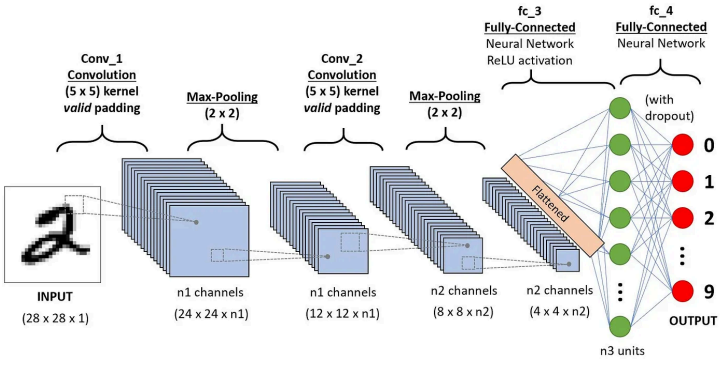

Imagen 2. Ejemplo de funcionamiento de una red convolucional.

Adicionalmente, tras cada capa convolucional se aplica una función de activación, lo que introduce no linealidades en el modelo, permitiendo que la red

aprenda relaciones más complejas entre las características. En este caso, se utiliza comúnmente la función de activación ReLU (Rectified Linear Unit), que ayuda a modelar patrones no lineales y mejora el rendimiento del modelo en tareas de clasificación. Esta combinación de capas convolucionales, funciones de activación y pooling permite que la red extraiga y combine características a diferentes niveles de abstracción, facilitando una clasificación más precisa de las imágenes.

Durante la implementación de la arquitectura para el primer modelo, se aplicaron los siguientes hiperparámetros para adecuar el modelo a una versión mucho más precisa, obteniendo clasificaciones acordes a las características de los datos del dataset.

random_seed = 42
SCALING_FACTOR = 2
AUGMENTATION = True
N_EPOCHS = 500
IMAGE_SIZE = (28, 28)
BUFFER_SIZE = 10_000
BATCH_SIZE = 1024
PATIENCE = 20

Para construir el modelo, se implementaron diversas modificaciones para mejorar la precisión de los resultados. Una de ellas fue la aplicación de una semilla aleatoria, lo que garantiza la consistencia en la partición de los datos y mantiene el mismo estado aleatorio a lo largo de los experimentos. Se seleccionó un número alto de épocas para asegurar un aprendizaje más completo y con un mejor rendimiento. Sin embargo, se incluyó el parámetro patience, que permite detener el entrenamiento si el modelo deja de mejorar después de un número específico de épocas, evitando así el sobre ajuste. El tamaño de lote se ajustó para optimizar la eficiencia computacional, mejorando la convergencia y maximizando el uso de la memoria, lo que reduce el tiempo de entrenamiento.

Se aplicó también un factor de escalamiento para normalizar los datos y se ajustó la resolución de las imágenes a 28x28 píxeles utilizando el parámetro

image_size. Además, se implementó un buffer que almacena temporalmente los datos antes del entrenamiento, permitiendo mezclar los datos de manera más eficiente y acelerar el proceso al reducir los tiempos de espera para acceder a ellos. Por último, se utilizó un criterio de parada anticipada que interrumpe el entrenamiento si no se observan mejoras después de 20 épocas consecutivas, lo que previene el sobre ajuste y ahorra tiempo computacional.

En cuanto a la arquitectura del primer modelo, se definió una capa de entrada correspondiente al tamaño de las imágenes. Se implementó una capa de data augmentation, que aplica transformaciones aleatorias como volteos horizontales y verticales, y un zoom de entre 0% y 20%, mejorando la capacidad de generalización del modelo al introducir variaciones en las imágenes de entrada. Se utilizaron dos capas convolucionales, cada una seguida de batch normalization y MaxPooling. La primera capa convolucional aplica 64 filtros de 3x3 con activación ReLU, mientras que la segunda utiliza 128 filtros. Posteriormente, se realiza un aplanamiento que convierte la matriz 2D de características en un vector 1D, preparándola para las capas completamente conectadas. Se incluyó una capa dropout con una tasa de 50% para evitar el sobre ajuste, seguida de una capa densa con 128 neuronas, activación ReLU y regularización L2 para prevenir el sobre ajuste. Finalmente, la capa de salida consta de 10 neuronas que corresponden a las 10 clases, con una activación softmax para calcular las probabilidades de clasificación.

Para la compilación del modelo, se utilizó el optimizador Adam, con la función de pérdida de entropía cruzada categórica. Las métricas de evaluación incluyeron la exactitud y la puntuación F1 ponderada, garantizando una evaluación robusta del desempeño del modelo.

Para el segundo modelo se decidió implementar un modelo visión transformer, el cual parte como una implementación de los módulos de atención de una red neuronal transformer, presentada por primera vez en 2017 a través del reporte *Attention is All you need (2017).* Una red visión transformer logra utilizar el concepto de atención para dividir una imagen en parches, así como la red transformer logra dividir las oraciones en palabras. A continuación se presenta un diagrama de la red neuronal ViT.

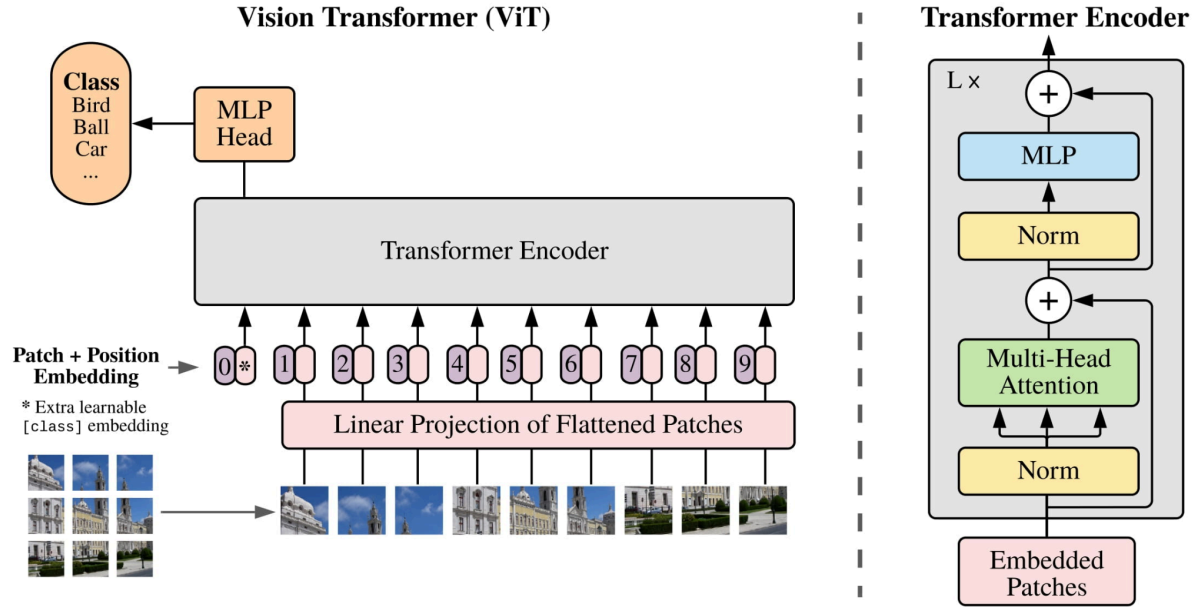

Imagen 3. De una red vision transformer (2021)

Para poder realizar la integración de datos a parches, es necesario realizar una transformación de dimensiones. En este caso, al estar trabajando con datos del conjunto Fashion-MNIST, se realizó una transformación de (60000, 785) y (10000 y 785) para los conjuntos de entrenamiento y prueba, respectivamente, a (60000, 28, 28, 1) y (10000, 28, 28, 1), tomando en cuenta que las imágenes están en formato monocanal, es decir, que su formato es blanco y negro. Posteriormente, se dividen en 7 parches de 4*4 píxeles encadenados.

Una vez se dividen en parches, se les asigna un token de clasificación, que funge como un parche adicional que representa al conjunto de parches de esa imagen. Posteriormente, los datos entran a un codificador de atención, en donde cada parche representa un objeto en un espacio vectorial de capas ocultas, para posteriormente ser procesado por un perceptrón multicapa y obtener una agrupación de los tokens de atención en dimensiones reducidas.

A manera de poder implementar la red en python, se tomó como referencia el modelo programado por Uygar Kurt, tomando en cuenta que parte del código utilizado en el reporte donde se describe la red por primera vez. Como medida para evitar el overfitting, se implementaron técnicas de data augmentation, tales como el

uso de dropout en neuronas y la rotación de imágenes. Adicionalmente, se implementó otra capa de atención para poder experimentar con el modelo, además de modificar los hiperparámetros presentados a continuación:

```
random_seed = 42
batch_size = 256
num_epochs = 60
learning_rate = 1e-4
n_classes = 10
patch_size = 4
image_size = 28
input_channels = 1
num_heads = 8
batch_size = 256
dropout = 0.2
hidden_dimension = 768
adam_weight_decay = 1e-3
adam_betas = (0.9, 0.999)
activation = 'gelu'
num_encoders = 6
embedding_dimension = patch_size*patch_size*input_channels # 4*4*1 = 16
num_patches = (image_size//patch_size)**2 #(28/4)^2 = 49
```

En la ejecución y la adecuación de la arquitectura aplicada al segundo modelo, donde se ejecutaron los siguientes hiperparámetros con el objetivo de precisar el modelo a resoluciones más idóneas, para contar con clasificaciones adecuadas en torno a las características de los datos del dataset.

En la arquitectura del segundo modelo se aplicaron las siguientes adecuaciones para contar con resultado más preciso, siendo estos cambios, la aplicación de una semilla para trabajar bajo el mismo conjunto de datos, se modificó el batch para contar con una mayor eficiencia computacional, puesto que, se reduce el tiempo de entrenamiento por las mejoras en la convergencia y el aumento de memoria, concediendo una adaptación más dinámica, así mismo, al aumentar la

cantidad de épocas se cuenta con un aprendizaje más preciso y enfocado a cada caracteristica. Contando con una tasa de aprendizaje muy precisa, en conjunto de un factor de escalamiento, aplicando la normalización de los datos en cuanto a la resolución de las imágenes, estas cuentan con un tamaño de 28 píxeles de ancho por 28 píxeles de alto.

Así mismo, el tamaño de cada parche por el cual se divide la imagen es de 4x4 píxeles, donde se hace referencia a una pequeña sección de la imagen para su procesamiento individual, de igual forma, cuenta con un espacio oculto en el modelo, siendo estas las representaciones internas del aprendizaje a lo largo de las capas ocultas, con una tasa de decaimiento del peso con el optimizador Adam, aplicado para una regularización y de un sobre ajuste, como con la función de activación gelu (gaussian errror linear unit).

Resultados

Red convolucional profunda

Desde el inicio el modelo se observa tener buenos resultados, en su primera época teniendo un accuracy de 0.66, terminando con un 0.92 en el entrenamiento. Como se observa en los siguientes diagramas, el modelo es bastante eficiente desde las primeras épocas, lo que indica que aun con menos épocas podría dar resultados aceptables. Se ve un ligero ruido en las métricas de verificación, pero puede ser aceptable debido a la poca variación que estos tienen.

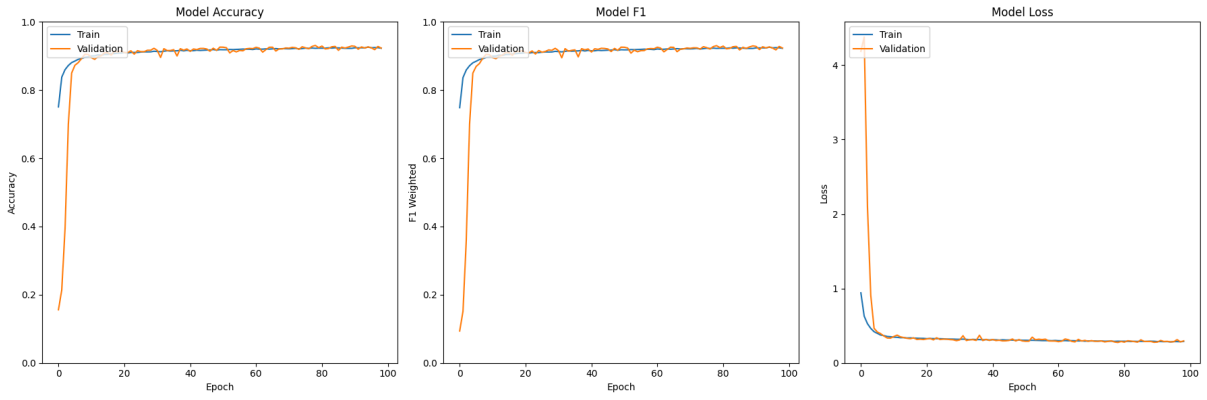

Diagrama 1.

CNN	Pérdida	Exactitud	F1
Entrenamiento	0.233	0.945	0.944
Validación	0.275	0.932	0.931
Prueba	0.302	0.921	0.920

Diagrama 2.

En esta tabla se observa una comparación de las métricas al final del entrenamiento y validación de nuestro modelo, como vemos estas métricas son casi iguales, lo que nos dice que no hay sobre ajuste entre los datos de entrenamiento, por lo que parece ser que el modelo es muy eficiente, ya que como vemos sus métricas son mayores a 0.90, que en un tipo de problema como este que tenemos más de 10 clases diferentes son resultados muy buenos, que nos dice que estos datos funcionan muy bien con una red neuronal convolucional.

Vision transformer

El entrenamiento comenzó con pérdidas de 2.14 y 47.92 para entrenamiento y validación, respectivamente, además de comenzar con accuracy de 0.16 y f1 de 0.16, concluyendo con los resultados mostrados a continuación:

CNN	Pérdida	Exactitud	F1
Entrenamiento	0.75	0.71	0.70
Validación	16.48	0.74	0.72
Prueba	0.31	0.7	0.7

Diagrama 3.

Posteriormente, se decidió realizar una prueba del modelo con 6 muestras para visualizar el comportamiento del modelo:

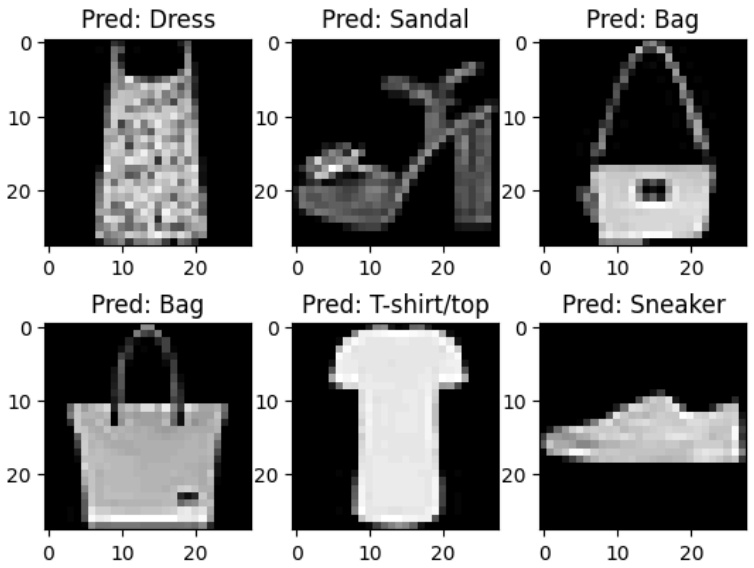

Diagrama 4.

Análisis

Como se puede observar, el procesamiento del modelo convolucional tuvo mejores resultados para clasificar prendas de ropa del conjunto de datos Fashion-MNIST con una ventaja aproximada de 20-25% para las métricas analizadas. Por otro lado, mientras las métricas del primer modelo son consistentes tanto en sus fases de entrenamiento como en las de validación y prueba, se puede observar cómo existe una inmensa diferencia entre la pérdida en su etapa de validación con respecto a la de entrenamiento y prueba a pesar de disminuir considerablemente a lo largo de su entrenamiento.

Fue una sorpresa el ver los resultados de este experimento, ya que confiábamos en que el visual transformer sería el más eficiente, dado que es un modelo más moderno y prometía una mayor eficacia en problemas como estos. Debido a lo anterior, se decidió realizar modificaciones al modelo implementado, añadiendo una capa adicional de atención y modificando los hiperparámetros, sin embargo, seguía teniendo resultados por debajo de los esperados en cuanto a resultados finales en sus métricas, pero con mejor desempeño a lo largo de su entrenamiento.

En cuanto a complejidad de arquitectura, el visual transformer definitivamente fue el más complicado de desarrollar. Esto se debe a la estructura compleja que

implica implementar un modelo así de poderoso. Las redes convolucionales ya han sido bastante estudiadas, por lo tanto, su implementación es muy directa, ya existiendo capas de Keras especializadas para esto.

Se podría considerar que el modelo mejor entrenado fue la red neuronal convolucional, debido a que está diseñada para capturar patrones locales, lo que le servía para ver bordes y formas específicas. Si bien los visual transformers son poderosos, se debe recordar que la diferencia en desempeño no puede ser del todo significativa para imágenes de baja resolución.

El modelo con los mejores resultados resulto ser el de CNN, ya que, obtuvo mejores predicciones durante la clasificación de las clases conrespecto a las imagenes, puesto que, la precision analizada fue la más alta y por ende, la que mejor se adecua a la base de datos.

Para próximos desarrollos implementando los modelos ViT, se recomendaría realizar los entrenamientos con conjuntos de imágenes con mayor número de muestras o implementar técnicas más avanzadas de aumento de datos, como añadir ruido, haciendo un reescalado o a través de duplicación de imágenes.

Conclusiones

Como ya mencionamos, en este proyecto analizamos el desempeño de dos modelos de redes neuronales, un visual transformer y una red neuronal convolucional con el conjunto de datos Fashion MNIST. Con unos resultados que marcan una buena diferencia entre estos modelos, siendo que la red convolucional fue la más eficiente con un accuracy de 0.92 y el visual transformer con 0.70. Estos resultados indican que la estructura de la CNN, con su capacidad para captar características locales a través de las capas convolucionales, parece estar mejor adaptada a las tareas de clasificación de imágenes en Fashion-MNIST.

Por otro lado, aunque el visual transformer no alcanzó la misma precisión que la CNN, su innovador enfoque de atención global y su capacidad para manejar

secuencias de datos sugieren que podría tener aplicaciones valiosas en otros contextos o con diferentes conjuntos de datos.

En conclusión, mientras que la CNN demostró ser más efectiva en este estudio específico, el visual transformer sigue siendo una arquitectura prometedora que merece una mayor exploración. Futuros trabajos podrían centrarse en optimizar y adaptar el ViT para mejorar su rendimiento en tareas similares o explorar sus capacidades en dominios diferentes.

Anexo

Link de los códigos realizados.

ViTPrueba.ipynb

Copy of vision_transformer.ipynb

Referencias

Dosovitskiy, A. et al. (2021) An image is worth 16x16 words: Transformers for image recognition at scale, arXiv.org. Available at: https://doi.org/10.48550/arXiv.2010.11929 .

Hybrid vision transformer (VIT Hybrid) (no date) Hybrid Vision Transformer (ViT Hybrid). Available at: https://huggingface.co/docs/transformers/main/model_doc/vit_hybrid .

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. Communications of the ACM, 60(6), 84–90. https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, Ł., & Polosukhin, I. (2017). Attention Is All You Need. https://arxiv.org/pdf/1706.03762

Uygar Kurt. (2023, 29 septiembre). *Implement and Train ViT From Scratch for Image Recognition - PyTorch* [Vídeo]. YouTube. https://www.youtube.com/watch?v=Vonyoz6Yt9c