Instituto Tecnológico de Estudios Superiores de Monterrey

Proyecto de Aprendizaje Profundo: Reporte Final

Diseño de redes neuronales y aprendizaje profundo (Gpo 201)

Docente: Dr. Santiago Enrique Conant Pablos

Diego Armando Mijares Ledezma - **A01722421**

Leonardo De Regil Cárdenas - **A00837118**

Alberto Rodríguez Reyes - **A01383805**

Asgard Andrés Mendoza Flores - **A00572566**

19 de octubre de 2024

Índice

Introducción

En la actualidad, el aprendizaje profundo ha estado revolucionando el campo de la inteligencia artificial, permitiendo avances significativos en la clasificación de imágenes. Uno de los conjuntos de datos que ha ganado popularidad en esta área es Fashion-MNIST, un recurso que proporciona un nuevo estándar para la evaluación de algoritmos de aprendizaje automático. Este conjunto, propuesto por Zalando Research, consta de 60,000 imágenes de entrenamiento y 10,000 de prueba, cada una representando prendas de vestir en escala de grises con un tamaño de 28 x 28 píxeles. Las imágenes están clasificadas en 10 categorías, lo que permite a los investigadores y desarrolladores explorar diversas arquitecturas de redes neuronales.

Fashion-MNIST se presenta como un reemplazo directo del clásico conjunto MNIST, que ha sido ampliamente utilizado para la clasificación de dígitos escritos a mano. Esta transición es significativa, ya que Fashion-MNIST proporciona un conjunto de datos más relevante y desafiante para las aplicaciones modernas de reconocimiento de imágenes. En este proyecto, nos proponemos utilizar redes neuronales convolucionales (CNN) y la arquitectura EfficientNet para clasificar y agrupar las prendas de vestir contenidas en este conjunto de datos. A través de este enfoque, buscamos no solo evaluar la efectividad de estas arquitecturas en tareas de clasificación de imágenes, sino también contribuir al desarrollo y comprensión de técnicas avanzadas en el campo del aprendizaje profundo.

Metodología y Resultados

Red Neuronal Convolucional

Razón de Elección de Modelo

El primer modelo que se decidió implementar para la clasificación del dataset de Fashion MNIST ha sido una red neuronal convolucional, dicha red fungirá como la red vista dentro del programa de la clase para la realización de este proyecto. Las Redes Neuronales Convolucionales son muy utilizadas para la tarea de clasificación de imágenes gracias a su capacidad de determinar patrones relevantes para la clasificación de las imágenes que se le alimentan (LeCun, Bengio, & Haffner, 1998). Dentro de los modelos vistos en clase se notó que pocos estaban diseñados específicamente para el reconocimiento y la clasificación en base a imágenes. Tres características que destacamos al seleccionar este modelo son las siguientes:

1. Detección y ajuste automático de rasgos para la clasificación en las imágenes.

2. Robustez para lidiar con el ruido en las imágenes: Las CNN tienen una gran cantidad de herramientas que les ayuda a mitigar el efecto que el ruido pueda tener en el desempeño del modelo; algunas de las herramientas probadas fueron las capas "dropout" y la normalización en batches.

3. Reputación sobre una gran precisión en tareas de clasificación de imágenes cuando se consigue generar una buena estructura y un buen ajuste de hiper-parámetros.

Pre-procesamiento Realizado para los Datos

El preprocesamiento de los datos en el proyecto se realiza en varias etapas clave para preparar el dataset Fashion-MNIST antes de entrenar los modelos de redes neuronales. La primera parte del preprocesamiento es cargar el conjunto de datos Fashion-MNIST. Este conjunto de datos viene pre dividido en dos subconjuntos: uno de entrenamiento (x_train, y_train) y otro de prueba (x_test, y_test). Cada imagen de prenda

es una matriz de (28, 28) píxeles en escala de grises, y las etiquetas de las prendas se almacenan como valores enteros de 0 a 9, representando las diferentes clases.

Para facilitar el entrenamiento de la red neuronal, se normalizan las imágenes. Esto se hace dividiendo los valores de los píxeles (que originalmente van de 0 a 255) por 255, lo que los escala en el rango [0, 1]. La normalización es importante porque mejora la eficiencia del aprendizaje y facilita que los modelos convergan más rápido.

Finalmente, las etiquetas de clase, que originalmente son enteros (por ejemplo, 0 para "T-shirt/top" y 1 para "Trouser"), se convierten en una codificación "one hot". Esto significa que cada etiqueta se transforma en un vector binario donde solo un elemento es 1, indicando la clase correspondiente, y los demás son 0. Esta codificación es necesaria para que la red neuronal pueda procesar correctamente las salidas durante la fase de entrenamiento y clasificación.

Implementación y Entrenamiento

Modelo LeNet - 5

En clase se revisó esta estructura en particular de una Red Neuronal Convolucional desarrollada por Yann LeCun, Léon Bottou, Yoshua Bengio, y Patrick Haffner en 1998 para el reconocimiento de dígitos escritos a mano. En una actividad de clase se probó la alta eficiencia de esta arquitectura para clasificar las imágenes de la base de datos de Fashion-MNIST por lo que se ha planteado tomar esta arquitectura LeNet-5 como base para retarnos a mejorar su desempeño.

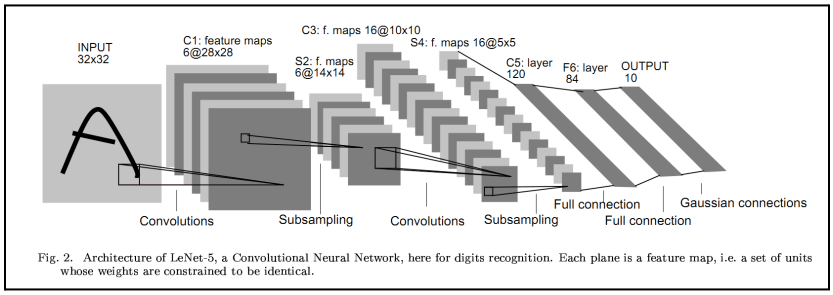


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Arquitectura de LeNet

Se utilizaron las siguientes capas de convolución, muestreo y capas densas para poder crear la red neuronal convolucional con la arquitectura de LeNet-5:

- **Capa de Convolución 1 (C1):** Se aplica una capa convolucional con 6 filtros, cada uno con un tamaño de 5x5 y una función de activación "tanh". Esta capa extrae características iniciales de las imágenes, detectando patrones como bordes o texturas.
- **Capa de Submuestreo (S2):** A continuación, se utiliza una capa de pooling máximo con un tamaño de ventana de 2x2, que reduce las dimensiones de las características extraídas por la convolución.
- **Capa de Convolución 2 (C3):** Se aplica otra capa convolucional con 16 filtros y un tamaño de kernel de 5x5, seguida de una activación "tanh". Esta segunda capa aprende características más complejas de las imágenes, como formas o combinaciones de patrones.
- **Capa de Submuestreo (S4):** Al igual que en la primera etapa, se emplea una capa de pooling máximo que reduce aún más las dimensiones de las características.
- **Capa Completamente Conectada (C5):** Las características resultantes se "aplanan" para ser utilizadas por una capa densa de 120 neuronas con activación "tanh".
- **Capa Completamente Conectada (F6):** Se añade una segunda capa densa con 84 neuronas y activación "tanh".
- **Capa de Salida:** Finalmente, se emplea una capa completamente conectada con 10 neuronas (una por cada clase de prenda) y una función de activación “softmax” para producir las probabilidades de clasificación.

Con esta arquitectura se logró un resultado de 0.89 de exactitud (accuracy) para el modelo. Gracias a las gráficas del coeficiente de pérdida y de accuracy pudimos darnos cuenta que el modelo parece haber quedado estancado, lo cual nos abre una ventana de oportunidad para generar una red que pueda desempeñarse mejor.

Modelo Basado en el Tutorial de Jason Brownlee

Antes de comenzar a generar una nueva red convolucional, se propuso investigar una arquitectura de red diseñada específicamente para el dataset de Fashion - MNIST para ver cuáles eran las diferencias entre ambas arquitecturas. La red que se revisó fue la que se implementa en el tutorial de Jason Brownlee para crear una CNN desde cero. La estructura de la red se muestra a continuación:

- **Capa de Convolución 1 (C1):** La red comienza con una capa convolucional que aplica 32 filtros con un tamaño de kernel de 3x3, utilizando la función de activación "relu". Esta capa tiene como objetivo extraer características iniciales de las imágenes, como bordes y texturas, mientras que el inicializador “he_uniform” se emplea para mejorar la eficiencia del entrenamiento.
- **Capa de Submuestreo (S2):** A continuación, se incluye una capa de pooling promedio (Average Pooling) con una ventana de 2x2. Esta capa reduce la dimensionalidad de las características extraídas, lo que disminuye el número de parámetros.
- **Capa Completamente Conectada (C3):** Después de aplanar las características extraídas, se introduce una capa completamente conectada con 100 neuronas y activación “relu”. Esta capa ayuda a aprender patrones no lineales y combina las características previamente extraídas para la clasificación final.
- **Capa de Salida:** Finalmente, se emplea una capa de salida con 10 neuronas, una por cada clase en el dataset Fashion - MNIST, y la función de activación softmax, que convierte las salidas en probabilidades para la clasificación.

Se puede notar que esta red tiene menos capas en general. Además usa funciones de activación 'relu' esto podría ayudar a hacer la red computacionalmente eficiente y propiciar la activación dispersa de neuronas debido a que aumenta la probabilidad de que el resultado de la activación sea cero, se eliminan los strides en la imágen y al optimizador se le agrega un coeficiente de momento. La función del momento es utilizada para acelerar el descenso del gradiente y puede ayudar a evitar el oscilamiento de la función de pérdida en mínimos locales. A grandes rasgos el momento introduce el concepto de velocidad a la razón de cambio del gradiente. Esta red consiguió una certeza

del 0.88, el modelo obtuvo resultados bastante similares a la red de LeNet-5. En base a estos resultados, se sospecha que el modelo puede mejorar si se incrementa el número de capas de convolución y si se agregan más neuronas en las capas densas.

Modelo Propio CNN

Para este modelo y en base a los resultados obtenidos se decidió incrementar a dos capas convolucionales, una capa de Dropout y una penúltima capa densa con más neuronas. El propósito en general es permitir que el modelo sea capaz de encontrar más patrones dentro de las imágenes que le ayude a identificar mejor cada categoría.

Por otra parte la capa de Dropout mitiga los efectos del overfitting en la red al cambiar los inputs de ciertas unidades a 0 con una frecuencia determinada por el coeficiente asignado de forma que la suma en general de todas las entradas de la capa se vea modificada. La capa Dropout aleatoriamente pone algunas unidades de entrada en 0 durante el entrenamiento para prevenir el sobreajuste. Las entradas no anuladas se escalan para mantener la suma total de las entradas constante. La estructura de la red se muestra a continuación:

- **Capa de Convolución 1 (C1):** La red comienza con una capa convolucional que aplica 32 filtros con un tamaño de kernel de 3x3, utilizando la función de activación "relu". Esta capa está diseñada para extraer características iniciales como bordes y patrones básicos de las imágenes de entrada. Se utiliza el inicializador "he_uniform" para mejorar la eficiencia del entrenamiento y optimizar los pesos de la red.
- **Capa de Submuestreo (S2):** A continuación, se emplea una capa de pooling máximo con una ventana de 2x2. Esta capa reduce la dimensionalidad de las características extraídas.
- **Capa de Convolución 2 (C3):** Se añade una segunda capa convolucional con 64 filtros, también con un tamaño de kernel de 3x3 y activación "relu". Esta capa permite que el modelo aprenda características más complejas y abstractas de las imágenes.

- **Capa de Submuestreo (S4):** Nuevamente, se utiliza una capa de pooling máximo con una ventana de 2x2 para reducir las dimensiones de las características.
- **Capa de Aplanado (C5):** Las características extraídas por las capas anteriores se aplanan para ser utilizadas por las capas densas, lo que convierte los mapas de características 2D en un vector 1D que puede ser procesado por las capas completamente conectadas.
- **Capa de Dropout (D6):** Se incluye una capa de Dropout con una tasa de 0.5, lo que ayuda a prevenir el sobreajuste durante el entrenamiento. Esta capa desactiva aleatoriamente el 50% de las neuronas en cada iteración para mejorar la generalización del modelo.
- **Capa Completamente Conectada (F7):** Se añade una capa completamente conectada con 128 neuronas y activación "relu". Esta capa combina las características extraídas y ayuda a aprender patrones no lineales en los datos.
- **Capa de Salida:** Finalmente, se utiliza una capa de salida con 10 neuronas (una por cada clase en el conjunto de datos) y una función de activación "softmax" que convierte las salidas en probabilidades de clasificación.

Después de haber entrenado el modelo, nos dió una certeza del .90, si bien hubo mejora, comoquiera se intentó generar un mejor modelo, por lo que se hizo una segunda iteración en donde se quiso analizar si mejoraría el resultado teniendo un coeficiente menor en la capa de dropout y si disminuir la tasa de aprendizaje lograría mejorar los resultados.

El resultado fue poco satisfactorio, pues el modelo no logró mejorar con estos cambios dentro de la arquitectura y los hiper parámetros. Se destacó que los resultados en este caso fueron mucho más consistentes, pero ni siquiera los datos atípicos son tan buenos como la iteración pasada del modelo. En base a estas dos iteraciones pasadas, se creó un tercer modelo en donde se probó con utilizar la función de “BatchNormalization“ dentro de la arquitectura de la red. La idea detrás de estas nuevas capas es que después de cada convolución se normalizan los valores generados por cada unidad para ajustarlos a una distribución con media cercana a 0 y con una desviación estándar cercana a 1. Esto evita que los valores de entrada para próximas capas cuenten con una gran variación y

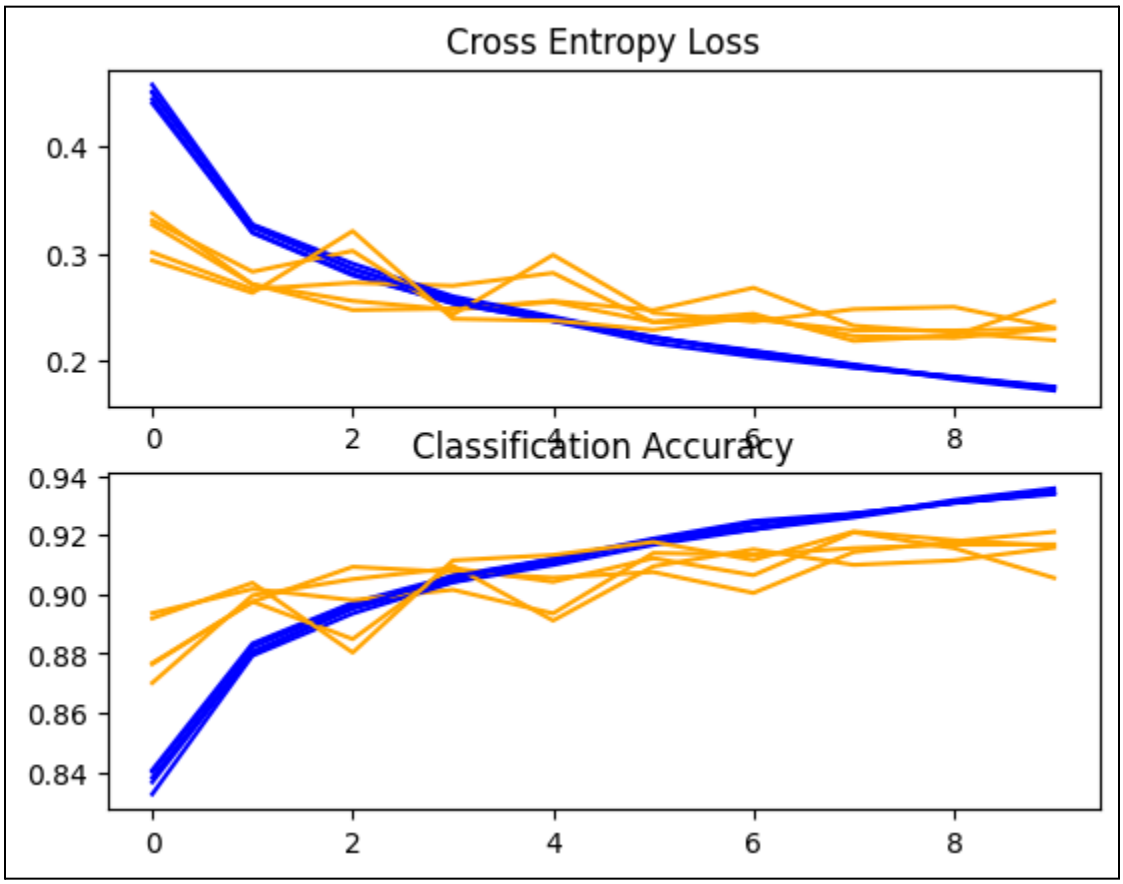
por esto puede ayudar a que las predicciones del modelo sean mejores. La estructura de la red se muestra a continuación:

- **Capa de Convolución 1 (C1):** La red comienza con una capa convolucional que aplica 32 filtros con un tamaño de kernel de 3x3, utilizando la función de activación "relu". Se utiliza el inicializador "he_uniform" para optimizar los pesos iniciales. Esta capa está seguida de una normalización por lotes (Batch Normalization).
- **Capa de Submuestreo (S2):** Se emplea una capa de pooling máximo con una ventana de 2x2 para reducir la dimensionalidad de las características extraídas.
- **Capa de Convolución 2 (C3):** La siguiente capa convolucional tiene 64 filtros con un tamaño de kernel de 3x3 y activación "relu", nuevamente con el inicializador "he_uniform". Esta capa extrae características más complejas, permitiendo que el modelo reconozca patrones más detallados. Se agrega otra capa de normalización por lotes para mejorar la estabilidad durante el entrenamiento.
- **Capa de Submuestreo (S4):** Al igual que en la primera etapa, se utiliza una capa de pooling máximo con una ventana de 2x2 para reducir la dimensión de las características.
- **Capa de Aplanado (C5):** Las características resultantes se aplanan, transformando los mapas de características 2D en un vector 1D que puede ser procesado por las capas densas siguientes.
- **Capa de Dropout (D6):** Se incluye una capa de Dropout con una tasa de 0.4, lo que ayuda a prevenir el sobreajuste al desactivar aleatoriamente un 40% de las neuronas durante el entrenamiento.
- **Capa Completamente Conectada (F7):** Se añade una capa completamente conectada con 128 neuronas, activación "relu" e inicialización "he_uniform". Esta capa es seguida de una normalización por lotes para mantener el entrenamiento estable.
- **Capa de Salida:** Finalmente, la capa de salida tiene 10 neuronas (una por cada clase del dataset) y una función de activación "softmax" que convierte las salidas en probabilidades de clasificación.

Este cambio a la arquitectura de la red mejoró casi en un 2% los resultados en comparación al modelo previo y un 4% en comparación al modelo LeNet-5 y al modelo de Machine Learning Mastery.

Evaluación del Modelo Entrenado

Además de las métricas de desempeño, es crucial mostrar gráficas y tablas para una evaluación visual del modelo. A continuación, se incluyen varias visualizaciones que ilustran cómo evolucionaron las métricas clave, como la pérdida (loss) y la precisión (accuracy) para nuestro modelo propio.

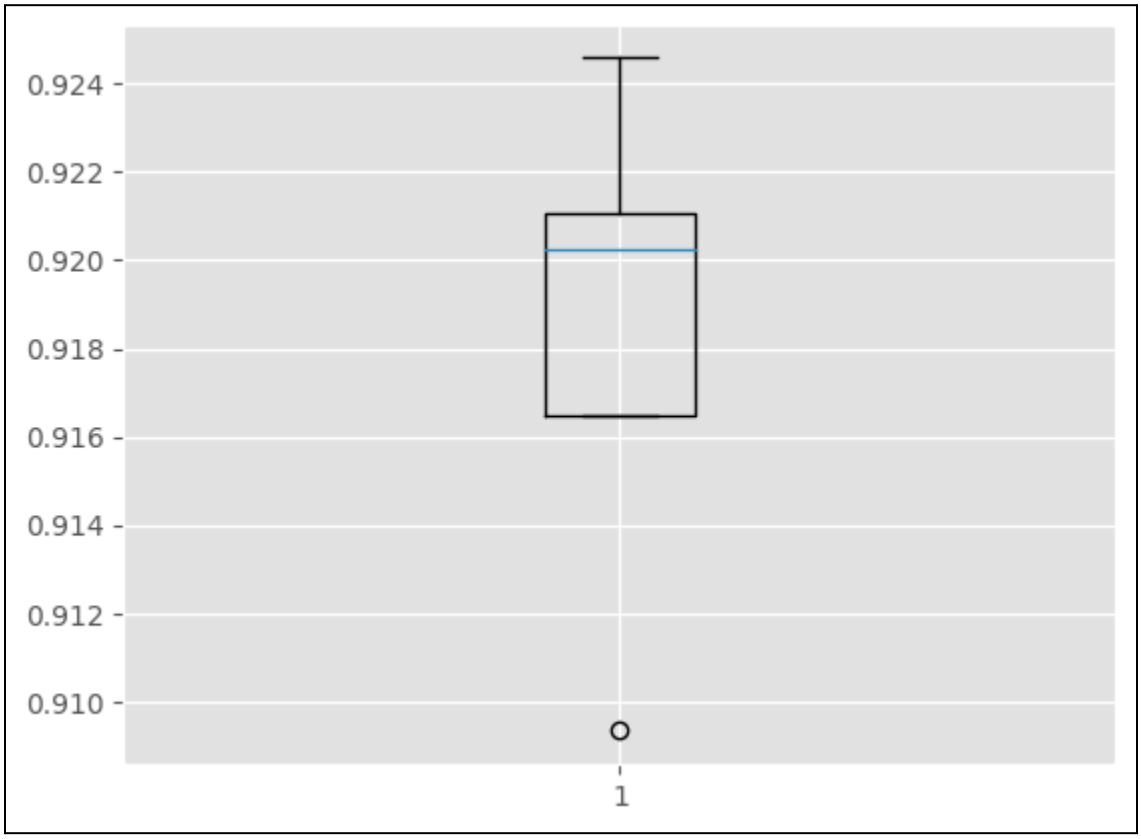


Gráfica de precisión

Tanto la pérdida como la precisión parecen converger a un valor estable a medida que aumenta el número de épocas de entrenamiento. Esto sugiere que el modelo está aprendiendo y no está sobre ajustando los datos de entrenamiento.

La precisión de clasificación alcanza valores bastante altos, lo que indica que el modelo es capaz de distinguir entre las diferentes clases con una alta probabilidad. La pérdida por entropía cruzada disminuye a lo largo del entrenamiento, lo que confirma que el modelo está mejorando en la tarea de clasificación.

La gráfica de abajo muestra un buen desempeño del modelo, con una precisión de entrenamiento que aumenta de manera constante hasta casi el 94% y una precisión de validación que sigue un patrón similar, estabilizándose en torno al 90-92%. No se observan signos de sobreajuste significativo, ya que ambas curvas están cercanas, lo que indica que el modelo generaliza bien los datos.



Gráfica de Boxplot

El análisis de los datos a través del boxplot revela que la mayoría de los valores se encuentran en un rango estrecho, entre 0.916 y 0.920, lo que indica una alta concentración de resultados en esta área. Esta proximidad sugiere que la variable analizada tiene un comportamiento consistente en la mayoría de los casos, con una fuerte agrupación alrededor de la mediana.

A continuación mostramos los diferentes rendimientos de cada uno de los modelos con los que estuvimos experimentando y trabajando.

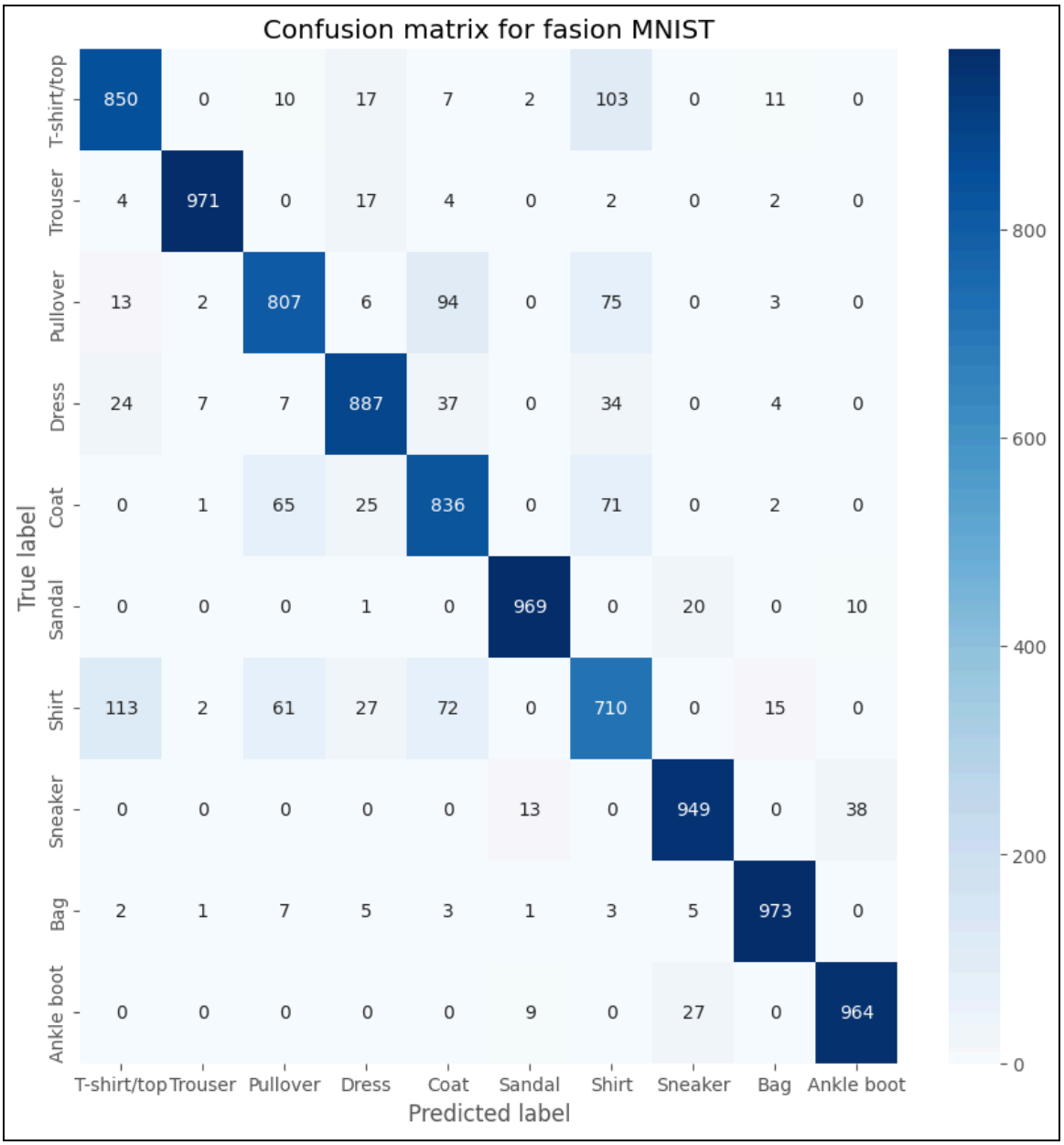
Modelo	**Accuracy**
LeNet - 5	0.89
Tutorial Jason Brownlee	0.88
Propio CNN	0.92

Tabla de Resultados

En base a la tabla, el modelo "Propio" presenta el mejor rendimiento con una precisión (accuracy) del 92%, superando a LeNet-5 (89%) y al tutorial de Jason Brownlee (88%). Esto sugiere que el enfoque o los ajustes realizados en el modelo propio han logrado optimizar su capacidad para realizar predicciones precisas en comparación con los otros modelos evaluados.

Ahora, se muestra una matriz de transición de nuestro modelo propio:

[En siguiente página por tamaño de imágen]

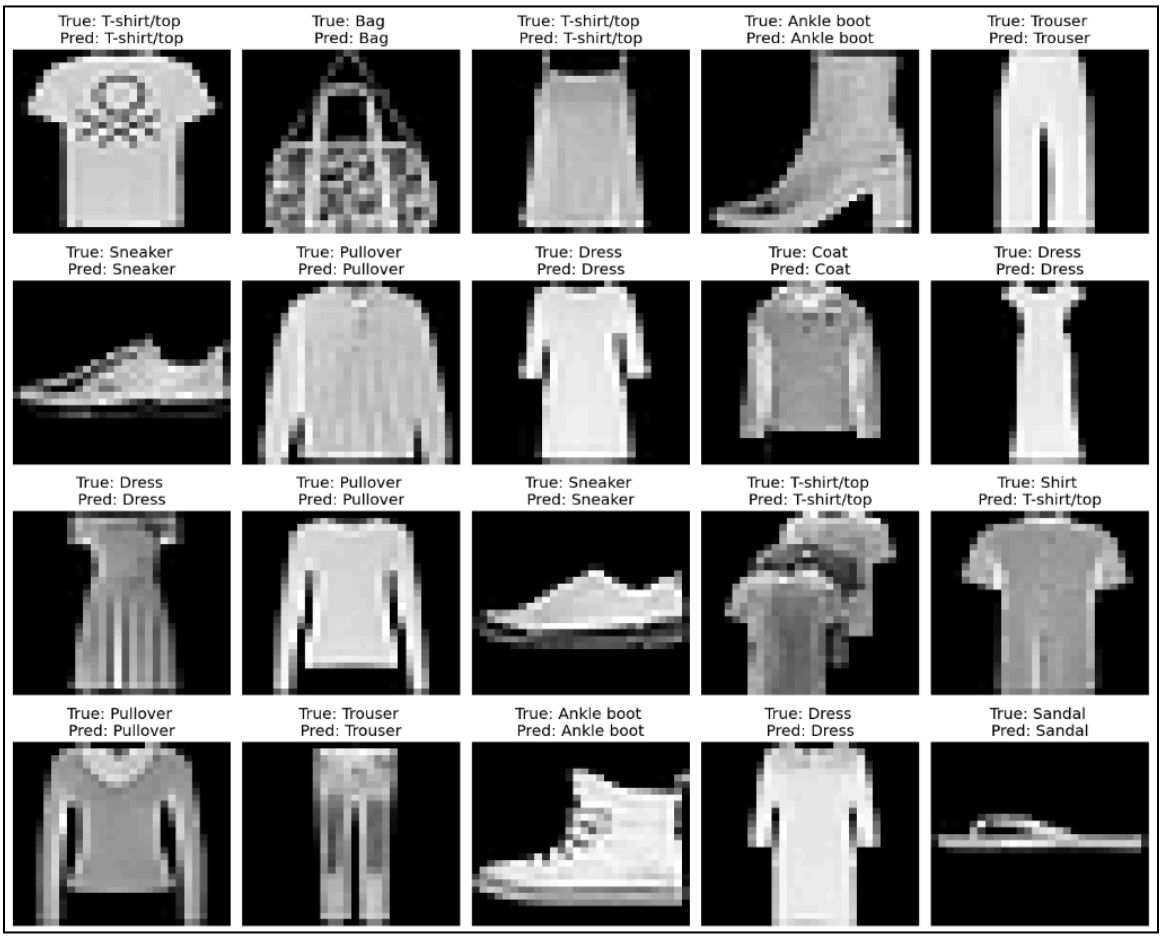


Matriz de confusión

La matriz de confusión ilustra el rendimiento del modelo en diversas clases de prendas. En términos generales, el modelo muestra un buen desempeño en la mayoría de las categorías, con altos valores en la diagonal principal, que indica las predicciones correctas. No obstante, se notan algunos errores en las categorías de "Camiseta/top" y "Camisa," donde el modelo confunde varias instancias de "Camisa" como

"Camiseta/top." Esto podría indicar que estas dos clases son difíciles de distinguir para el modelo. La categoría con mayores problemas parece ser "Camisa," ya que presenta un número significativo de confusiones con otras clases, lo que sugiere que se requieren mejoras específicas para esta categoría.

A continuación mostramos algunos ejemplos del rendimiento del modelo. Podemos ver que efectivamente clasifica muy bien las diferentes imágenes.



Clasificación imágenes ejemplo

Red Neuronal EfficientNet

Razón de selección de modelo

EfficientNet fue seleccionado debido a su arquitectura eficiente y escalable, que equilibra el uso de recursos computacionales y la precisión del modelo. EfficientNet utiliza un enfoque de escalado compuesto, que ajusta uniformemente la profundidad, el ancho y la resolución de entrada del modelo, lo que permite un mejor rendimiento con menos parámetros en comparación con otras arquitecturas. Aunque EfficientNet fue diseñado para tareas de clasificación de imágenes más complejas, su enfoque eficiente lo hace una opción atractiva para datasets más pequeños como Fashion - MNIST. Además, se beneficia del uso de capas convolucionales separables en profundidad, lo que reduce la complejidad computacional manteniendo una alta precisión. EfficientNet también utiliza la activación *swish*, la cual ha demostrado un mejor rendimiento que *ReLU* en algunas redes convolucionales, lo que puede contribuir a una mejora en la convergencia y precisión del modelo. Esta arquitectura ha sido ampliamente evaluada y recomendada para tareas de clasificación de imágenes debido a su balance entre rendimiento y eficiencia (Tan & Le, 2019).

Pre-procesamiento Realizado para los Datos

Las imágenes de Fashion MNIST fueron normalizadas dividiendo sus valores por 255, de forma que los píxeles están en el rango de [0, 1]. Esta normalización es esencial para asegurar que las entradas del modelo tengan una escala uniforme, facilitando el proceso de aprendizaje. Adicionalmente, las imágenes en Fashion MNIST están en escala de grises, por lo que originalmente tienen una forma de (28, 28). Para que sean compatibles con las capas convolucionales del modelo, se reformatean añadiendo una dimensión extra que representa el canal de color, resultando en una forma de (28, 28, 1). Finalmente, para la clasificación, las etiquetas fueron convertidas a una representación categórica mediante la función *to_categorical*, ya que Fashion MNIST es un problema de clasificación multiclase con 10 categorías.

Implementación y Entrenamiento

A continuación se muestra la estructura de la red EfficientNet:

- **Capa de Convolución 1 (C1):** La red comienza con una capa convolucional con 32 filtros y un tamaño de kernel de 3x3, utilizando "swish" como función de activación. Esta capa reduce el tamaño de la imagen de entrada mediante un stride de 2. Se emplea normalización por lotes para estabilizar el entrenamiento y mejorar la eficiencia.
- **Bloque Convolucional Separable en Profundidad (C2):** A continuación, se aplica un bloque de convolución separable en profundidad. Primero, se realiza una convolución en profundidad (depthwise) para cada canal, seguida de una convolución de punto (pointwise) que combina los resultados a través de todos los canales. Se utilizan 64 filtros con un kernel de 3x3 y activación "swish". Este bloque permite que el modelo extraiga características más complejas de las imágenes sin aumentar excesivamente el costo computacional.
- **Bloque Convolucional Separable en Profundidad (C3):** Se añade otro bloque de convolución separable con 128 filtros, donde el stride de 2 reduce aún más la dimensión espacial de las características.
- **Bloque Convolucional Separable en Profundidad (C4):** Con 256 filtros y un stride de 2, este bloque profundiza aún más la capacidad del modelo para reconocer patrones complejos, como combinaciones de formas y texturas.
- **Bloques Adicionales (C5 y C6):** Para aumentar la complejidad del modelo, se añaden más bloques de convolución separable con 256 y 512 filtros, respectivamente. Estos bloques permiten que la red capte características aún más profundas, mejorando su capacidad para clasificar imágenes más complejas.
- **Capa de Aplanado Global (C7):** Después de los bloques convolucionales, se aplica una capa de Global Average Pooling, que reduce las características a un vector de tamaño fijo sin importar las dimensiones de entrada, extrayendo la media de cada mapa de características.

- **Capa Completamente Conectada (F8):** Se añade una capa densa con 128 neuronas y activación "swish", lo que permite que la red combine las características aprendidas y las relaciones no lineales entre ellas.
- **Capa de Salida:** Finalmente, la capa de salida tiene 10 neuronas (una por cada clase en el dataset de Fashion MNIST) con activación "softmax", lo que genera probabilidades de clasificación para cada clase.

El modelo EfficientNet fue modificado específicamente para ajustarse al dataset de Fashion MNIST, un conjunto de imágenes en escala de grises de 28x28 píxeles, que presenta menos complejidad visual que los datasets para los cuales EfficientNet fue diseñado originalmente. Las modificaciones clave son las siguientes:

1. **Tamaño de entrada**:
   EfficientNet normalmente acepta imágenes de mayor resolución. En este caso, el tamaño de entrada fue reducido a (28, 28, 1), que corresponde a las imágenes en escala de grises de Fashion MNIST.
2. **Reducción de filtros**:
   Se redujo el número de filtros en las capas convolucionales para ajustarse a la simplicidad de las imágenes. En lugar de comenzar con un número elevado de filtros (como se haría para imágenes complejas), se empezó con 32 filtros en la primera capa y luego se incrementaron gradualmente a medida que la red profundiza. Esto evita el sobreajuste y mantiene el modelo eficiente.
3. **Data Augmentation**:
   Aunque EfficientNet puede manejar grandes cantidades de datos, Fashion MNIST es un dataset relativamente pequeño. Para evitar el sobreajuste, se aplicó aumento de datos mediante la generación de imágenes con pequeñas rotaciones, traslaciones y zooms. Esto incrementó la diversidad de los ejemplos de entrenamiento, ayudando al modelo a generalizar mejor.
4. **Optimización específica**:
   La activación swish, que es estándar en EfficientNet, fue mantenida debido a sus ventajas demostradas sobre ReLU, mejorando la convergencia y la precisión. Sin embargo, se ajustó el aprendizaje del modelo con el uso de callbacks como

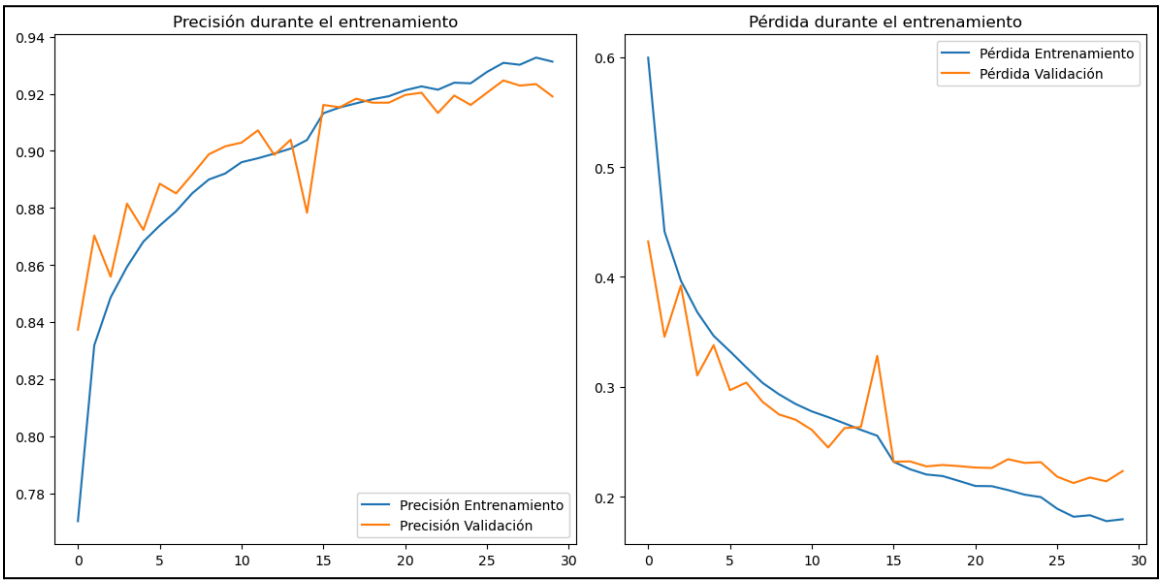
EarlyStopping y ReduceLROnPlateau, lo que permitió una mejor adaptación a los datos, deteniendo el entrenamiento antes de que se produjera el sobreajuste.

5. **Capa de salida personalizada**:

   La capa de salida fue modificada para tener 10 unidades (una para cada clase de Fashion MNIST), con una función de activación Softmax, adecuada para problemas de clasificación multiclase.

Evaluación del Modelo Entrenado

Además de las métricas de desempeño, es crucial mostrar gráficas y tablas para una evaluación visual del modelo. A continuación, se incluyen varias visualizaciones que ilustran cómo evolucionaron las métricas clave, como la pérdida (loss) y la precisión (accuracy), tanto en el conjunto de entrenamiento como en el de validación, a lo largo de las 30 épocas que tomó entrenar EfficientNet.



Gráficas de precisión y de pérdida

La gráfica de la izquierda muestra la evolución de la precisión tanto en el conjunto de entrenamiento como en el de validación a lo largo de las épocas. Se observa que la precisión aumenta, lo que indica que el modelo está aprendiendo. La curva de

validación sigue de cerca la de entrenamiento, lo cual es un buen signo, ya que significa que no hay un overfitting significativo. La precisión en el conjunto de validación alcanza aproximadamente 0.91 al final del entrenamiento.

La gráfica de la derecha presenta la pérdida en los conjuntos de entrenamiento y validación. La pérdida disminuye para ambos conjuntos, lo que refleja que el modelo está mejorando su capacidad para hacer predicciones correctas. Aunque la curva de pérdida de validación fluctúa un poco más, sigue la tendencia de la curva de entrenamiento, sugiriendo una buena generalización del modelo.

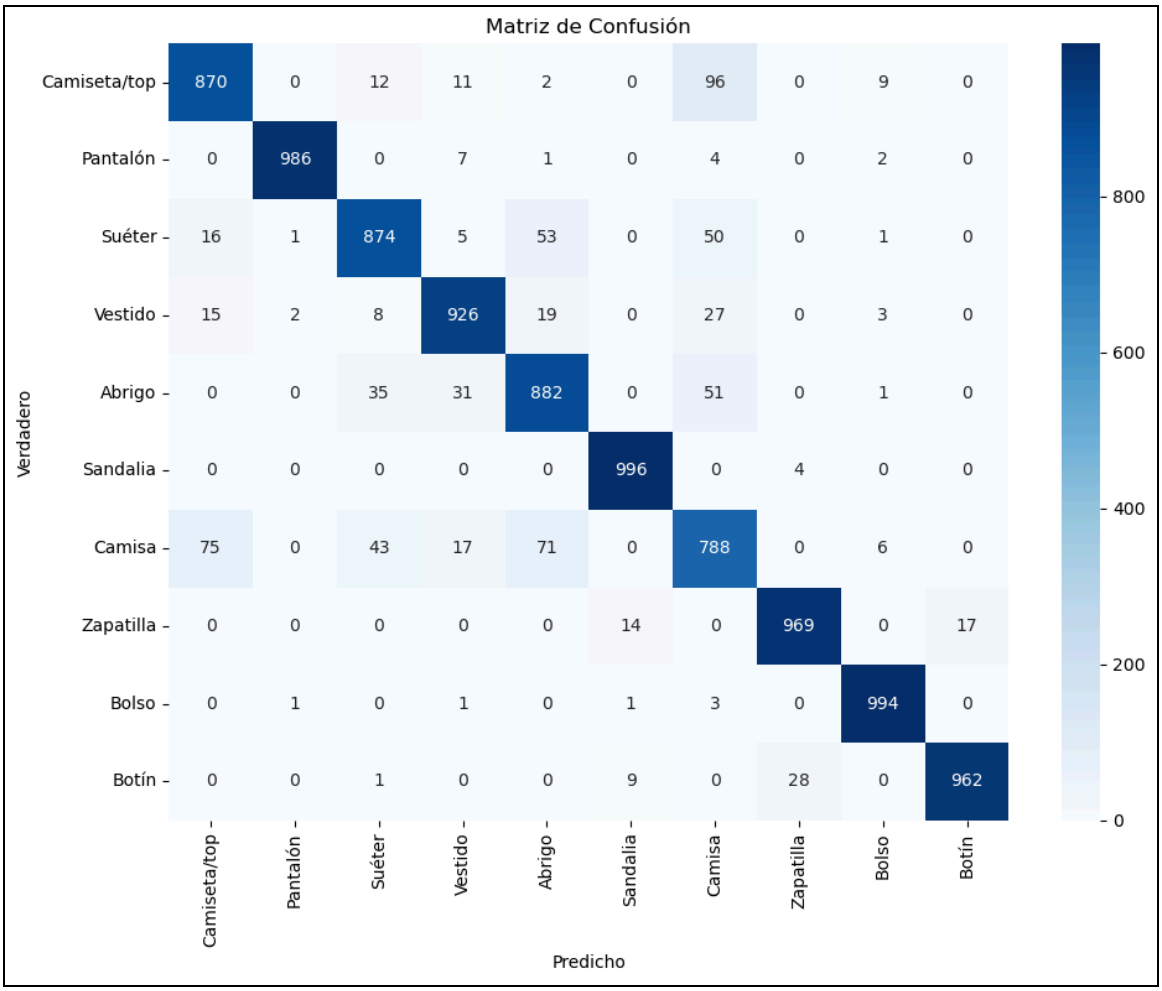
Siguiente, se pueden apreciar 3 de los resultados de las épocas del modelo. La primera, la del medio, y la última.

Epoch	**Accuracy (Train)**	**Loss (Train)**	**Accuracy (Validation)**	**Loss (Validation)**	**Learning Rate**
1	0.7086	0.7791	0.8373	0.4324	0.0010
15	0.9040	0.2554	0.8783	0.3283	0.0010
30	0.9299	0.1803	0.9191	0.2235	2.5000e-04

Tabla de Resultados

Los resultados muestran el progreso de entrenamiento de un modelo en 30 épocas. En la primera época, la exactitud del modelo fue de 70.86%, con una pérdida de 0.7791, mejorando la validez con una exactitud de 83.73% y una pérdida de 0.4324. A medida que avanza el entrenamiento, se observa una mejora continua en la precisión, alcanzando 92.47% en la época 27, con una disminución de la pérdida a 0.1811. El aprendizaje se ajusta en la época 16, donde la tasa de aprendizaje se reduce a la mitad, lo que estabiliza la pérdida y mejora la precisión en las épocas finales.

Siguiente, se puede observar la matriz de confusión:



Matriz de confusión

La matriz de confusión muestra el desempeño del modelo en diferentes clases de ropa. En general, el modelo tiene buen rendimiento para la mayoría de las categorías, con valores altos en la diagonal principal (que representa las predicciones correctas). Sin embargo, se observan algunos errores en categorías como "Camiseta/top" y "Camisa," donde el modelo predice mal varias instancias de "Camisa" como "Camiseta/top". Esto puede indicar que estas dos clases son difíciles de diferenciar para el modelo.

La categoría que parece tener más problemas es la de "Camisa", con un número considerable de confusiones con otras clases, lo cual sugiere que el modelo necesita

mejoras específicas para esta categoría. Otras clases, como "Pantalón" y "Sandalia," tienen una precisión prácticamente perfecta.

Por otro lado, puede ser más útil una visualización de las predicciones **erróneas** junto a las imágenes del dataset:

Predicciones erróneas ejemplo

En la imagen, se pueden observar los tipos de errores que cometió el modelo. No obstante, considerando que el rendimiento del modelo fue bueno, es importante también visualizar ejemplos de predicciones correctas:

Predicciones correctas ejemplo

Se logra apreciar más justamente el buen desempeño del modelo, junto a las probabilidades. La mayoría cerca al 1.0, sin embargo, si se pueden observar otras cerca del 0.69. Pero la gran mayoría tiene un desempeño casi perfecto.

Discusión

Comportamiento para predecir la categoría

CNN

El modelo CNN demostró ser altamente efectivo en el conjunto de datos, logrando una precisión del 92%. Su capacidad para ajustarse rápidamente a las características relevantes de las imágenes sin evidenciar sobreajuste indica que la arquitectura elegida es adecuada para este problema. Durante el entrenamiento, el comportamiento de la pérdida fue consistente y estable, lo que sugiere un aprendizaje eficiente a lo largo de las épocas.

La arquitectura CNN es justificada por su habilidad para captar patrones jerárquicos en los datos, lo que permite un equilibrio óptimo entre precisión y eficiencia computacional. En general, el modelo cumplió con nuestras expectativas, destacándose como una solución robusta para la clasificación de prendas de ropa en el dataset Fashion-MNIST.

EfficientNet

EfficientNet demostró ser un modelo eficiente y robusto en este conjunto de datos. Su capacidad de ajustarse rápidamente a las características relevantes de las imágenes en pocas épocas muestra que el diseño escalable y eficiente del modelo es adecuado para este problema. Sin embargo, durante las primeras épocas, el modelo experimentó una fluctuación en la pérdida de validación, lo cual podría ser un signo de que el aprendizaje inicial fue algo elevado, pero se corrigió con éxito al reducirlo en la época 17.

El uso de EfficientNet está justificado por su arquitectura optimizada, la cual proporciona un equilibrio entre la precisión y la eficiencia computacional, reduciendo el número de parámetros mientras mantiene un desempeño competitivo. Por ende, funcionó como esperábamos.

Identificación del Modelo con Mayor Dificultad de Entrenamiento

De los modelos entrenados, el CNN básico mostró más dificultades en su entrenamiento comparado con EfficientNet. Las razones incluyen:

- **Menor capacidad de generalización**: Aunque las redes convolucionales son poderosas, un CNN básico no tiene la misma capacidad de generalización que EfficientNet. Este último está diseñado para ser eficiente y escalable en cuanto a sus parámetros, lo que le permite aprovechar mejor los datos.
- **Sobreajuste**: El CNN mostró señales de sobreajuste más tempranas, donde la pérdida de entrenamiento continuaba disminuyendo mientras la de validación comenzaba a aumentar, indicando que el modelo no estaba generalizando bien. Se fue corrigiendo esto, mientras que el EfficientNet nunca demostró este problema.
- **Menos robustez a variaciones en los datos**: Al no estar preentrenado ni optimizado para ajustarse a los datos, el CNN luchaba más para obtener buenos resultados consistentes sin sobreentrenar.

Identificación del Modelo con Mejor Rendimiento en Entrenamiento

El modelo EfficientNet superó al CNN en términos de rendimiento, alcanzando una precisión de entrenamiento de 94.75% en comparación con la precisión promedio del CNN, que fue de 90.33% (±0.642). Esta diferencia significativa sugiere que EfficientNet es más efectivo en la captura de patrones en los datos de entrenamiento.

Las razones por las cuales EfficientNet se desempeñó mejor incluyen su arquitectura optimizada, basada en el principio de escalado compuesto, que equilibra el ancho, la profundidad y la resolución de la red de manera eficiente. Esto permite que el modelo aprenda características más complejas sin aumentar excesivamente el número de parámetros. Además, EfficientNet generalmente se pre-entrena en conjuntos de datos grandes como ImageNet, lo que le permite beneficiarse de un conocimiento previo

valioso. Su menor sobrecarga computacional y la capacidad de incorporar técnicas de regularización de manera más efectiva también contribuyen a su mejor rendimiento, ayudando a prevenir el sobreajuste en conjuntos de datos más pequeños.

Desempeño del Modelo: Evaluación Comparativa Basada en Métricas Seleccionadas

Al evaluar el rendimiento de los modelos, se observa que el modelo EfficientNet alcanza una precisión promedio de 92.8%, superando la precisión media del modelo CNN, que es de 91.835%. Este resultado sugiere que EfficientNet ofrece un mejor rendimiento general en la clasificación de imágenes. Además, la precisión del modelo CNN muestra variaciones menores, con un rango de precisión entre 89.872% y 91.835%, mientras que EfficientNet mantiene una tendencia más consistente hacia la mejora de la precisión a lo largo de las épocas de entrenamiento. Por lo tanto, se concluye que, en términos de precisión, EfficientNet es superior al modelo CNN en este conjunto de datos.

En términos de complejidad, se ha observado que el modelo CNN tiene una arquitectura más sencilla, lo que puede ser beneficioso en situaciones donde se requiere un entrenamiento más rápido y una menor necesidad de recursos computacionales. Sin embargo, esta simplicidad puede limitar su capacidad para capturar características más complejas de las imágenes en comparación con EfficientNet, que, aunque es más complejo, muestra un rendimiento superior en tareas de clasificación. En resumen, mientras que el modelo CNN puede ser adecuado para aplicaciones menos exigentes, EfficientNet representa una opción más robusta y efectiva para desafíos de clasificación de imágenes que requieren mayor precisión.

Observaciones Sobre Tiempos y Recursos Requeridos.

CNN

La red neuronal CNN mostró un buen rendimiento, con un tiempo aproximado de 2100 segundos (35 minutos). La memoria utilizada por la CNN se sitúa en un rango de entre 20 a 30 MB. Esto sugiere que, aunque la CNN es competitiva en términos de velocidad, puede requerir un poco más de recursos en comparación con EfficientNet, el cual se evalúa siguiente.

EfficientNet

La red neuronal EfficientNet mostró un tiempo de ejecución de 2623.94 segundos (43 minutos), lo que indica un rendimiento considerablemente eficiente pero tardado. Durante este proceso, se utilizó una memoria de 19.95 MB, reflejando su diseño optimizado que permite un uso reducido de recursos sin comprometer la precisión. Estos resultados destacan la efectividad pero tardanza de EfficientNet en la tarea de clasificación de imágenes, ofreciendo una solución escalable y eficiente para aplicaciones de aprendizaje profundo, dependiendo del tiempo disponible.

Estos hallazgos subrayan la importancia de elegir la arquitectura adecuada según los requisitos de tiempo y memoria en aplicaciones de aprendizaje profundo, adaptándose a las limitaciones del entorno de implementación.

Conclusiones

Posibles mejoras de cada modelo y en general.

CNN

En conclusión, a pesar de haber realizado un grid search exhaustivo en busca de mejoras en el modelo CNN para clasificar las prendas del dataset Fashion-MNIST, no se encontraron configuraciones que optimizaran su rendimiento más allá de lo alcanzado. Esto sugiere que el modelo podría estar operando cerca de su límite de capacidad actual.

Es posible que cualquier mejora adicional resida en configuraciones más remotas o en enfoques alternativos, lo que implica que futuras investigaciones podrían enfocarse en explorar arquitecturas más complejas, técnicas de regularización avanzadas o la incorporación de datos adicionales para potenciar el desempeño del modelo.

EfficientNet

Conforme al ajuste de hiperparámetros, se pudiera experimentar con técnicas de búsqueda como grid search o random search para encontrar la combinación óptima de hiperparámetros, incluyendo el learning rate, el tamaño del lote (batch size) y las funciones de activación.

Para la regularización, se pudieran implementar técnicas adicionales de regularización como dropout y L2 regularization para mitigar el sobreajuste. Así mismo, aunque sí se utilizó data augmentation, se pudiera incrementar la variabilidad en el conjunto de datos de entrenamiento mediante técnicas más robustas, como mayor variabilidad de imágenes a través de métodos manuales, para no depender de la calidad del proceso de librerías de data augmentation. Adicionalmente, el código utiliza early stopping, sin embargo, este en ciertas corridas si se activaba y en otras no, por lo que pudiera ser útil reforzarlo de manera más rígida.

Por último, se puede considerar el uso de técnicas de transfer learning para aprovechar modelos preentrenados en conjuntos de datos más grandes y diversos, como ImageNet, y luego afinar esos modelos para el conjunto de datos específico.

General

En conclusión, a pesar de no haber encontrado mejoras significativas tras un exhaustivo grid search en el modelo CNN para el dataset Fashion-MNIST, se identifican varias áreas para potenciales optimizaciones. El modelo podría estar operando cerca de su límite de capacidad actual, pero futuras investigaciones podrían explorar configuraciones más remotas o enfoques alternativos. Se sugiere experimentar con técnicas de búsqueda de hiper parámetros, como grid search o random search, para optimizar parámetros clave como el learning rate y el tamaño del lote.

Además, la implementación de técnicas adicionales de regularización, como dropout y L2 regularization, podría ayudar a mitigar el sobreajuste. Aunque se aplicó data augmentation, aumentar la variabilidad del conjunto de datos mediante métodos más robustos podría mejorar aún más el rendimiento. También se recomienda reforzar el uso de early stopping para asegurar su efectividad en todas las corridas. Por último, considerar el uso de transfer learning aprovechando modelos pre-entrenados en conjuntos de datos más amplios, como ImageNet, podría permitir afinar el modelo para el conjunto específico y mejorar su desempeño general.

Aprendizajes del Proyecto

Se resalta la importancia de la arquitectura de los modelos. Se aprendió que la selección del modelo adecuado es crucial. EfficientNet, gracias a su diseño optimizado, permite obtener mejores resultados con menos recursos comparado con arquitecturas más simples como CNN.

Así mismo, la relevancia del ajuste de hiper parámetros nota ser de gran importancia. El ajuste de estos, como el learning rate, tuvo un impacto significativo en el

rendimiento del modelo. Aprender a manejar estos ajustes es esencial para mejorar el desempeño del modelo.

Finalmente, se destaca la evaluación continua. La importancia de monitorear continuamente las métricas de desempeño y hacer ajustes sobre la marcha para mejorar los resultados es indudablemente el aprendizaje más importante del proyecto.

Lo más disfrutable del proyecto

El aspecto más disfrutable fue ver cómo las diferentes arquitecturas de redes neuronales afectan el rendimiento del modelo, especialmente al comparar un modelo más simple como un CNN con un modelo optimizado y avanzado como EfficientNet. Ver el impacto de los ajustes en tiempo real y cómo la precisión mejoraba fue gratificante.

Lo menos disfrutable del proyecto

Una de las partes más tediosas fue ajustar los hiper parámetros manualmente y esperar los tiempos de entrenamiento para obtener los resultados. Además, los problemas con el sobreajuste en el CNN fueron frustrantes, ya que requerían una atención constante para evitar que el modelo se desviara en su entrenamiento.

Bibliografía

LeCun, Y., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278-2324. https://doi.org/10.1109/5.726791

Tan, M., & Le, Q. V. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. In International Conference on Machine Learning (pp. 6105-6114). PMLR.