Índice

I. Introducción

En la era digital actual, la capacidad de procesar y analizar grandes cantidades de datos está revolucionando numerosos campos, como la industria de la moda. En este contexto, el análisis de imágenes de artículos de vestir, haciendo uso de técnicas avanzadas de aprendizaje profundo, es una herramienta esencial.

El conjunto de datos Fashion-MNIST, proporcionado por Zalando Research, surge como un recurso valioso para la comunidad de Inteligencia Artificial. Diseñado como un reemplazo del conjunto de datos MNIST original, que contiene dígitos escritos a mano, Fashion-MNIST presenta imágenes en escala de grises de 28x28 de diferentes artículos de vestir, cada una forma parte de una de las 10 clases. Aunque la estructura y el tamaño de los datos son similares a los del MNIST original, el contenido, centrado en la moda, presenta desafíos distintos y oportunidades para la exploración y el análisis.

Las Redes Neuronales Convolucionales (CNN) han demostrado ser particularmente efectivas en tareas de visión computacional, superando a los modelos de aprendizaje automático tradicionales. Estas arquitecturas, que pertenecen al aprendizaje profundo, se caracterizan por su capacidad para reconocer patrones complejos y, por lo tanto, requieren grandes conjuntos de datos para entrenamiento. Fashion-MNIST, con su conjunto de entrenamiento de 60,000 ejemplos y su conjunto de prueba de 10,000, distribuidos de forma uniforme entre las 10 clases, es ideal para trabajar con redes profundas y explorar su potencial en la clasificación y agrupación de prendas de vestir.

Más allá de la simple clasificación, el agrupamiento de prendas de vestir puede ofrecer insights valiosos para la industria de la moda, como identificar las tendencias más recientes. Además, en un mundo cada vez más conectado, donde las redes sociales juegan un papel crucial en la forma de pensar, de actuar y de vestir de sus usuarios, es esencial considerar aplicar estos modelos avanzados a datos generados por redes sociales como Twitter, Facebook e Instagram.

En este proyecto, se busca explorar y aplicar modelos de aprendizaje profundo y aprendizaje automático al conjunto de datos Fashion-MNIST, con el objetivo de clasificar prendas de vestir. A través de este análisis, se busca obtener una comprensión más profunda de las capacidades y limitaciones de estas técnicas de aprendizaje profundo y su aplicabilidad en el mundo real.

II. Modelo visto en clase: CNN

La elección de utilizar Redes Neuronales Convolucionales (CNNs) para el conjunto de datos Fashion-MNIST se fundamenta en la naturaleza y las capacidades de estas redes. En el dataset de Fashion-MNIST, las imagenes contienen píxeles individuales que pueden agruparse para formar bordes, que a su vez pueden formar patrones y texturas, y estos patrones pueden finalmente definir distintas prendas de vestir. Las CNNs, con su arquitectura de capas ocultas, incluyendo capas convolucionales, son especialmente aptas para capturar estos patrones.

El componente principal de una CNN es el operador de convolución. Esta operación permite funciona como aplicar filtros, que detectan características específicas en una imagen. A medida que se avanza en la profundidad de la red, las características aprendidas se vuelven más abstractas, permitiendo la identificación de elementos más complejos en las prendas.

Otra ventaja inherente a las CNNs es la reducción de la dimensionalidad. Mediante operaciones como el max-pooling, las CNNs reducen la dimensión del resultado de aplicar los kernels o filtros, mientras conservan los patrones más esenciales. Esta reducción no solo optimiza la computación, sino que también mejora la invarianza de la red a pequeñas variaciones en las imágenes. Además, haciendo uso de capas de dropout, las CNN se vuelven menos propensas al sobreajuste.

Finalmente, se seleccionó esta arquitectura para Fashion-MNIST debido a su éxito en tareas de clasificación de imágenes. Las CNNs, como se menciona en Krizhevsky et al. (2012) han mostrado un rendimiento excepcional en conjuntos de datos de imágenes, lo que justifica su aplicabilidad y eficacia en la clasificación de prendas de vestir a partir de imágenes.

1. Preparación de datos.

- ➔ Carga de datos.

  Se cargan los datos de entrenamiento y prueba del conjunto de datos Fashion-MNIST.

- ➔ Verificación de dimensiones.

  Tras la carga, se verifica la estructura de los datos, asegurando que las dimensiones sean las esperadas.

- ➔ Verificación de nulos.

  Se verifica la presencia de valores nulos en los conjuntos de entrenamiento y prueba, tanto en las imágenes como en las etiquetas. Esto es esencial para garantizar que no haya datos faltantes que puedan afectar el entrenamiento.

➔ Redimensionar.

Tenemos imágenes como vectores 1D, cada uno con 784 píxeles. Antes de introducir los datos a la CNN, debemos redimensionar los datos a matrices 3D de (28x28x1).

➔ Escalado.

Las imágenes se escalan dividiendo por 255 para que todos los valores estén en el rango [0,1]. Esto ayuda a que el modelo converja más rápido y evita problemas numéricos durante el entrenamiento.

➔ Codificación one-hot.

Las etiquetas se convierten con la codificación one-hot para que puedan ser usadas con una función de pérdida de entropía cruzada categórica en el entrenamiento.

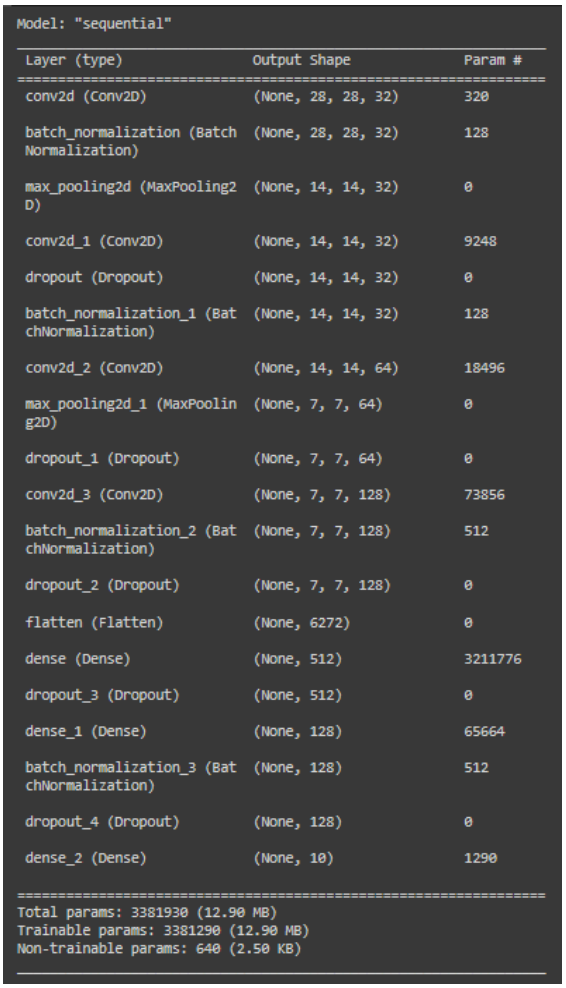
2. Implementación y entrenamiento.

➔ Arquitectura.

Se utiliza una Red Neuronal Convolucional (CNN) que incluye varias capas convolucionales, capas de pooling, capas de normalización por batch, capas de dropout para regularización y capas densas al final.

Las capas convolucionales detectan características locales, desde bordes simples hasta patrones complejos, construyendo una representación jerárquica de la imagen a medida que se avanza en profundidad. Las capas de pooling reducen la dimensionalidad espacial, minimizando así la cantidad de parámetros y cálculos, lo que ayuda a prevenir el sobreajuste y mejora la eficiencia computacional. La normalización por batch equilibra las activaciones de las neuronas, acelerando el proceso de entrenamiento y mejorando la estabilidad de la red. Las capas de dropout, por su parte, añaden un mecanismo de regularización al apagar aleatoriamente ciertas neuronas, lo que evita el sobreajuste. Por último, las capas densas al final juntan estas características y facilitan la clasificación.

La función de activación que se utilizó fue ReLU (Rectified Linear Unit), ReLU activa una neurona sólo si la entrada es positiva; de lo contrario, la neurona se queda inactiva. Esto introduce una propiedad de no linealidad en la red, lo que permite que la red aprenda representaciones más complejas y no lineales de los datos.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 32)        320

 batch_normalization (Batch  (None, 28, 28, 32)        128
 Normalization)

 max_pooling2d (MaxPooling2  (None, 14, 14, 32)        0
 D)

 conv2d_1 (Conv2D)           (None, 14, 14, 32)        9248

 dropout (Dropout)           (None, 14, 14, 32)        0

 batch_normalization_1 (Bat  (None, 14, 14, 32)        128
 chNormalization)

 conv2d_2 (Conv2D)           (None, 14, 14, 64)        18496

 max_pooling2d_1 (MaxPoolin  (None, 7, 7, 64)          0
 g2D)

 dropout_1 (Dropout)         (None, 7, 7, 64)          0

 conv2d_3 (Conv2D)           (None, 7, 7, 128)         73856

 batch_normalization_2 (Bat  (None, 7, 7, 128)         512
 chNormalization)

 dropout_2 (Dropout)         (None, 7, 7, 128)         0

 flatten (Flatten)           (None, 6272)              0

 dense (Dense)               (None, 512)               3211776

 dropout_3 (Dropout)         (None, 512)               0

 dense_1 (Dense)             (None, 128)               65664

 batch_normalization_3 (Bat  (None, 128)               512
 chNormalization)

 dropout_4 (Dropout)         (None, 128)               0

 dense_2 (Dense)             (None, 10)                1290

=================================================================
Total params: 3381930 (12.90 MB)
Trainable params: 3381290 (12.90 MB)
Non-trainable params: 640 (2.50 KB)
_________________________________________________________________
```

1. conv2d (Conv2D):

- Tipo: Convolucional
- Activación: ReLU
- Forma de salida: (None, 28, 28, 32)
- Número de parámetros: (3 * 3 * 1 + 1) * 32 = 320
- Realiza operaciones de convolución para extraer características espaciales. Como la entrada tiene dimensiones (28, 28, 1), especificar el padding a same, se le aplica relleno para que al aplicar el filtro éste encaje completamente dentro de la imagen. Lo que mantiene la dimensión de salida, solo en vez de 1 canal son 32 por los 32 filtros aplicados.

2. batch_normalization (Batch Normalization):

- Tipo: Normalización por batch
- Forma de salida: (None, 28, 28, 32)
- Número de parámetros: (32 * 4) = 128

- Normaliza las activaciones de la capa anterior, acelerando el entrenamiento y proporcionando cierta regularización.

3. max_pooling2d (MaxPooling2D):

- Tipo: Max Pooling
- Forma de salida: (None, 14, 14, 32)
- Número de parámetros: 0
- Reduce la dimensionalidad espacial tomando el valor máximo en ventanas de un tamaño específico. Se reducen a la mitad, de 28 a 14.

4. conv2d_1 (Conv2D):

- Tipo: Convolucional
- Activación: ReLU
- Forma de salida: (None, 14, 14, 32)
- Número de parámetros: (3 * 3 * 32 + 1) * 32 = 9248
- Realiza operaciones de convolución para extraer características espaciales. Como la entrada tiene dimensiones (14, 14, 32), al especificar el padding a same, se le aplica relleno para que al aplicar el filtro éste encaje completamente dentro de la imagen. Lo que mantiene la dimensión de salida.

5. dropout (Dropout):

- Tipo: Dropout
- Forma de salida: (None, 14, 14, 32)
- Número de parámetros: 0
- Aplica regularización al apagar aleatoriamente ciertas neuronas durante el entrenamiento para prevenir sobreajuste.
- Cada unidad tiene una probabilidad del 25% de ser apagada.

6. batch_normalization_1 (Batch Normalization):

- Tipo: Normalización por batch
- Forma de salida: (None, 14, 14, 32)
- Número de parámetros: (32 * 4) = 128

7. conv2d_2 (Conv2D):

- Tipo: Convolucional
- Activación: ReLU
- Forma de salida: (None, 14, 14, 64)
- Número de parámetros: (3 * 3 * 32 + 1) * 64 = 18496
- Padding: same. Se le aplica relleno para que al aplicar el filtro éste encaje completamente dentro de la imagen. Lo que mantiene la dimensión de salida. Solo los canales cambian según la cantidad de filtros.

8. max_pooling2d_1 (MaxPooling2D):

- Tipo: Max Pooling
- Forma de salida: (None, 7, 7, 64)
- Número de parámetros: 0
- Reduce la dimensionalidad espacial tomando el valor máximo en ventanas de un tamaño específico. Se reducen a la mitad, de 14 a 6.

9. dropout_1 (Dropout):

- Tipo: Dropout
- Forma de salida: (None, 7, 7, 64)
- Número de parámetros: 0
- Aplica regularización al apagar aleatoriamente ciertas neuronas durante el entrenamiento para prevenir sobreajuste.
- Cada unidad tiene una probabilidad del 25% de ser apagada.

10. conv2d_3 (Conv2D):

- Tipo: Convolucional
- Activación: ReLU
- Forma de salida: (None, 7, 7, 128)
- Número de parámetros: (3 * 3 * 64 + 1) * 128 = 73856
- Padding: same. Se le aplica relleno para que al aplicar el filtro éste encaje completamente dentro de la imagen. Lo que mantiene la dimensión de salida. Solo los canales cambian según la cantidad de filtros.

11. batch_normalization_2 (Batch Normalization):

- Tipo: Normalización por batch
- Forma de salida: (None, 7, 7, 128)
- Número de parámetros: (128 * 4) = 512

12. dropout_2 (Dropout):

- Tipo: Dropout
- Forma de salida: (None, 7, 7, 128)
- Número de parámetros: 0
- Aplica regularización al apagar aleatoriamente ciertas neuronas durante el entrenamiento para prevenir sobreajuste.
- Cada unidad tiene una probabilidad del 25% de ser apagada.

13. flatten(Flatten):

- Tipo: Aplanamiento
- Forma de salida: (None, 6272)
- Número de parámetros: 0

- Esta capa aplana su entrada, convirtiéndola en un vector unidimensional. Esto se hace típicamente antes de conectar la entrada con capas completamente conectadas.

14. dense (Dense):

- Tipo: Completamente Conectada
- Activación: ReLU
- Forma de salida: (None, 512)
- Número de parámetros: (6272 + 1) * 512 = 3211776
- Esta capa tiene neuronas que están completamente conectadas a todas las activaciones en la capa anterior.

15. dropout_3 (Dropout):

- Tipo: Dropout
- Forma de salida: (None, 512)
- Número de parámetros: 0
- Aplica regularización al apagar aleatoriamente ciertas neuronas durante el entrenamiento para prevenir sobreajuste.
- Cada unidad tiene una probabilidad del 50% de ser apagada.

16. dense_1 (Dense):

- Tipo: Completamente Conectada
- Activación: ReLU
- Forma de salida: (None, 128)
- Número de parámetros: (512 + 1) * 128= 65664
- Esta capa tiene neuronas que están completamente conectadas a todas las activaciones en la capa anterior.

17. batch_normalization_3 (Batch Normalization):

- Tipo: Normalización por batch
- Forma de salida: (None, 128)
- Número de parámetros: (128 * 4) = 512

18. dropout_4 (Dropout):

- Tipo: Dropout
- Forma de salida: (None, 128)
- Número de parámetros: 0
- Aplica regularización al apagar aleatoriamente ciertas neuronas durante el entrenamiento para prevenir sobreajuste
- Cada unidad tiene una probabilidad del 50% de ser apagada.

19. dense_2 (Dense):

- Tipo: Completamente Conectada

- Activación: Softmax
- Forma de salida: (None, 10)
- Número de parámetros: (128 + 1) * 10= 1290
- Esta capa tiene neuronas que están completamente conectadas a todas las activaciones en la capa anterior.
- Esta capa aplica una función de activación a su entrada. En el contexto de una capa final de clasificación, se suele usar softmax para obtener probabilidades de pertenencia a cada clase.
- Tiene 10 neuronas (una para cada clase de prenda).

El total de parámetros del modelo es 3381930, de los cuales 338129 son entrenables. La memoria que requiere este modelo es de aproximadamente 12.9 MB.

➔ Optimizador.

Se utiliza el optimizador Adam con una tasa de aprendizaje inicial de 0.001. Adam es conocido por su eficiencia y capacidad para adaptar la tasa de aprendizaje durante el entrenamiento.

➔ Entrenamiento.

Se utiliza un generador de datos para alimentar el modelo con imágenes y etiquetas durante el entrenamiento. También se utiliza un programador de tasa de aprendizaje para reducir la tasa de aprendizaje a medida que avanzan las épocas. Se usaron 40 épocas y batches con un tamaño de 128.
Se usa también Data Augmentation,que es una técnica utilizada en el procesamiento de imágenes para aumentar la cantidad de datos de entrenamiento al aplicar diversas transformaciones a las imágenes originales. Al entrenar un modelo con estas imágenes transformadas, este pueda generalizar mejor y ser más robusto ante variaciones en los datos no vistos anteriormente.

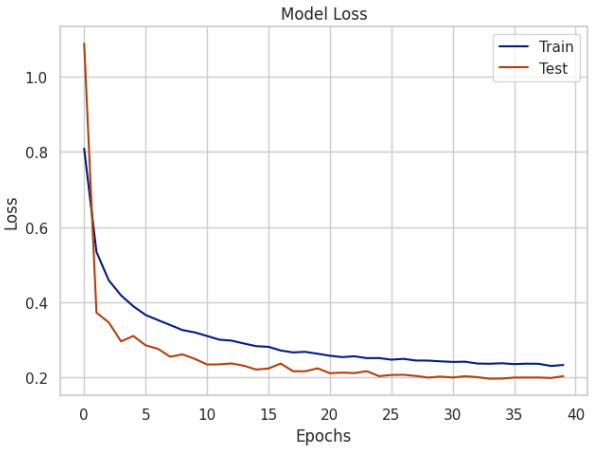
3. Evaluación del modelo entrenado.

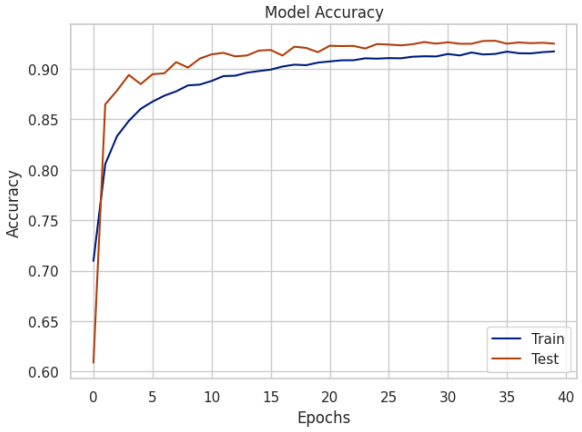
Se evalúa el modelo en el conjunto de datos de prueba para obtener la pérdida y la precisión. Se visualiza la pérdida y la precisión del modelo a lo largo de las épocas para el conjunto de entrenamiento y prueba.

El valor de pérdida es una métrica que evalúa la calidad del modelo. En este caso, la pérdida es de aproximadamente 0.1952. Cuanto menor sea este valor, mejor, ya que indica que el modelo está haciendo predicciones más precisas en el conjunto de datos de prueba.

La precisión mide la proporción de predicciones correctas realizadas por el modelo en el conjunto de datos de prueba. Un valor de precisión del 0.93 o 93% significa

que el modelo ha acertado en aproximadamente el 93% de las muestras de prueba. Esto también es un resultado muy bueno.





Una pérdida baja y una alta precisión sugieren que el modelo es capaz de hacer predicciones precisas en el conjunto de datos de prueba.

En la gráfica de pérdida, la línea azul muestra la pérdida del modelo en los datos de entrenamiento. La pérdida comienza muy alta, pero disminuye bruscamente en las primeras épocas y luego se estabiliza gradualmente. La línea roja representa la pérdida en los datos de prueba. También disminuye inicialmente, pero luego permanece relativamente estable durante las épocas restantes.

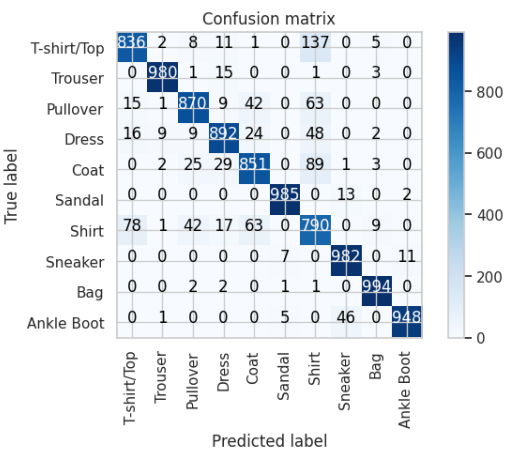
Dado que ambas líneas siguen el mismo comportamiento y no hay una gran separación entre ellas, se puede decir que hay un buen ajuste en el modelo.

En el gráfico de precisión, la línea azul representa la precisión del modelo en los datos de entrenamiento. Comienza más baja y aumenta constantemente a lo largo de las épocas. La línea roja representa la precisión del modelo en los datos de prueba. Inicialmente, tiene un aumento significativo y luego permanece relativamente estable.

La precisión en ambos conjuntos de datos de entrenamiento y prueba parece estabilizarse después de un cierto número de épocas. Esto sugiere que el modelo ha aprendido la mayoría de los patrones en los datos en ese punto.

En general, el modelo parece estar funcionando bien tanto en los datos de entrenamiento como en los de prueba según los gráficos.

	precision	recall	f1-score	support
T-shirt/Top	0.91	0.83	0.87	1000
Trouser	0.99	0.98	0.99	1000
Pullover	0.90	0.88	0.89	1000
Dress	0.93	0.91	0.92	1000
Coat	0.90	0.88	0.89	1000
Sandal	0.99	0.98	0.99	1000
Shirt	0.73	0.84	0.78	1000
Sneaker	0.95	0.99	0.97	1000
Bag	0.99	0.99	0.99	1000
Ankle Boot	0.99	0.95	0.97	1000
accuracy			0.93	10000
macro avg	0.93	0.93	0.93	10000
weighted avg	0.93	0.93	0.93	10000



Precisión: Indica el número de resultados positivos correctos dividido por el número de todos los resultados positivos (verdaderos positivos + falsos positivos).

Recall: Mide el número de resultados positivos correctos dividido por el número de todas las muestras que deberían haber sido identificadas como positivas. Un alto recall significa que la mayoría de las muestras positivas se clasificaron correctamente.

Puntuación F1: Proporciona un equilibrio entre precisión y recall, especialmente si sus puntuaciones son significativamente diferentes.

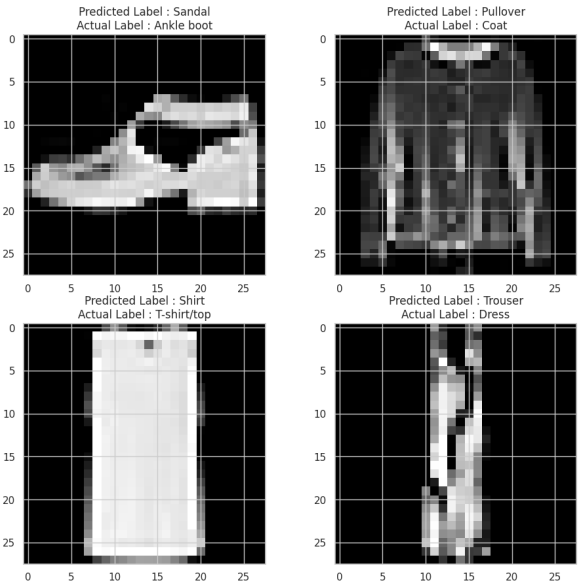
Se puede observar que Trouser, Sandal y Bag tienen una puntuación f1 casi perfecta, lo que indica que es fácil para el modelo distinguir pantalones, sandalias y bolsas de otras clases. Por otro lado, Shirt tiene las puntuaciones más bajas entre las clases, lo que sugiere que las camisas podrían estar siendo confundidas con otras clases, posiblemente camisetas/tops o suéteres. Esto se puede observar en la matriz de confusión, pues Shirt tiene el valor de la diagonal más bajo (por eso está en un azul más claro), y tiene valores un poco más grandes para otras clases, es decir, clasificaciones erróneas.

Otras clases como Sneaker y Ankle Boot tienen puntuaciones altas, lo que sugiere que el modelo funciona bien distinguiendo estas clases.

Macro avg, este es el promedio no ponderado de la métrica en todas las clases. Para precisión, recall y puntuación F1, es 0.93, lo que sugiere un rendimiento consistente en diferentes clases.

Weighted avg, este es el promedio de la métrica ponderado por el número de instancias verdaderas para cada etiqueta. Dado que las muestras se distribuyen por igual entre las clases, es igual al promedio macro en este caso.

En conclusión, el modelo muestra un rendimiento sólido en el conjunto de datos con una exactitud del 93%. Sin embargo, hay margen para mejorar, como la clase de Shirt. Posibles pasos podrían ser recopilar ajustar el modelo específicamente para las clases con peor rendimiento.



Se puede observar como los errores de clasificación del modelo son en imágenes que incluso para el ojo humano puede resultar difícil clasificar.

4. Conclusiones del método con la base de patrones utilizada.

Las CNN son adecuadas para tareas de clasificación de imágenes debido a su capacidad para aprender patrones. La normalización por batch y el dropout ayudan a mejorar la generalización del modelo y a evitar el sobreajuste. La adaptación de la tasa de aprendizaje durante el entrenamiento ayudó a mejorar la convergencia y el rendimiento del modelo.

Las CNN son usadas comúnmente para el data set MNIST con números manuscritos, Fashion-MNIST es un conjunto de datos que puede ser más desafiante que el MNIST original debido a la similitud entre las clases de prendas de vestir y la variabilidad dentro de las clases. Por esto, fue esencial preprocesar y escalar las imágenes para garantizar un entrenamiento eficiente.

El modelo entrenado es capaz de predecir con precisión de 93% la categoría de una prenda de vestir dada una imagen de entrada. La precisión y la pérdida visualizadas en las gráficas proporcionan una idea clara del rendimiento del modelo, que es muy bueno, a lo largo del tiempo y su capacidad para generalizar en datos no vistos.

Una CNN simple es buena para clasificar imágenes, pero para clasificar imágenes de prendas de vestir como las del conjunto de datos Fashion-MNIST se necesita una preparación adecuada de los datos, una arquitectura bien diseñada y técnicas de regularización y optimización.

Dentro de la arquitectura se usaron 3 capas convolucionales, con la idea de que la primera detectara patrones simples como líneas y curvas, la segunda patrones un poco más complejos como contornos y la última fuera capaz de generalizar una abstracción de cada tipo de prenda. Para mejorar el rendimiento del modelo no sólo se aplicaron dichos filtros, sino que se implementaron más de 10 capas con diferentes propósitos. También, durante el entrenamiento se ajustaron parámetros como la tasa de aprendizaje para mejorar la convergencia del modelo y se implementó Data Augmentation con el propósito de que el modelo fuera capaz de reconocer los patrones sin que factores como rotación o brillo afecten.

II. Modelo investigado: Combinación de CNN y RBF

La eficacia de las Redes Neuronales Convolucionales (CNN) en el manejo de datos de imágenes y su capacidad para aprender patrones locales y estructuras en los datos está bien documentada en la literatura. Como se menciona en Krizhevsky et al. (2012), las CNN son especialmente aptas para detectar patrones jerárquicos y locales en datos de imágenes, lo que las convierte en una elección popular para tareas de clasificación de imágenes.

Por otro lado, las RBF son conocidas por su capacidad para aprender una representación no lineal de los datos, lo que puede ser beneficioso en tareas donde la relación entre las características y las etiquetas no es lineal según Bishop (2006). Además, las redes RBF pueden servir como una capa de clasificación efectiva que mapea las características extraídas a las etiquetas de clase.

La combinación de CNN y RBF puede ser vista como una arquitectura híbrida donde la CNN se utiliza para extraer características relevantes de las imágenes y la RBF se utiliza para clasificar estas características en diferentes clases, como lo menciona Jafarpisheh et al. (2021). La CNN puede aprender características discriminativas de los datos mientras que la RBF puede mapear estas características a un espacio donde la clasificación se vuelve más sencilla.

La combinación también puede ser beneficiosa en tareas donde se requiere aprender una métrica de distancia o similaridad entre las imágenes. Como menciona Amirian & Schwenker (2020) las redes RBF pueden ser adaptadas para aprender una métrica de distancia que puede ser utilizada para comparar las características extraídas por la CNN.

También hacen referencia a que las redes RBF pueden mejorar la interpretabilidad del modelo proporcionando una función de activación que tiene un significado geométrico claro, lo cual puede ser útil en tareas donde la interpretabilidad es importante.

Algunas arquitecturas híbridas también han mostrado ser robustas frente a ciertas condiciones según Li et al. (2021), como la presencia de ejemplos adversarios, lo

que puede ser una ventaja en aplicaciones donde la seguridad y la robustez son críticas.

En el contexto de Fashion MNIST, que es un conjunto de datos de imágenes de artículos de moda, la combinación de CNN y RBF puede ayudar a aprender representaciones efectivas de los datos y clasificar los artículos en diferentes categorías de manera precisa e interpretable.

1. Preparación de datos.

- ➔ Carga de datos.

  Se cargan los datos de entrenamiento y prueba del conjunto de datos Fashion-MNIST.

- ➔ Verificación de dimensiones.

  Tras la carga, se verifica la estructura de los datos, asegurando que las dimensiones sean las esperadas.

- ➔ Verificación de nulos.

  Se verifica la presencia de valores nulos en los conjuntos de entrenamiento y prueba, tanto en las imágenes como en las etiquetas. Esto es esencial para garantizar que no haya datos faltantes que puedan afectar el entrenamiento.

- ➔ Redimensionar.

  Tenemos imágenes como vectores 1D, cada uno con 784 píxeles. Antes de introducir los datos a la CNN, debemos redimensionar los datos a matrices 3D de (28x28x1).

- ➔ Escalado.

  Las imágenes se escalan dividiendo por 255 para que todos los valores estén en el rango [0,1]. Esto ayuda a que el modelo converja más rápido y evita problemas numéricos durante el entrenamiento.

- ➔ Codificación one-hot.

  Las etiquetas se convierten con la codificación one-hot para que puedan ser usadas con una función de pérdida de entropía cruzada categórica en el entrenamiento.

2. Implementación y entrenamiento.

- ➔ Arquitectura.

Se implementó una arquitectura de red neuronal convolucional (CNN) complementada con una capa de función de base radial (RBF). La CNN comenzó con una capa convolucional y de pooling para extraer características esenciales de las imágenes. Tras estas capas, se aplanaron los datos para ser procesados por una capa densa o completamente conectada. Se introdujo una capa de dropout para mejorar la regularización y reducir el riesgo de sobreajuste. La adición de la capa RBF es notable, ya que las RBFs pueden capturar patrones no lineales y ofrecer una forma diferente de procesar la información en comparación con las capas convencionales.

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_1 (Conv2D)           (None, 26, 26, 32)        320

 max_pooling2d_1 (MaxPoolin  (None, 13, 13, 32)        0
 g2D)

 flatten_1 (Flatten)         (None, 5408)              0

 dense_1 (Dense)             (None, 64)                346176

 dropout_1 (Dropout)         (None, 64)                0

 rbf_layer_1 (RBFLayer)      (None, 10)                640

 activation (Activation)     (None, 10)                0

=================================================================
Total params: 347136 (1.32 MB)
Trainable params: 347136 (1.32 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

1. conv2d_1 (Conv2D):

- Tipo: Capa Convolucional
- Activación: ReLU
- Forma de salida (Output Shape): (None, 26, 26, 32)
- Número de parámetros: (3 * 3 * 1 + 1) * 32 = 320
- Esta capa realiza operaciones de convolución en la entrada para extraer características espaciales. Como la entrada tiene dimensiones (28, 28, 1), al no especificar el padding, el valor predeterminado es valid, por lo que no se le aplica ningún relleno y sólo se aplica el filtro cuando encaja completamente dentro de la imagen. Lo que reduce la dimensión de salida en comparación con la entrada, sólo los canales pasan de ser 1 a 32, por los 32 filtros aplicados.

2. max_pooling2d_1 (MaxPooling2D):

- Tipo: Max Pooling
- Forma de salida: (None, 13, 13, 32)
- Número de parámetros: 0

- Esta capa reduce la dimensionalidad (volumen) de la entrada al tomar el valor máximo en una ventana de cierto tamaño. Esto ayuda a reducir el costo computacional y también puede ayudar a evitar el sobreajuste.

3. flatten_1 (Flatten):

- Tipo: Aplanamiento
- Forma de salida: (None, 5408)
- Número de parámetros: 0
- Esta capa aplana su entrada, convirtiéndola en un vector unidimensional. Esto se hace típicamente antes de conectar la entrada con capas completamente conectadas.

4. dense_1 (Dense):

- Tipo: Completamente Conectada
- Activación: ReLU
- Forma de salida: (None, 64)
- Número de parámetros: (5408 + 1) * 64 = 346176
- Esta capa tiene neuronas que están completamente conectadas a todas las activaciones en la capa anterior.

5. dropout_1 (Dropout):

- Tipo: Dropout
- Forma de salida: (None, 64)
- Número de parámetros: 0
- Dropout es una técnica de regularización donde aleatoriamente se apagan ciertas neuronas durante el entrenamiento para evitar el sobreajuste.
- Cada unidad tiene una probabilidad del 15 % de ser apagada.

6. rbf_layer_1 (RBFLayer):

- Tipo: Capa de función de base radial (Radial Basis Function, RBF)
- Forma de salida: (None, 10)
- Número de parámetros: 64 * 10 = 640
- Esta capa RBF toma una entrada, calcula su distancia a cada uno de los centros mu, y luego aplica la función de base radial a estas distancias. Es decir, las capas RBF son una forma de transformar la entrada basada en su distancia a ciertos puntos centrales; se utilizan en problemas de clasificación.

  En el código de la capa se incluyen las siguientes funciones y parámetros:

  1. __init__:

- ➢ units: Es el número de neuronas o nodos.
- ➢ gamma: Es un hiperparámetro que determina el ancho de la función RBF. Un valor de gamma más grande hará que la función sea más estrecha, y un valor más pequeño la hará más ancha.

2. build:

- ➢ mu: Es un conjunto de pesos que representan los centros de las funciones RBF. Cada neurona en la capa tiene un centro, y estos centros son parámetros entrenables de la red. Se inicializan usando el inicializador he_uniform.
- ➢ input_shape: Es la forma de los datos de entrada a esta capa.

3. call:

- ➢ En esta función se define la lógica de forward propagation de la capa.
- ➢ diff: Calcula la diferencia entre cada entrada y los centros mu.
- ➢ l2: Calcula la distancia euclidiana al cuadrado entre las entradas y los centros.
- ➢ res: Aplica la función de base radial, es la función de activación RBF.

4. compute_output_shape:

- ➢ Define la salida de la capa.

7. **activation (Activation):**

- Tipo: Activación Softmax
- Forma de salida: (None, 10)
- Número de parámetros: 0
- Esta capa aplica una función de activación a su entrada. En el contexto de una capa final de clasificación, se suele usar softmax para obtener probabilidades de pertenencia a cada clase.
- Tiene 10 neuronas (una para cada clase de prenda).

El total de parámetros del modelo es 347136, de los cuales todos son entrenables. La memoria que requiere este modelo es de aproximadamente 1.32 MB.

➔ Optimizador.

Se utiliza el optimizador Adam con una tasa de aprendizaje inicial de 0.001. Adam es conocido por su eficiencia y capacidad para adaptar la tasa de aprendizaje durante el entrenamiento.

➔ Entrenamiento.

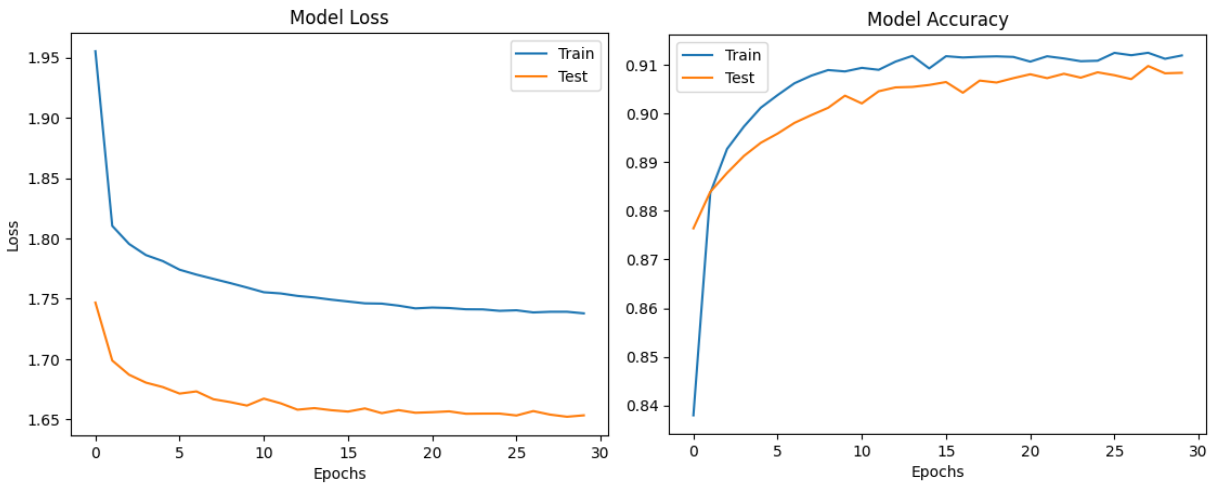
Se alimenta el modelo con imágenes y etiquetas durante el entrenamiento. La tasa de aprendizaje disminuye exponencialmente a medida que avanzan las épocas. Se usaron 30 épocas y batches de tamaño predeterminado de la función fit (32). La idea es permitir que el modelo realice ajustes más grandes al principio del entrenamiento cuando está lejos del óptimo y ajustes más pequeños a medida que se acerca a la convergencia.

3. Evaluación del modelo entrenado.

Una vez entrenado, el modelo se evaluó utilizando el conjunto de datos de prueba. Esta evaluación proporcionó métricas, como la pérdida y la precisión, que indican cuán bien se desempeña el modelo en datos no vistos.

La pérdida es una métrica que mide qué tan bueno es el modelo. En este caso, la pérdida es de aproximadamente 1.6533. Entre más pequeño sea este valor, mejor, ya que indica que el modelo está haciendo mejores predicciones en el conjunto de datos de prueba.

La precisión mide la proporción de predicciones correctas realizadas por el modelo en el conjunto de datos de prueba. Un valor de precisión del 0.91 ó 91% significa que el modelo ha acertado en alrededor del 91% de las muestras de prueba. Esto también es un buen resultado.



Una pérdida baja y una alta precisión sugieren que el modelo es capaz de hacer predicciones precisas en el conjunto de datos de prueba.

En la gráfica de pérdida, la línea azul muestra la pérdida del modelo en los datos de entrenamiento. La pérdida comienza muy alta, pero disminuye bruscamente en las primeras épocas y luego se estabiliza gradualmente. La línea naranja representa la pérdida en los datos de prueba. También disminuye inicialmente, pero luego permanece relativamente estable durante las épocas restantes.

Se observa una brecha entre la pérdida en el conjunto de datos de entrenamiento y la pérdida en el conjunto de datos de prueba, lo que podría ser un signo de sobreajuste. El sobreajuste ocurre cuando el modelo funciona muy bien en los datos de entrenamiento pero no tan bien en nuevos datos no vistos, ya que aprendió de más o memorizó las observaciones, en vez de aprender los patrones generales. Sin embargo, el conjunto de datos que tiene menor pérdida es el de prueba (línea naranja está por debajo de la línea azul), no el de entrenamiento, si fuese el caso contrario pudiera indicar sobreajuste en el modelo.

En el gráfico de precisión, la línea azul representa la precisión del modelo en los datos de entrenamiento. Comienza más baja y aumenta constantemente a lo largo de las épocas. La línea naranja representa la precisión del modelo en los datos de prueba. Inicialmente, tiene un aumento significativo y luego permanece relativamente estable.

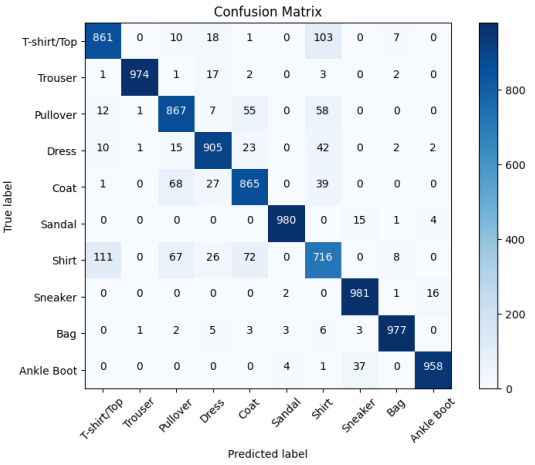
La precisión en ambos conjuntos de datos de entrenamiento y prueba parece estabilizarse después de un cierto número de épocas. Esto sugiere que el modelo ha aprendido la mayoría de los patrones en los datos en ese punto.

En general, el modelo parece estar funcionando bien tanto en los datos de entrenamiento como en los de prueba según los gráficos.

```
313/313 [==============================] - 3s 9ms/step
              precision    recall  f1-score   support

 T-shirt/Top       0.86      0.86      0.86      1000
     Trouser       1.00      0.97      0.99      1000
    Pullover       0.84      0.87      0.85      1000
       Dress       0.90      0.91      0.90      1000
        Coat       0.85      0.86      0.86      1000
      Sandal       0.99      0.98      0.99      1000
       Shirt       0.74      0.72      0.73      1000
     Sneaker       0.95      0.98      0.96      1000
         Bag       0.98      0.98      0.98      1000
  Ankle Boot       0.98      0.96      0.97      1000

    accuracy                           0.91     10000
   macro avg       0.91      0.91      0.91     10000
weighted avg       0.91      0.91      0.91     10000
```



Precisión: Indica el número de resultados positivos correctos dividido por el número de todos los resultados positivos (verdaderos positivos + falsos positivos).

Recall: Mide el número de resultados positivos correctos dividido por el número de todas las muestras que deberían haber sido identificadas como positivas. Un alto recall significa que la mayoría de las muestras positivas se clasificaron correctamente.

Puntuación F1: Proporciona un equilibrio entre precisión y recall, especialmente si sus puntuaciones son significativamente diferentes.

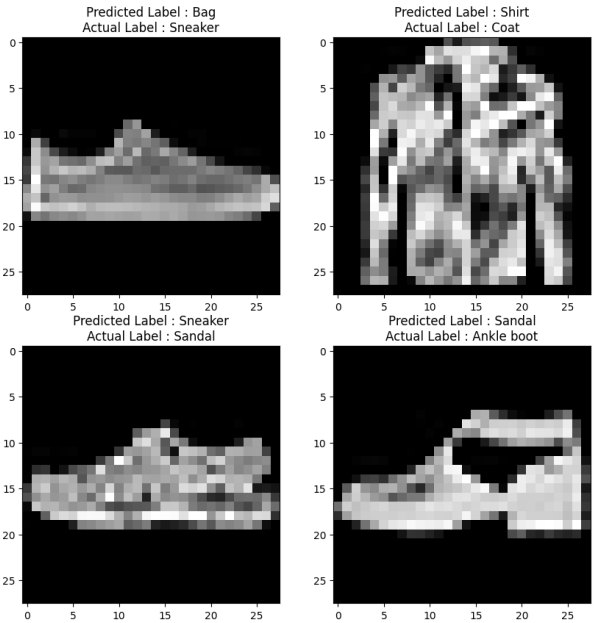
Se puede observar que Trouser tienen una puntuación f1 casi perfecta, lo que indica que es fácil para el modelo distinguir pantalones de otras clases. Mientras que Shirt tiene las puntuaciones más bajas entre las clases, lo que sugiere que las camisas podrían estar siendo confundidas con otras clases, posiblemente camisetas/tops o suéteres. Esto se puede observar en la matriz de confusión, pues Shirt tiene el valor de la diagonal más bajo (por eso está en un azul más claro), y tiene valores un poco más grandes para otras clases, es decir, clasificaciones erróneas.

Otras clases como Sandal, Sneaker, Bag y Ankle Boot tienen puntuaciones altas, lo que sugiere que el modelo funciona bien distinguiendo estas clases.

Macro avg, este es el promedio no ponderado de la métrica en todas las clases. Para precisión, recall y puntuación F1, es 0.91, lo que sugiere un rendimiento consistente en diferentes clases.

Weighted avg, este es el promedio de la métrica ponderado por el número de instancias verdaderas para cada etiqueta. Dado que las muestras se distribuyen por igual entre las clases, es igual al promedio macro en este caso.

En conclusión, el modelo muestra un rendimiento sólido en el conjunto de datos con una exactitud del 91%. Sin embargo, hay margen para mejorar, especialmente para la clase de Shirt. Posibles pasos podrían ser recopilar datos más diversos o aumentar los datos o ajustar el modelo específicamente para las clases con peor rendimiento.



Se puede observar como los errores de clasificación del modelo son en imágenes que incluso para el ojo humano puede resultar difícil clasificar, a excepción del Sneaker, que no parece tanto Bag al ojo humano.

4. Conclusiones del método con la base de patrones utilizada.

La combinación de capas convolucionales con una capa RBF es una aproximación interesante para el conjunto de datos Fashion-MNIST. Las RBFs pueden ofrecer una representación más rica y no lineal de los datos. Esta combinación podría ser especialmente útil para un conjunto de datos como Fashion-MNIST, que tiene una variedad de patrones en las prendas.

Fashion-MNIST es un conjunto de datos desafiante debido a la diversidad de patrones en las prendas de vestir y las diferencias sutiles entre diferentes prendas. La elección de combinar CNNs con RBFs busca aprovechar las fortalezas de ambas técnicas para abordar estos desafíos.

Con esta arquitectura híbrida, me pareció interesante como con una sola convolución, al agregar la capa de RBF se alcanza una precisión de clasificación que requiere de más capas en una arquitectura CNN. Además, en el entrenamiento, desde las primeras 5 épocas el modelo ya aprendió la mayoría de los patrones.

Como se mencionó anteriormente el modelo es bueno con una precisión de 91%, sin embargo, hay prendas como Shirt con las que tiene problemas clasificando.

III. Comparación general de los 2 modelos

Ambos modelos demuestran que las Redes Neuronales Convolucionales (CNN) son efectivas en tareas de clasificación de imágenes, como el dataset Fashion-MNIST. Aunque el modelo CNN se destaca por su precisión ligeramente superior en el conjunto de prueba. Ambos modelos muestran que la preparación adecuada de los datos, el escalado y la regularización son importantes para obtener buenos resultados en tareas de clasificación de imágenes.

Los dos modelos fueron entrenados con éxito, pues en ambos se obtuvieron buenos resultados en las métricas (precisión, recall, puntuación f1). Ambos tienen una precisión bastante alta en el conjunto de prueba. Sin embargo, el modelo CNN tiene una precisión ligeramente superior (93%) en comparación con el modelo híbrido de CNN con RBF (91%). En cuestión de clasificación, ambos son muy buenos clasificando prendas como Trouser, Sandal y Bag, mientras que se equivocan un poco más cuando se trata de Shirts.

El modelo CNN tiene un rendimiento ligeramente mejor en términos de precisión en el conjunto de prueba en comparación con el modelo híbrido de CNN con RBF . Por lo tanto, considerando la precisión, el modelo CNN parece tener un mejor desempeño. Además, la pérdida de la CNN fue mucho menor que la del modelo con RBF, lo que también indica que la CNN es mejor clasificando las clases de prendas. Sin embargo, para lograr ese resultado en la CNN fue necesario implementar una arquitectura más compleja, con muchas más capas que las que tiene el modelo híbrido. Este último, tiene una arquitectura de CNN muy básica, pero con ayuda de un capa RBF puede llegar a resultados equiparables con una arquitectura mucho más compleja en la CNN. Además, el modelo de la CNN también es más robusto al haberse implementado Data Augmentation.

Llegar a la arquitectura final fue complicado con la CNN, pues empecé con una arquitectura simple y agregué capas para mejorar el rendimiento, por lo que involucró correr muchos modelos para ver con cuáles mejoraban las métricas. Además de que correr muchas épocas también requiere un gran esfuerzo computacional. Por otro lado, aunque la arquitectura de CNN del modelo híbrido es bastante sencilla y fue de las primeras que plantee, ajustar la RBF sí fue un reto pues no sólo era una red nueva para mí, por lo que tuve que investigar, sino que no viene incorporada en las librerías como las capas convolucionales, así que diseñar una capa RBF fue algo un poco latoso.

El modelo CNN ocupa más memoria que el híbrido, requiere más megabytes. Según el summary, la CNN requiere una memoria de aproximadamente 12.9 MB, mientras que el CNN con RBF requiere 1.32 MB. La CNN ocupa casi 10 veces más memoria pues tiene una arquitectura mucho más compleja con un total de 19 capas. Considerando esta gran diferencia en términos de recursos requeridos, el modelo con RBF es bastante bueno, pues tiene resultados casi tan buenos como la CNN pero es mucho menos pesado. En cuestión de tiempo, ambos se tardan alrededor de 20 minutos en correr las 30 y 40 épocas respectivamente.

IV. Conclusiones

La CNN, aunque mostró una mayor precisión, también necesitó de una arquitectura más compleja y un mayor espacio de memoria, lo que podría ser un desafío en aplicaciones donde los recursos son limitados. Por otro lado, el modelo CNN con RBF, con una arquitectura más sencilla, logró resultados cercanos en términos de precisión, siendo más eficiente en términos de memoria.

Sin embargo, una de las principales áreas de mejora para ambos modelos podría ser la experimentación con Data Augmentation. Dado que esta técnica mejoró

significativamente el rendimiento de la CNN, es razonable suponer que también podría beneficiar al modelo híbrido. Además, para el modelo híbrido, el ajuste de la capa RBF es crucial para optimizar aún más su rendimiento. También se pueden incorporar técnicas como la Normalización de batches, que fueron útiles en la CNN, no solo agiliza el proceso de entrenamiento sino que mejora la estabilidad. Aunque hay que considerar que al agregar más capas puede llegar a requerir tanta memoria como la CNN.

La CNN es bastante buena, sólo falla un poco en la clase Shirts, al igual que la CNN con RBF, por lo que se puede buscar que mejore el modelo en términos de esa clase. Se puede revisar si hay estilos específicos de camisas que sean muy difíciles de clasificar. También se pueden aplicar técnicas de aumento de datos específicamente a la clase "Shirts". Otra opción puede ser podrías darle más peso durante el entrenamiento, lo que haría que el modelo penalice más los errores en esta clase.

Modelos como Squeeze-and-Excitation Networks o las redes permiten que el modelo se centre en las partes más relevantes de la imagen. Estos pueden ser muy útiles para tareas donde ciertas regiones de la imagen son más informativas que otras, este puede ser el caso de algunas prendas, por lo que se podría intentar incorporar estas arquitecturas en la CNN o la CNN con RBF.

Si el tiempo o los recursos son limitados, considerar modelos más ligeros como MobileNet o EfficientNet, puede ser útil pues están diseñados para ser eficientes en términos de cómputo y memoria.

No hay un modelo único que sea el mejor en todos los aspectos. La elección del modelo debería basarse en el problema específico, los datos disponibles, y las restricciones de recursos. Considerando mis dos modelos, si la precisión es la principal preocupación y los recursos no son un problema, la CNN es la elección. Sin embargo, si se busca una solución más ligera con resultados cercanos, el modelo CNN con RBF es una buena alternativa.

A lo largo del desarrollo de este proyecto, aprendí la profundidad y complejidad de las Redes Neuronales Convolucionales (CNN) y su capacidad para abordar problemas de clasificación de imágenes. Además, descubrí que podía combinar una CNN con la Función de Base Radial (RBF) y cómo esta combinación puede mejorar la precisión y reducir la complejidad del modelo. La experimentación fue crucial para entender cómo mejoraba o empeoraba cada modelo.

Lo que más disfruté fue este proceso de experimentación. Ver cómo pequeños ajustes en la arquitectura o en los hiper parámetros podrían generar cambios significativos en el rendimiento del modelo. Aunque el proceso de experimentación fue enriquecedor, también fue un poco tedioso y me tomó mucho tiempo. Otro aspecto desafiante fue la implementación de la capa RBF, ya que no está implementada en la librerías como tensor flow y pytorch.

Referencias:

Amirian, Mohammadreza & Schwenker, Friedhelm. (2020). Radial Basis Function Networks for Convolutional Neural Networks to Learn Similarity Distance Metric and Improve Interpretability. IEEE Access

Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.

Jafarpisheh, N., Zaferani, E.J., Teshnehlab, M. et al. (2021). A Deep Neural Network Combined with Radial Basis Function for Abnormality Classification. Mobile Netw Appl 26, 2318–2328.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In Advances in Neural Information Processing Systems 25 (NIPS 2012).

Li, J., Gao, J., Li, XX. (2021). Adversarial Examples Defense via Combining Data Transformations and RBF Layers. In: Pham, D.N., Theeramunkong, T., Governatori, G., Liu, F. (eds) PRICAI 2021: Trends in Artificial Intelligence. PRICAI 2021. Lecture Notes in Computer Science(), vol 13032. Springer, Cham.