Situación Problema

Análisis de imágenes de artículos de vestir con aprendizaje profundo

Annette Pamela Ruiz Abreu - A01423595

Tecnológico de Monterrey, Campus Monterrey

TC2035.101 - Diseño de redes neuronales y aprendizaje profundo (Gpo 101)

Dr. Santiago Enrique Conant Pablos

09 de septiembre de 2023

Índice

Análisis de imágenes de artículos de vestir con aprendizaje profundo

En la era actual, la inteligencia artificial se ha convertido en una fuerza transformadora en diversas esferas de la sociedad. La IA se define como la capacidad de las máquinas para realizar tareas que, si fueran ejecutadas por seres humanos, requerirían de inteligencia humana. Esto incluye la capacidad de aprender de los datos y mejorar con la experiencia, lo que ha permitido la creación de sistemas autónomos y aplicaciones innovadoras que revolucionan la forma en que interactuamos con la tecnología. (*¿Qué es la inteligencia artificial (IA)? | IBM*, s. f.) Un aspecto importante de la IA es el aprendizaje automático, el cual se basa en la idea de que las computadoras pueden aprender patrones y realizar tareas específicas a través de la exposición a datos.

Las redes neuronales son métodos de aprendizaje automático que procesan información y extraen patrones en conjuntos de datos grandes y complejos. Las redes neuronales convolucionales (CNN) son arquitecturas de aprendizaje profundo especializadas en el procesamiento de imágenes. Utilizan capas de convolución para detectar características como bordes y texturas, seguidas de capas de agrupación para reducir la dimensionalidad. (*¿Qué son las redes neuronales convolucionales? | IBM*, s. f.) Las redes neuronales recurrentes (RNN) son otro tipo de arquitectura y son únicas en su capacidad para mantener una memoria interna que les permite recordar información previa en una secuencia y utilizarla para influir en la toma de decisiones en momentos posteriores. Las redes neuronales bidireccionales son una variante de la arquitectura de redes de las RNN que extraen datos futuros para mejorar su precisión. (*¿Qué son las redes neuronales recurrentes? | IBM*, s. f.)

En este informe, se abordará el uso y el análisis de este conjunto de datos con el objetivo de desarrollar arquitecturas y modelos de redes neuronales capaces de clasificar y agrupar eficientemente prendas de vestir.

Problema

De acuerdo con Statista, las ventas en línea de prendas de vestir alcanzaron un valor global de 533 billones de dólares en el año 2018, y se espera un crecimiento proyectado a 872 billones para el año 2023 (Forgas, 2023). Este aumento en las compras en línea ha ejercido una presión significativa sobre las infraestructuras de los sitios de compras y ha resaltado la necesidad imperante de mejoras continuas. Para abordar esta creciente demanda y asegurar la retención de clientes en un mercado cada vez más competitivo, se requiere una inversión constante en tecnologías avanzadas como la inteligencia artificial y el aprendizaje profundo, que tienen el potencial de transformar la forma en que los consumidores interactúan y compran prendas en línea.

Justificación

La precisión en la clasificación y agrupación de prendas de vestir es esencial para mejorar la experiencia del cliente en el comercio electrónico y la industria de la moda. La capacidad de brindar recomendaciones personalizadas y una navegación intuitiva en línea depende en gran medida de la capacidad de las máquinas para comprender y organizar las prendas de manera efectiva. Además, la automatización de este proceso permite a las empresas reducir costos y tiempo, optimizando la gestión de inventario y la selección de productos a presentar a los consumidores.

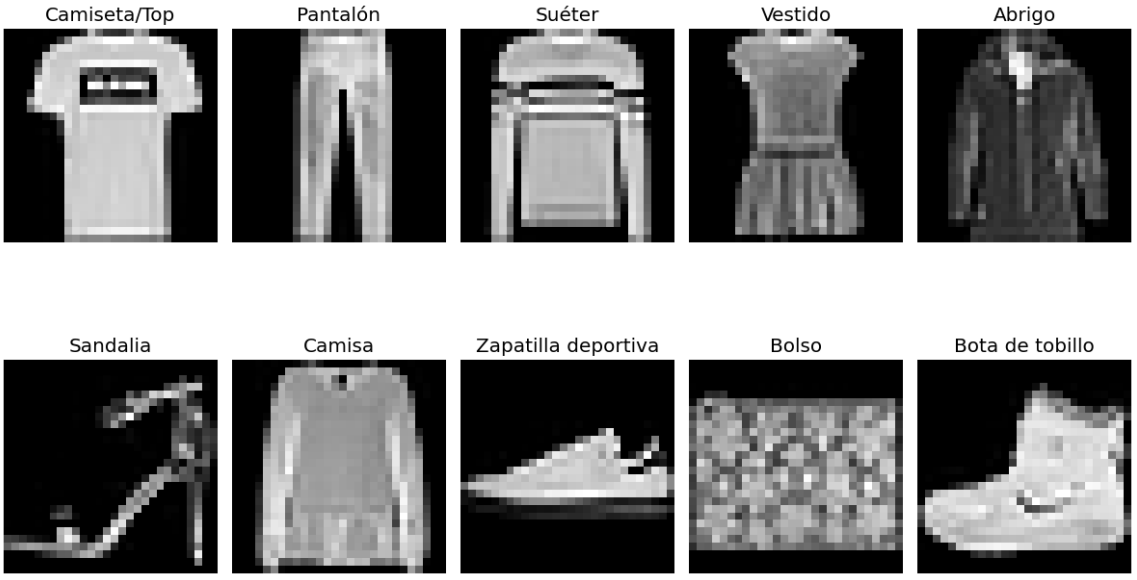
Objetivo

El objetivo de este proyecto es desarrollar y evaluar diversos modelos de redes neuronales capaces de clasificar y agrupar prendas de vestir utilizando el conjunto de datos Fashion-MNIST. Se explorarán diversas arquitecturas y enfoques para lograr una clasificación precisa y una agrupación significativa de las imágenes de prendas.

Antecedentes

Base de datos

El conjunto de datos Fashion-MNIST ha emergido como una herramienta fundamental para explorar y desarrollar modelos de redes neuronales. Proporcionado por Zalando Research, este conjunto contiene imágenes de prendas de vestir que se distribuyen en un conjunto de entrenamiento de 60,000 ejemplos y un conjunto de prueba de 10,000 ejemplos. Cada conjunto es un arreglo que contiene las matrices de las imágenes. Cada imagen, en tonos de grises y con dimensiones de 28x28 píxeles, se encuentra etiquetada dentro de una de las 10 clases disponibles. Las diez clases disponibles son: camiseta, pantalón, suéter, vestido, abrigo, sandalia, camisa, zapatilla deportiva, bolso, bota de tobillo. (*Fashion MNIST*, 2017)



Red neuronal convolucional

Una red neuronal convolucional es un tipo de red neuronal que imita el cortex visual humano. Su objetivo es reconocer diversas características en las entradas, lo cual le permite lograr la identificación de objetos y realizar una forma de "visión". Para lograr esto, la CNN está compuesta por múltiples capas ocultas que presentan una jerarquía en su funcionamiento.

Las primeras capas se dedican a detectar elementos básicos como líneas y curvas, y a medida que se avanza hacia capas más profundas, la red se especializa en el reconocimiento de patrones complejos como rostros o las siluetas de animales. Estas capas profundas utilizan conexiones ponderadas y funciones de activación para aprender representaciones cada vez más abstractas de las características presentes en los datos, lo que les permite tomar decisiones precisas sobre la clasificación de objetos o características en las imágenes de entrada.

- **Capas Convolucionales:** Las capas convolucionales son el componente central de una CNN. Utilizan filtros (kernels) que se aplican de manera deslizante sobre la entrada y realizan operaciones de convolución para extraer características locales. Estos filtros son entrenados para detectar patrones específicos en la imagen, como bordes, texturas o formas.
- **Capas de Pooling:** Después de las capas convolucionales, a menudo se agregan capas de pooling, como la capa de MaxPooling. Estas capas reducen la dimensionalidad de las características extraídas y ayudan a mantener las características más importantes.
- **Capas Fully Connected:** Después de las capas convolucionales y de pooling, típicamente se agregan capas completamente conectadas (dense layers) para realizar la clasificación final. Estas capas se asemejan a las capas en una red neuronal estándar y toman las características aprendidas para realizar la tarea de clasificación.
- **Regularización:** Las CNNs a menudo utilizan técnicas de regularización, como Dropout y BatchNormalization, para mejorar la generalización y prevenir el sobreajuste.
- **Transfer Learning:** Las CNNs preentrenadas en conjuntos de datos masivos, como ImageNet, se utilizan con frecuencia como punto de partida para tareas específicas de

visión por computadora. Esto se conoce como transferencia de aprendizaje y puede ahorrar tiempo y recursos al reutilizar modelos previamente entrenados.

(Na & Na, 2020)

Red neuronal recurrente

Las redes neuronales recurrentes (RNN) son un tipo de arquitectura de redes neuronales artificiales diseñadas para trabajar con datos secuenciales o temporales. A diferencia de las redes neuronales convolucionales, que son especialmente eficaces en el procesamiento de datos espaciales, las RNN están optimizadas para capturar patrones y dependencias en secuencias de datos, lo que las hace esenciales en una amplia gama de aplicaciones. Las RNN se componen de unidades recurrentes que forman conexiones cíclicas entre sí. Esta estructura permite que la información fluya de un paso de tiempo (o elemento de secuencia) al siguiente, lo que les permite capturar la dependencia temporal en los datos.

A diferencia de las redes neuronales feedforward tradicionales, las RNN tienen una memoria interna que les permite mantener y actualizar información a medida que procesan secuencias. Esta memoria a corto plazo se renueva en cada paso de tiempo y es lo que permite a las RNN recordar información reciente. Las variantes avanzadas de RNN, como las LSTM y las GRU, introducen una memoria a largo plazo que les permite capturar dependencias a largo plazo en las secuencias. El entrenamiento de RNN se realiza mediante el algoritmo de retropropagación a través del tiempo (BPTT). Este algoritmo es una extensión de la retropropagación estándar y se utiliza para calcular gradientes en las conexiones recurrentes de la red. Sin embargo, las RNN pueden sufrir de problemas de desvanecimiento o explosión de gradientes durante el entrenamiento, lo que puede dificultar su convergencia. Las variantes LSTM y GRU abordan en gran medida estos problemas.

(Canadas, 2022)

Red neuronal recurrente bidireccional

Las redes neuronales recurrentes bidireccionales (RNN Bidireccionales) son una evolución significativa de las redes neuronales recurrentes estándar. Estas redes son ampliamente utilizadas en tareas relacionadas con el procesamiento de secuencias, como el reconocimiento de voz, el procesamiento de lenguaje natural y la visión por computadora. La principal característica que distingue a las RNN Bidireccionales es su capacidad para capturar información de contexto tanto del pasado como del futuro en una secuencia de datos. A diferencia de las RNN unidireccionales, que procesan los datos en una sola dirección, de izquierda a derecha o viceversa, las RNN Bidireccionales utilizan dos conjuntos de neuronas: una para procesar la secuencia desde el inicio hasta el final y otra para procesarla desde el final hasta el inicio. Esto permite que el modelo capture patrones complejos en las secuencias, ya que puede considerar tanto el contexto previo como el posterior a cada punto en la secuencia. (Corredera, 2023)

Modelación

Exploración de datos

1. Importar las siguientes librerías: pandas, numpy, matplotlib.pyplot, tensorflow, keras.
2. Descargar los datos de la base de datos Fashion MNIST: `fashion_mnist.load_data()`. Al descargar los datos guardarlos en cuatro variables diferentes: x_entrenamiento, y_entrenamiento, x_prueba, y_prueba.
3. Explorar los datos.
    a. Datos de entrenamiento

```
Tipo de dato: <class 'numpy.ndarray'>
Forma: (60000, 28, 28)
Etiquetas: [0 1 2 3 4 5 6 7 8 9]
Valor mínimo: 0
Valor máximo: 255
```

    b. Datos de prueba

```
Tipo de dato: <class 'numpy.ndarray'>
Forma: (10000, 28, 28)
Etiquetas: [0 1 2 3 4 5 6 7 8 9]
Valor mínimo: 0
Valor máximo: 255
```

Preparación de datos

1. Dividir todos los datos entre 255 para que los valores vayan del 0 al 1.
2. Convertir las etiquetas en vectores categóricos. Ejemplo: Si la cateogoría ers 5, el vector resultante es: [0, 0, 0, 0, 0, 1, 0, 0, 0, 0].
3. Convertir el arreglo bidimensional de los valores de las imágenes en un arreglo tridimensional con el valor de 1, pues las imágenes usan una escala de grises. Esto es para poder entrenar al modelo correctamente.

Red neuronal convolucional

Creación del modelo

El modelo que se presentará es el mejor modelo que se encontró después de experimentar con diferentes valores y capas. A continuación se presenta una descripción concisa de una arquitectura de red neuronal convolucional (CNN) implementada utilizando el framework Keras. Esta CNN ha sido diseñada para abordar tareas de clasificación de imágenes, y su estructura se detalla a continuación. Cada capa y configuración se ha seleccionado cuidadosamente para maximizar la eficiencia del modelo en el procesamiento y la clasificación de datos visuales, en particular, imágenes en escala de grises de 28x28 píxeles. A continuación se explica el código capa por capa y la razón por la que se incluyó ese capa o se hizo esa modificación:

1. **layers.Conv2D(32, (3, 3), strides=(1, 1), padding='same', activation='tanh', input_shape=(28, 28, 1))**: Esta es la capa de convolución inicial. Tiene 32 filtros de 3x3, utiliza una función de activación tangente hiperbólica (tanh), y espera una entrada de imágenes de 28x28 píxeles en escala de grises. Se agregó el argumento de "padding" con el valor de "same" para que la salida tenga el mismo tamaño que la entrada. Esto es útil para mantener la información en los bordes de las imágenes cuando se realiza la convolución en una red neuronal convolucional. Además, tras experimentar con las funciones de activación "relu" y "tanh", se usó la función de activación 'tanh' porque mejoraba el resultado. Esta función ofrece ventajas en el entrenamiento de redes neuronales debido a su simetría centrada en cero, lo que significa que su salida es cercana a cero cuando la entrada lo es, ayudando a la estabilidad del entrenamiento. Además, su rango limitado entre -1 y 1 previene gradientes explosivos y puede ser más resistente al problema de desvanecimiento del

gradiente. Los strides determinan la cantidad de desplazamiento que se aplica al filtro a medida que se desplaza sobre la entrada durante la operación de convolución.

2. **layers.BatchNormalization()**: Se utiliza una capa de normalización por lotes después de cada capa de convolución. Esto ayuda a acelerar el entrenamiento y mejorar la estabilidad de la red.
3. **layers.MaxPooling2D((2, 2), strides=(2, 2), padding='valid')**: Capa de pooling máxima que reduce la dimensionalidad de las características al tomar el valor máximo en regiones de 2x2. Es una práctica común poner una capa de pooling después de una capa convolucional.
4. **layers.Dropout(0.2)**: Capa de dropout que apaga el 20% de las neuronas durante el entrenamiento, lo que ayuda a prevenir el sobreajuste.
5. Se repiten capas de convolución, normalización, pooling y dropout para aumentar la profundidad de la red y aprender características más complejas. La cantidad de capas que se utilizaron en el modelo final fueron resultado de experimentación que se discutirá en la discusión de resultados.
6. **layers.Flatten()**: Esta capa aplana las características de la última capa convolucional para que puedan ser alimentadas a las capas densas.
7. **layers.Dense(16, activation='relu')**: Capa densa (totalmente conectada) con 16 neuronas y una función de activación relu.
8. **layers.BatchNormalization()**: Normalización por lotes después de la capa densa. Esta técnica es útil para mejorar la estabilidad y la velocidad de convergencia durante el entrenamiento de redes neuronales, especialmente en arquitecturas profundas.
9. **layers.Dropout(0.5)**: Capa de dropout que apaga el 50% de las neuronas durante el entrenamiento para prevenir el sobreajuste.

10. **layers.Dense(10, activation='softmax')**: Capa de salida con 10 neuronas y una función de activación softmax, que produce probabilidades de pertenencia a cada una de las 10 categorías de clasificación. Se usó la función softmax porque esta devuelve la probabilidad de que el elemento sea de cierta clase, la probabilidad más alta gana.

El modelo final fue el siguiente:

```
model = models.Sequential([
        layers.Conv2D(32,    (3,    3),    strides=(1,    1),    padding='same',
        activation='tanh', input_shape=(28, 28, 1)),
        layers.BatchNormalization(),
        layers.Conv2D(32,    (3,    3),    strides=(1,    1),    padding='same',
        activation='tanh'),
        layers.BatchNormalization(),
        layers.MaxPooling2D((2, 2), strides=(2, 2), padding='valid'),
        layers.Dropout(0.2),
        layers.Conv2D(64,    (3,    3),    strides=(1,    1),    padding='same',
        activation='tanh'),
        layers.BatchNormalization(),
        layers.Conv2D(64,    (3,    3),    strides=(1,    1),    padding='same',
        activation='tanh'),
        layers.BatchNormalization(),
        layers.MaxPooling2D((2, 2), strides=(2, 2), padding='valid'),
        layers.Dropout(0.3),
        layers.Conv2D(128,    (3,    3),    strides=(1,    1),    padding='same',
        activation='tanh'),
        layers.BatchNormalization(),
        layers.Conv2D(128,    (3,    3),    strides=(1,    1),    padding='same',
        activation='tanh'),
        layers.BatchNormalization(),
        layers.MaxPooling2D((2, 2), strides=(2, 2), padding='valid'),
        layers.Dropout(0.4),
        layers.Flatten(),
        layers.Dense(16, activation='relu'),
        layers.BatchNormalization(),
        layers.Dropout(0.5),
        layers.Dense(10, activation='softmax')
])
```

Entrenamiento

Antes de entrenar el modelo, se definieron los hiperparámetros que utilizará.

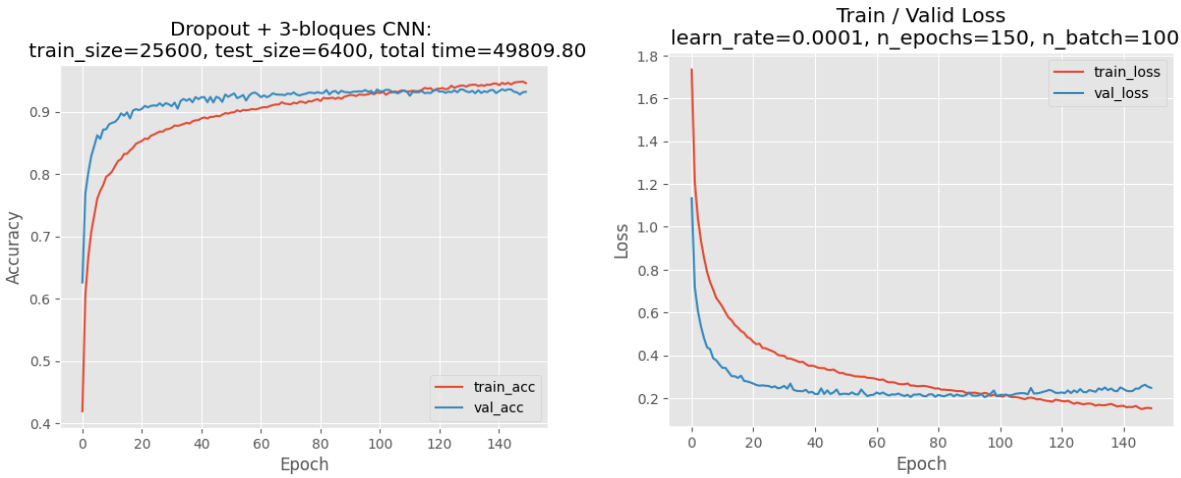
```
épocas de entrenamiento = 150

número de muestras de datos = 100
```

```
tasa de aprendizaje = 0.0001
```

Estos valores de los hiperparámetros se escogieron tras primero experimentar con 50 épocas, 0.001 de tasa de aprendizaje y al analizar el historial de entrenamiento y validación de un modelo de red neuronal. Tras realizar diferentes pruebas, se encontró que al aumentar las épocas y disminuir la tasa de entrenamiento, el *accuracy* del modelo mejoraba 1%.

En el proceso de entrenamiento de una red neuronal, se configuró su comportamiento mediante la selección de un optimizador y una función de pérdida adecuados. El optimizador utilizado fue RMSprop, que se encargó de ajustar los pesos de la red durante el entrenamiento con el objetivo de minimizar la función de pérdida conocida como "categorical_crossentropy." Esta función cuantificó la discrepancia entre las predicciones del modelo y las etiquetas reales, permitiendo así que la red aprendiera a realizar clasificaciones precisas. Se escogió esta métrica porque el entrenamiento de la red era más rápido que cuando se utilizó “adam” y obtuvo mejores resultados.

Además, durante el entrenamiento se evaluó la métrica de precisión (*accuracy*) para monitorear el rendimiento de la red a medida que ajustaba sus parámetros. Este entrenamiento duró 49809.80 segundos; es decir, 13.836 horas. Las gráficas presentadas a continuación muestran el historial de entrenamiento y validación de un modelo de red neuronal. En la primera gráfica, se representan las curvas de precisión (*accuracy*) en el conjunto de entrenamiento y validación a lo largo de las épocas de entrenamiento. Esto proporciona una visión de cómo el rendimiento del modelo evoluciona durante el proceso de entrenamiento. En la segunda gráfica, se muestran las curvas de pérdida (*loss*) tanto en el conjunto de entrenamiento como en el de validación a medida que avanza el entrenamiento. Estas curvas reflejan cómo la función de pérdida del modelo disminuye con el tiempo y si hay signos de sobreajuste o subajuste. Ambas gráficas son útiles para evaluar el rendimiento y la capacidad de generalización del modelo, lo que ayuda a ajustar los hiperparámetros y la arquitectura de la red para lograr un mejor desempeño en tareas de clasificación.



Evaluación

La siguiente tabla proporciona un informe detallado del rendimiento del modelo de clasificación de la categorización de prendas de vestir en diez clases diferentes. Cada fila de la tabla representa una categoría de prenda de vestir, mientras que las columnas presentan diversas métricas de evaluación:

- **Precision:** Esta métrica mide la precisión del modelo en la clasificación de elementos positivos. En este contexto, indica la proporción de prendas de vestir clasificadas correctamente en cada categoría.
- **Recall:** Recall mide la capacidad del modelo para identificar la mayoría de los elementos positivos en cada categoría. Representa la proporción de prendas de vestir relevantes que se clasificaron correctamente.
- **F1-score:** El F1-score es una medida que combina precision y recall en una sola puntuación, lo que proporciona una comprensión equilibrada del rendimiento del modelo.
- **Support:** Esta columna muestra la cantidad de muestras de entrenamiento en cada categoría, lo que indica el número de elementos que pertenecen a esa clase en el conjunto de datos de validación.

- **Accuracy:** La última fila muestra la precisión general del modelo en todas las categorías, lo que significa la proporción de predicciones correctas en todo el conjunto de datos.

	precision	recall	f1-score	support
Camiseta	0.90755008	0.85610465	0.88107704	688
Pantalón	0.99109792	0.98235294	0.98670606	680
Suéter	0.94329897	0.87699681	0.9089404	626
Vestido	0.92455621	0.93562874	0.93005952	668
Abrigo	0.86666667	0.90277778	0.88435374	576
Sandalia	0.98974359	0.98805461	0.98889838	586
Camisa	0.77620397	0.85225505	0.81245367	643
Zapatilla deportiva	0.96456086	0.97507788	0.96979086	642
Bolso	0.99058085	0.98133748	0.9859375	643
Bota de tobillo	0.97975078	0.97067901	0.9751938	648
accuracy			0.931875	0.931875
macro avg	0.93340099	0.9321265	0.9323411	6400
weighted avg	0.93372011	0.931875	0.93237447	6400

Model: "convolucional"		
Layer (type)	**Output Shape**	**Param #**
conv2d_6 (Conv2D)	(None, 28, 28, 32)	320
batch_normalization_7 (Batch Normalization)	(None, 28, 28, 32)	128
conv2d_7 (Conv2D)	(None, 28, 28, 32)	9248
batch_normalization_8 (Batch Normalization)	(None, 28, 28, 32)	128
max_pooling2d_3 (MaxPooling 2D)	(None, 14, 14, 32)	0
dropout_4 (Dropout)	(None, 14, 14, 32)	0
conv2d_8 (Conv2D)	(None, 14, 14, 64)	18496
batch_normalization_9 (Batch Normalization)	(None, 14, 14, 64)	256
conv2d_9 (Conv2D)	(None, 14, 14, 64)	36928

Total params	306906
Trainable params	305978
Non-trainable params	928

Conclusiones

Los resultados del informe de este modelo indican un buen rendimiento del modelo en la clasificación de prendas de vestir, con altas precisiones, recalls y F1-scores en la mayoría de las categorías. La precisión general (*accuracy*) del modelo es del 93%, lo que sugiere que el modelo es capaz de realizar clasificaciones precisas en un amplio espectro de categorías de prendas de vestir. Estos resultados son indicativos de un modelo eficaz y bien entrenado en la tarea de clasificación de imágenes de moda.

Red neuronal recurrente

Creación del modelo

El modelo que se presentará a continuación representa una arquitectura de red neuronal recurrente (RNN) que ha sido cuidadosamente configurada y refinada para abordar la tarea de clasificación de imágenes. Las capas dentro de su arquitectura son las siguientes:

1. **Capa SimpleRNN(256, activation='relu', input_shape=(28, 28), return_sequences=True):** Esta es la primera capa SimpleRNN en la red con 256 neuronas y la función de activación ‘relu’. La opción 'return_sequences=True' indica que esta capa debe devolver secuencias en lugar de un solo valor en cada paso de tiempo. La capa SimpleRNN es una de las formas más básicas de RNN y se utiliza para modelar relaciones en secuencias temporales. A diferencia de las redes neuronales feedforward convencionales, las RNN tienen conexiones recurrentes que les permiten mantener y actualizar un estado oculto o memoria intermedia a medida que procesan cada elemento de una secuencia.
2. **Capa SimpleRNN(128, activation='relu', return_sequences=True):** Esta capa es igual que la anterior, pero tiene menos neuronas. Conforme se fueron agregando capas, se fueron quitando neuronas.

3. **Capa SimpleRNN(64, activation='relu'):** Esta capa es igual que la anterior, pero tiene menos neuronas.
4. **Capa de BatchNormalization()**: Normalización por lotes después de la capa densa. Esta técnica es útil para mejorar la estabilidad y la velocidad de convergencia durante el entrenamiento de redes neuronales, especialmente en arquitecturas profundas.
5. **Capa Dense(10, activation='softmax'):** Esta es la capa de salida de la red. Tiene 10 unidades porque hay 10 número de clases y se usa la función de activación 'softmax' porque esta obtiene probabilidades de pertenencia a cada clase.

El modelo final fue el siguiente:

```
model = tf.keras.models.Sequential([

tf.keras.layers.SimpleRNN(256,activation='relu',     input_shape=(28,     28),
return_sequences=True),

tf.keras.layers.SimpleRNN(128,activation='relu',return_sequences=True),

tf.keras.layers.SimpleRNN(64,activation='relu'),

tf.keras.layers.BatchNormalization()

tf.keras.layers.Dense(10, activation='softmax'),
])
```

Entrenamiento

Antes de entrenar el modelo, se definieron los hiperparámetros que utilizará.

```
épocas de entrenamiento = 25
número de muestras de datos = 65
tasa de aprendizaje = 0.01
```

Estos valores de los hiperparámetros se escogieron tras primero experimentar con hasta 50 épocas, probando muchas tasas de aprendizaje (0.001, 0.01, 0.1) y al analizar el historial de entrenamiento y validación de un modelo de red neuronal. Tras realizar diferentes pruebas y graficar los resultados se encontró un hallazgo muy importante. Después de 40

épocas, la precisión del modelo caía a 0.1 y la pérdida aumentaba. Esto puede ser un indicio de sobreajuste o de una tasa de aprendizaje muy alta.

Para entrenar el modelo previamente descrito se utilizó un optimizador estocástico de descenso de gradiente (SGD) y una tasa de aprendizaje de 0.01. Esta tasa de aprendizaje regula el tamaño de los pasos que el optimizador toma para converger hacia una solución óptima. Se eligió esta tasa de aprendizaje porque hace que el entrenamiento sea más rápido y mejora los resultados. La función de pérdida que se usó fue la entropía cruzada categórica (categorical_crossentropy). Esta función de pérdida es comúnmente utilizada en problemas de clasificación multiclase y mide cuán bien se ajustan las predicciones del modelo a las etiquetas verdaderas del conjunto de entrenamiento. Finalmente, se especificaron las métricas que se registrarán durante el entrenamiento para evaluar el rendimiento del modelo. La métrica seleccionada es la precisión (*accuracy*), que mide la proporción de predicciones correctas del modelo en relación con las etiquetas verdaderas en el conjunto de entrenamiento. Este entrenamiento duró 2320 segundos; es decir, aproximadamente 40 minutos.

Evaluación

La siguiente tabla proporciona un informe detallado del rendimiento del modelo de clasificación de la categorización de prendas de vestir en diez clases diferentes. Cada fila de la tabla representa una categoría de prenda de vestir, mientras que las columnas presentan diversas métricas de evaluación explicadas anteriormente.

	precision	recall	f1-score	support
Camiseta	0.84	0.82	0.83	1000
Pantalón	0.99	0.97	0.98	1000
Suéter	0.80	0.84	0.82	1000
Vestido	0.88	0.90	0.89	1000
Abrigo	0.78	0.88	0.82	1000
Sandalia	0.98	0.96	0.97	1000
Camisa	0.78	0.63	0.69	1000
Zapatilla deportiva	0.93	0.98	0.96	1000
Bolso	0.96	0.98	0.97	1000
Bota de tobillo	0.98	0.94	0.96	1000
accuracy			0.89	10000
macro avg	0.89	0.89	0.89	10000
weighted avg	0.89	0.89	0.89	10000

Model: "recurrente"		
Layer (type)	**Output Shape**	**Param #**
simple_rnn (SimpleRNN)	(None, 28, 256)	72960
simple_rnn_1 (SimpleRNN)	(None, 28, 128)	49280
simple_rnn_2 (SimpleRNN)	(None, 64)	12352
batch_normalization (Batch Normalization)	(None, 64)	256
dense (Dense)	(None, 10)	650

Total params	135498
Trainable params	135370
Non-trainable params	128

Conclusiones

Los resultados del informe de este modelo muestran una precisión general del 89%, demostrando su eficacia en la mayoría de las categorías. Sin embargo, se identifican áreas de mejora, especialmente en la clasificación de clases específicas como "Camisa" y "Abrigo". Este análisis nos proporciona una visión general de la capacidad del modelo y resalta que el rendimiento no es perfecto.

Red neuronal recurrente bidireccional

Creación del modelo

El modelo que se presentará es el mejor modelo que se encontró después de experimentar con diferentes valores y capas. Este modelo de red neuronal tiene varias capas. A continuación, se explican cada una de las capas en el orden en el que están en el modelo:

1. **Capa Bidireccional de LSTM (Long Short-Term Memory):** Dentro de esta capa hay otra 'layers.Bidirectional(layers.LSTM(256, return_sequences=True), input_shape=(28, 28))'. Esta es una capa bidireccional de LSTM, lo que significa que procesa la secuencia de entrada tanto en sentido ascendente como descendente. En este caso, se espera una entrada de secuencias con forma (28, 28). La capa tiene 256 unidades LSTM y devuelve secuencias en su salida ('return_sequences=True'), lo que significa que las salidas de cada paso de tiempo se pasan a la siguiente capa en la secuencia.
2. **Capa Bidireccional de LSTM:** Dentro de esta capa hay otra 'layers.Bidirectional(layers.LSTM(128))'. Esta es otra capa bidireccional de LSTM, que sigue procesando las secuencias de manera bidireccional. Aquí, la capa tiene 128 unidades LSTM y no se especifica 'return_sequences', lo que significa que esta capa produce una única salida para toda la secuencia de entrada, en lugar de producir

secuencias de salida. Esto puede utilizarse para capturar características de alto nivel de la secuencia de entrada.

3. **Capas Densas y de Dropout:** Se agrega una capa densa y luego una de dropout y esto se repite dos veces:
    - 'layers.Dense(256, activation='relu')'
    - 'layers.Dropout(0.3)'
    - 'layers.Dense(128, activation='relu')'
    - 'layers.Dropout(0.3)'

    Estas capas completamente conectadas se utilizan para procesar las características extraídas de las capas LSTM bidireccionales. La primera capa tiene 256 neuronas y utiliza la función de activación ‘relu’ para introducir no linealidad en el modelo. La capa de dropout con una tasa del 30% se utiliza para regularizar el modelo y prevenir el sobreajuste. La segunda capa completamente conectada tiene 128 neuronas y también utiliza la función de activación ‘relu’, seguida de otra capa de dropout.
4. **Capa de Salida Densa:** Usando 'layers.Dense(20, activation='softmax')' se agrega la capa de salida del modelo que tiene 10 neuronas. La función de activación softmax se utiliza en la capa de salida para convertir las salidas en probabilidades para cada clase. Esto permite que el modelo realice clasificación multiclase, donde la clase con la probabilidad más alta se considera la predicción final.

Entrenamiento

Antes de entrenar el modelo, se definieron los hiperparámetros que utilizará.

```
épocas de entrenamiento = 20
número de muestras de datos = 64
tasa de aprendizaje = 0.01
```

Además, al hacer la compilación del modelo se probaron los optimizadores ‘adam’ y ‘rmsprop’, aunque al final se optó por el ‘adam’. Estos hiperparámetros se eligieron después

de experimentar con diversos valores y para no ralentizar el entrenamiento. Este entrenamiento duró 77 minutos.

Evaluación

La siguiente tabla proporciona un informe detallado del rendimiento del modelo de clasificación de la categorización de prendas de vestir en diez clases diferentes. Cada fila de la tabla representa una categoría de prenda de vestir, mientras que las columnas presentan diversas métricas de evaluación explicadas anteriormente.

	precision	recall	f1-score	support
Camiseta	0.85	0.85	0.85	1000
Pantalón	0.99	0.99	0.99	1000
Suéter	0.84	0.83	0.83	1000
Vestido	0.93	0.88	0.90	1000
Abrigo	0.83	0.87	0.85	1000
Sandalia	0.99	0.96	0.98	1000
Camisa	0.73	0.75	0.74	1000
Zapatilla deportiva	0.92	0.99	0.96	1000
Bolso	0.99	0.99	0.99	1000
Bota de tobillo	0.98	0.94	0.96	1000
accuracy			0.90	10000
macro avg	0.91	0.90	0.90	10000
weighted avg	0.91	0.90	0.90	10000

Model: "recurrente"		
Layer (type)	**Output Shape**	**Param #**
bidirectional_5 (Bidirectional)	(None, 28, 512)	583680
bidirectional_6 (Bidirectional)	(None, 256)	656384
dense_7 (Dense)	(None, 256)	65792
dropout_4 (Dropout)	(None, 256)	0
dense_8 (Dense)	(None, 128)	32896
dropout_5 (Dropout)	(None, 128)	0
dense_9 (Dense)	(None, 10)	1290

Total params	1,340,042

Trainable params	1,340,042
Non-trainable params	0

Conclusiones

Se observa que la precisión global alcanza el 90%, lo que significa que el modelo acierta en la etiqueta de clase correcta en la mayoría de las ocasiones. Además, al analizar las métricas por clase, se puede ver que la mayoría de las categorías obtienen puntuaciones altas de precisión, recall y puntuación F1, lo que indica una buena capacidad de clasificación. Esto sugiere que el modelo es eficaz en la identificación de prendas de vestir en la base de datos Fashion MNIST. Sin embargo, vale la pena mencionar que algunas categorías, como "Camiseta" y "Camisa", tienen puntuaciones ligeramente más bajas en comparación con otras clases. Esto podría deberse a la similitud entre las prendas o desafíos específicos en la detección de esas categorías.

Discusión

El modelo más complicado definitivamente fue la red neuronal recurrente porque los hiperparámetros que mejor funcionaban para el convolucional, eran los parámetros que peores resultados daban en el recurrente, así que tomó tiempo descubrir eso y realizar los cambios necesarios. Además, la red neuronal recurrente fue más complicada porque no es la mejor para la clasificación de imágenes y se usa más en tareas de procesamiento de secuencias, como el procesamiento del lenguaje natural (NLP) o la predicción de series temporales. Aunque esta red neuronal típicamente no se usa para tareas de clasificación, me pareció interesante intentar utilizarla para esta tarea; ya que es una de las redes neuronales que más me intrigaba, y quería conocer su arquitectura y comparar su rendimiento con la convolucional. Por lo anterior, adaptarla para la clasificación de imágenes requirió un ajuste más detallado de su arquitectura y parámetros, lo que implicó un proceso de experimentación adicional. En contraste, la red neuronal convolucional se ha diseñado específicamente para tareas de visión por computadora, lo que facilitó su configuración inicial. Otra observación fue que entre más capas convoluciones y de pooling se agregaban, mejores eran los resultados; mientras que en la red recurrente era al revés. Las capas adicionales empeoraban los resultados y las RNN tienden a sufrir más el problema de desvanecimiento o explosión de gradientes, lo que puede dificultar la convergencia del modelo y hacer que el entrenamiento sea más lento. Además, las RNN son sensibles a la elección de hiperparámetros, como la tasa de aprendizaje y la arquitectura de la red, lo que requiere una cuidadosa afinación para obtener buenos resultados. Comparando los resultados de la red neuronal recurrente con la red neuronal recurrente bidireccional, podemos observar que la variante obtuvo mejores resultados con menos épocas y evitó de mejor manera el sobreajuste. Esto nos lleva a concluir que para esta tarea de predicción de imágenes, la variante de la RNN es más eficaz.

Evidentemente, el modelo de red neuronal convolucional se entrenó de manera más efectiva y obtuvo mejores resultados en términos de métricas de precisión, recall y F1-score. Su precisión general (*accuracy*) también fue superior, alcanzando el 93%, en comparación con el 89% del modelo RNN. Esto se debe a que las CNN son especialmente adecuadas para tareas de clasificación de imágenes, ya que pueden aprender automáticamente características relevantes de las imágenes a través de las capas de convolución y reducción de dimensionalidad. Además, el modelo CNN se entrenó de manera más estable y no mostró signos de sobreajuste, lo que sugiere que su capacidad de generalización fue más sólida en comparación con el modelo RNN.

Sin embargo, el modelo que requirió de más tiempo y recursos fue la red convolucional. Esto fue debido a que esta red tiene muchas más capas, se entrenó por muchas más épocas (150), el tamaño de lotes era mayor y la tasa de aprendizaje era mucho más pequeña. El tiempo de entrenamiento fue una de las ventajas de la red recurrente; ya que, aunque esta tuvo una puntuación de precisión menor que la convolucional (4% menos), el tiempo de entrenamiento fue mucho más bajo. Con esto podemos concluir que se sacrificó un poco de precisión por menor tiempo de procesamiento.

Conclusiones

Para el modelo de red neuronal convolucional algunas mejoras potenciales podrían incluir la exploración de arquitecturas más complejas como redes neuronales convolucionales profundas o la aplicación de técnicas de transferencia de aprendizaje utilizando modelos preentrenados en conjuntos de datos más grandes. Además, se podría aumentar el tamaño del conjunto de datos de entrenamiento para mejorar aún más la capacidad de generalización del modelo. En cuanto al modelo de red neuronal recurrente se podría investigar el uso de capas LSTM o GRU en lugar de las capas SimpleRNN para abordar problemas de desvanecimiento de gradientes y mejorar la eficacia en tareas de clasificación de imágenes. Sin embargo, es importante notar que se intentó utilizar capas LSTM para este proyecto, pero estas daban peores resultados que las SimpleRNN. Para mejorar el modelo de red neuronal recurrente bidireccional, se pueden explorar ajustes en la arquitectura, optimizadores y tasas de aprendizaje, regularización, tamaño de lote y épocas, funciones de activación, mecanismos de atención, alternativas en arquitecturas de RNN, hiperparámetros, y análisis de errores.

En general, el método que mostró un mejor desempeño en la tarea de clasificación de imágenes de moda fue el modelo de red neuronal convolucional. Este modelo superó al modelo de red neuronal recurrente en términos de métricas de precisión, recall y F1-score, así como en la precisión general (*accuracy*). El modelo CNN demostró ser más eficaz y estable en la tarea de procesamiento de imágenes, lo que lo convierte en la elección preferida para esta tarea en particular.

En este proyecto se aprendió a analizar las arquitecturas de las redes neuronales y a experimentar para llegar a los mejores resultados posibles. Los aspectos más importantes de una red neuronal son la cuidadosa selección de hiperparámetros, la arquitectura de la red y la elección de métricas de evaluación adecuadas. Además, se exploró la programación de una red neuronal recurrente que no se había visto en clase. Lo que más me gustó del proyecto es

la experimentación con diferentes configuraciones y observar como pequeñas modificaciones cambiaban los resultados drásticamente. Además, me gustó explorar la programación de las redes recurrentes bidireccionales.

Lo que menos me gustó del proyecto es que los datos de fashion MNIST están muy cuidados y ya preprocesados, entonces casi siempre dan muy buenos resultados desde el inicio aunque no hagas una red muy compleja. Sin embargo, esto también fue de gran utilidad para entender la arquitectura de la red y no atorarse con resultados negativos.

En general, este proyecto demuestra un enfoque sólido en la aplicación práctica de la inteligencia artificial en el campo de la clasificación de imágenes de moda y ofrece resultados prometedores en términos de precisión y eficacia del modelo.

Bibliografía

AulaMarketing.net. (2023). Qué es el machine learning o aprendizaje automático: claves y conceptos básicos. *AulaMarketing.net*. https://aulamarketing.net/que-es-el-machine-learning-aprendizaje-automatico#:~:text=Qu%C3%A9%20es%20el%20machine%20learning%20o%20aprendizaje%20autom%C3%A1tico,-Entender%20qu%C3%A9%20es&text=El%20aprendizaje%20autom%C3%A1tico%20se%20basa,expl%C3%ADcita%20para%20cada%20situaci%C3%B3n%20espec%C3%ADfica.

Canadas, R. (2022). Redes neuronales recurrentes | Qué son las RNN. *abdatum*. https://abdatum.com/tecnologia/redes-neuronales-recurrentes

Corredera, P. Á. (2023, 27 julio). ¿Qué es una red neuronal? *CIBERNINJAS*. https://ciberninjas.com/red-neuronal/

Fashion MNIST. (2017, 7 diciembre). Kaggle. https://www.kaggle.com/datasets/zalando-research/fashionmnist

Forgas, E. (2023, 20 julio). 11 Estadísticas de la moda online y los ecommerce de ropa. *SaleCycle*. https://www.salecycle.com/es/blog/estadisticas/estadisticas-moda-online-ecommerce/

Na, & Na. (2020). Convolutional Neural Networks: La teoría explicada en español | Aprende Machine Learning. *Aprende Machine Learning*. https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/

¿Qué es la inteligencia artificial (IA)? | IBM. (s. f.). https://www.ibm.com/mx-es/topics/artificial-intelligence

¿Qué son las redes neuronales convolucionales? | IBM. (s. f.). https://www.ibm.com/mx-es/topics/convolutional-neural-networks

¿Qué son las redes neuronales recurrentes? | IBM. (s. f.). https://www.ibm.com/mx-es/topics/recurrent-neural-networks#:~:text=Una%20red%20neuronal%20recurrente%20(RNN,voz%20y%20subt%C3%ADtulos%20de%20im%C3%A1genes.