Advances in Engineering and Innovation

Advances in Engineering and Innovation
Vol. 9, No. 19, pp. 350-358
Enero - Junio 2024

www.progreso.tecnm.mx/revistaAEI
ISSN: 2448-685X

Modelo de aprendizaje automático de red neuronal para la categorización de artículos de ropa con base a su fotografía

Neural network machine learning model for clothing item categorization based on photography

Irving Metelin-Rosado[1*], Erick Del Jesus Tamayo-Loeza[1], Ramón Salvador Mezquita-Martínez[1] e Iván de Jesús May-Cen[1]

[1]*Tecnologico Nacional de México campus Progreso, Boulevard Tecnológico de Progreso S/N x 62, CP 97320, Progreso, Yucatán, México.*

**Corresponding author:*
imetelin@itsprogreso.edu.mx

Resumen. La industria de la moda ha crecido enormemente, y la clasificación de la gran cantidad de artículos de ropa disponibles es un desafío. La relevancia de los modelos de aprendizaje automático y la visión por computadora para solucionar este problema, presenta un estudio que se enfoca en el desarrollo de modelos de aprendizaje automáticos capaz de categorizar prendas de ropa basándose en sus imágenes. El objetivo es mejorar la experiencia del usuario y facilitar la búsqueda de exploración de productos específicos de la moda. Para ello se emplean algoritmos de aprendizaje profundo y técnicas de procesamiento de imágenes para superar desafíos como la variabilidad en las imágenes y las cambiantes tendencias de moda. La precisión del modelo puede beneficiar a las empresas a la mejora de etiquetados de productos, optimización de las recomendaciones y ofrecimiento de una navegación más personalizada.

Palabras clave: Modelo, Aprendizaje, Categorización, fotografía, red neuronal.

Abstract. The fashion industry has grown significantly, and classifying the vast number of clothing items available is a challenge. The relevance of machine learning models and computer vision to address this issue is highlighted in a study that focuses on the development of automatic learning models capable of categorizing clothing based on their images. The goal is to enhance the user experience and streamline the search for specific

Recibido: 25/Septiembre/2023
Aceptado: 13/Noviembre/2023

fashion products. To achieve this, deep learning algorithms and image processing techniques are employed to overcome challenges such as image variability and changing fashion trends. The accuracy of the model can benefit companies by improving product labeling, optimizing recommendations, and providing a more personalized navigation experience.

Keywords: Model, Learning, Categorization, photography, neural network.

I. INTRODUCCIÓN

En la era digital actual, el comercio electrónico ha experimentado un crecimiento sin precedentes, y la industria de la moda no es una excepción. Con el auge de las plataformas de compras en línea, la vasta cantidad de artículos de ropa disponibles para los consumidores ha generado una necesidad crítica de herramientas inteligentes capaces de organizar y clasificar eficientemente estos productos. En este contexto, los modelos de aprendizaje automático y la visión por computadora han emergido como una solución prometedora para abordar este desafío.

El presente estudio se centra en la creación y desarrollo de un modelo de aprendizaje automático para la categorización de artículos de ropa con base a su fotografía. Este modelo busca agregar una herramienta en la selección de usuarios, brindándoles una forma más eficiente y precisa de buscar, explorar y encontrar productos específicos de la industria de la moda. (Goodfeller et. Al, 2018).

El principal objetivo de este proyecto es aprovechar el poder del aprendizaje automático para entrenar un modelo capaz de identificar automáticamente la categoría y el tipo de prendas de ropa presentes en una imagen proporcionada. Para ello, se utilizó un extenso conjunto de datos etiquetado que abarca una amplia gama de categorías de ropa, como camisetas, pantalones, calzado, faldas, sacos, entre otros.

El desarrollo del modelo implicó un enfoque multidisciplinario, combinando técnicas de visión por computadora, procesamiento de imágenes y aprendizaje automático. Se emplearon algoritmos de aprendizaje profundo, como redes neuronales, para extraer características relevantes y representativas de las imágenes de moda, permitiendo que el modelo aprenda a discernir entre diferentes categorías de prendas.

Se abordaron desafíos específicos, como la variabilidad en las imágenes debido a factores como poses, iluminación y fondo, así como la presencia de estilos de moda cambiantes y tendencias en constante evolución. Se buscará la robustez y la escalabilidad del modelo para garantizar su eficiencia en un entorno de producción real. (LeCun et al., 2015).

La importancia de este proyecto reside en su potencial para mejorar significativamente la experiencia del consumidor y optimizar la administración del inventario para las empresas de comercio electrónico de moda. Un modelo de categorización preciso puede facilitar el proceso de etiquetado de productos, mejorar las recomendaciones de productos y contribuir a una navegación más intuitiva y personalizada en las plataformas de compras en línea.

A. Redes neuronales en la clasificación de imágenes

Las redes neuronales son un tipo de algoritmo de aprendizaje automático que ha demostrado ser extremadamente poderoso en la clasificación y procesamiento de datos, especialmente en tareas de visión por computadora como el reconocimiento de imágenes. En el contexto del tema propuesto, las redes neuronales juegan un papel fundamental en la categorización precisa de artículos de ropa basada en fotografías.

Las arquitecturas de las redes neuronales están inspiradas en la estructura y funcionamiento del cerebro humano y están diseñadas específicamente para procesar imágenes. La arquitectura básica de una red neuronal se compone de varias capas interconectadas como: capa entrada, capas convolucionales, capas de activación, capas de agrupación y capa totalmente conectada.

Los entrenamientos de las redes neuronales se categorizan los artículos de ropa, por lo tanto, es necesario un conjunto de datos etiquetados que contenga imágenes de diferentes categorías de prendas. Durante el proceso de entrenamiento, la neurona ajusta los pesos y los parámetros de sus capas para minimizar la diferencia entre las predicciones del modelo y las etiquetas reales de las imágenes en el conjunto de entrenamiento.

Las capas densas, también conocidas como capas totalmente conectadas, son uno de los tipos fundamentales de capas utilizadas en redes neuronales artificiales (RNA) o modelos de aprendizaje profundo. Estas capas juegan un papel crucial en la transformación

y procesamiento de los datos a medida que pasan por la red. (Trask et al., 2019)
En una capa densa, cada neurona o unidad en la capa está conectada a todas las salidas de la capa anterior. Esto significa que cada entrada se conecta con cada neurona de la capa densa mediante una ponderación (peso) asociada. Durante el proceso de entrenamiento, estos pesos se ajustan para que el modelo pueda aprender a capturar patrones y relaciones relevantes en los datos de entrada.

La operación en una capa densa se puede describir matemáticamente como una multiplicación de matrices, seguida de una función de activación no lineal. Supongamos que tenemos una entrada con dimensiones (n, m), donde n es el número de muestras de entrada y m es el número de características en cada muestra. Si la capa densa tiene k neuronas, entonces tiene pesos y un sesgo para cada neurona.

La función de activación introduce no linealidades en el modelo lo que le permite aprender relaciones y patrones más complejos que los que se podrían capturar con funciones lineales. Una función de activación comúnmente utilizadas en capas densas es la función ReLU (Rectified Linear Unit).

B. Tensorflow en el procesamiento de imágenes
TensorFlow es una de las bibliotecas más populares y utilizadas para implementar modelos de aprendizaje automático y aprendizaje profundo. En la categorización de artículos de ropa basada en fotografías se puede emplear TensorFlow para crear un modelo de aprendizaje profundo que sea capaz de clasificar diferentes tipos de prendas de vestir (Patrick et al 2023).

TensorFlow también ofrece herramientas para el procesamiento de imágenes, lo que resulta fundamental en la preparación de los datos antes de entrenar el modelo. Las imágenes del conjunto de datos pueden ser redimensionadas, normalizadas y transformadas en tensores de manera eficiente utilizando las funciones y utilidades de procesamiento de imágenes proporcionadas por TensorFlow.

Para la aplicación de la librería se utilizó, Fashion-MNIST, que consiste en un popular conjunto de datos utilizado en el campo del aprendizaje automático y la visión por computadora. Es una variante del conjunto de datos MNIST, que ha sido ampliamente manipulado como un punto de referencia para algoritmos de aprendizaje automático, A diferencia del conjunto de datos MNIST, que contiene imágenes de dígitos escritos a mano, el conjunto de datos Fashin-MNIST está compuesto por imágnes en escala de grises de prendas de ropa.

C. Análisis del modelo de entrenamiento
La problemática establecida es un tema de clasificación, por lo tanto, se debe de reproducir como entrada la imagen y la salida debe de especificar la categoría que pertenece. Si se tienen diez categorías posibles, la red tendrá diez neuronas de salida, donde menciona a qué categoría cree que pertenece.

Figura 1. Representación de la imagen con la red neuronal. Fuente: Elaboración propia.

Para poder agregar una imagen a una red neuronal, se estableció una cantidad de caracteres conforme a los pixeles que contiene la imagen en blanco y negro, cada pixel tendrá una numeración estimada de acuerdo a la nitidez del color, dejando 1 como totalmente negro y 254 a blanco. Es importante especificar que mientras tenga mejor resolución la imagen, la cantidad de pixeles aumenta, una vez definido la calidad se propuso con un conjunto de pixeles de 28 x 28, así, 784 pixeles entran a 784 neuronas a la red y con una salida de 10 neuronas (LeCun et al., 2015).

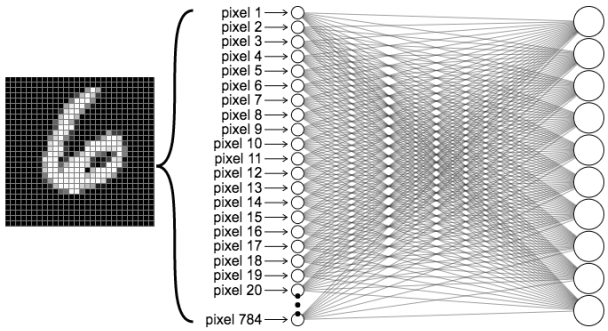

Figura 2. Representación de los pixeles28 x 28 entrando a una red neuronal, con salida de 10 neuronas. Fuente: Elaboración propia.

Aunque la red neuronal convolucional es normalmente utilizada para la clasificación de imágenes, esta vez se usó una red neuronal densa, con el objetivo de experimentar los alcances de entrenamiento y programación de una red convencional. (Vettriselval et al 2021).

Para una clasificación de imágenes, es necesario establecer el tipo de ecuación que brinda la capacidad de resolver problemas de separación de valores. Con la página de tensorflow para la visualización demostración de la resolución de las redes neuronales, se puede mostrar el comportamiento de solución de una red neuronal con ecuaciones lineales.

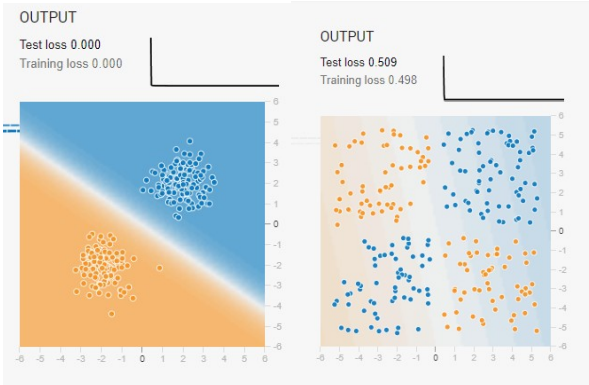

Figura 3. Solución de ecuación lineal para la separación de valores. Fuente: Elaboración propia.

En la primera ilustración, se puede apreciar que la solución no se le hizo complicado a la red para la separación de los valores con la ecuación. Es notorio que los valores ya estaban con cierta clasificación desde el inicio, pero si se observa en la segunda imagen donde los valores están de cierta manera revueltos, a la ecuación se le dificulta resolverlo, manteniendo bastantes porcentajes de error en el proceso.

Como conclusión en este experimento, una ecuación lineal para una red neuronal se le dificulta en gran medida. Para una solución con mayor eficiencia es necesario darle herramientas a la red para poder resolver problemas más complejos y no esté limitada solo solucionar con ecuaciones lineales, estas herramientas son las capas ocultas y las funciones de activación.

En una red neuronal donde solo está la entrada conectada a la salida, se tiene muy poco espacio para transformaciones, debido a que solo se pueden resolver problemas con una multiplicación y una suma. Por lo tanto, si se quiere dar a la red más espacio para que realice transformaciones más complejas, entonces se le agregan capas ocultas que siguen la misma regla de realizar multiplicaciones y sumas, sin embargo, al añadirlas a la red la red cuenta con más opciones para poder realizar operaciones lineales para poder unificar los resultados.

Pero de todas maneras el solo usar capas ocultas no es suficiente para la solución de la clasificación de las imágenes, debido a que éstas solo realizan operaciones lineales y como se vio en las ilustraciones pasadas en el que los resultados no fueron muy satisfactorios. Es por ello que las neuronas necesitan salir de las estimaciones de resolver problemas lineales, una vez que una neurona suma su sesgo y va a dar su resultado, se pasa por una función de activación el cual brinda una nueva evaluación de acuerdo la función establecida.

Existen diversas funciones de activación que brindan varias maneras de entrenamiento en las neuronas, entra las más destacadas se encuentran: ReLU, Tanh, Sigmoid y Linear. Para esta ocasión se utilizó la función de activación ReLU, que consiste en una función que propone que al salir el resultado rectifica desde abajo cuando el valor es menor que cero.

En las siguientes imágenes se puede apreciar el comportamiento al agregar más capas ocultas a la neurona y la función de activación en problemas de clasificación más complejos. En la primera dos ilustraciones pueden observarse que no se le complica resolver a pesar de que los puntos están dispersos y el error es mínimo, debido a que no solo propone procedimientos lineales, sino que se apoya con la función de activación. Mientras que, en la tercera muestra sí se puede apreciar el error y no llega a alcanzar a cubrir en su totalidad el resultado, pero muestra un resultado casi satisfactorio.

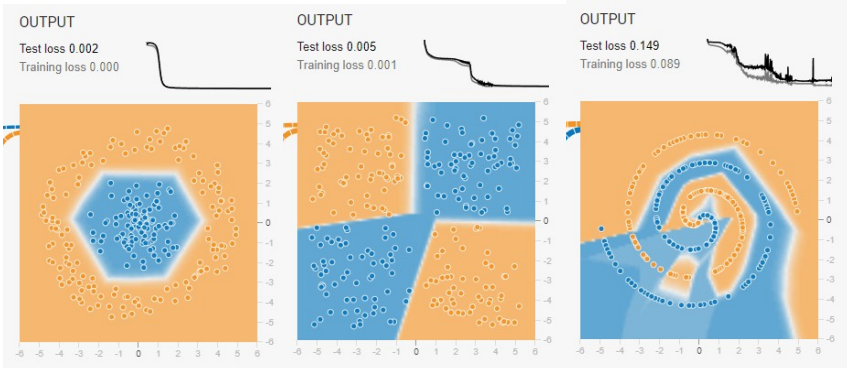

Figura 4. Muestras del entrenamiento con más capas ocultas y la función de activación ReLU. Fuente: Elaboración propia.

Al realizar las pruebas para implantar los números de capas ocultas y la función de activación en la red, quedó definido que las neuronas entrarán con las 784 neuronas que consisten en la cantidad de pixeles por cada imagen de 28x28, dos capas ocultas con 50 neuronas cada una en el que utilizó la función de activación ReLU, y con salida de 10 neuronas que fundamentan en los nombramientos de cada artículo.

D. Procedimiento de entrenamiento de la red neuronal

Debido a que la clasificación es por imágenes, es necesario agregar las muestras de las ilustraciones de los artículos de ropa que ayuden entrenar a la red neuronal. Para evitar esta tediosa actividad, TensorFlow junto con la tienda Zalando se dieron a la tarea de ofrecer un set de datos de 70,000 imágenes que están categorizadas correctamente con las funciones que se están proponiendo en el proyecto, con el fin de entrenar las redes neuronales de este tipo y poder descargarlas desde la importación de la librería.

```
[1] import tensorflow as tf
    import tensorflow_datasets as tfds

    datos, metadatos = tfds.load('fashion_mnist', as_supervised=True, with_info=True)

    Downloading and preparing dataset 29.45 MiB (download: 29.45 MiB, generated: 36.42 MiB, total: 65.87 MiB) to /root/tensorflow_datasets/fashion_mnist/3.0.1...
    Dl Completed...: 100% 4/4 [00:02<00:00, 1.49 url/s]
    Dl Size...: 100% 29/29 [00:02<00:00, 21.78 MiB/s]
    Extraction completed...: 100% 4/4 [00:02<00:00, 1.25 file/s]
    Dataset fashion_mnist downloaded and prepared to /root/tensorflow_datasets/fashion_mnist/3.0.1. Subsequent calls will reuse this data.
```

Figura 5. Importación de librerías de TensorFlow. Fuente: Elaboración propia.

Se importó la librería TensorFlow con 'datasets' y la base de datos 'fashion_mist', fueron cargadas las imágenes disponibles que serán necesarias para el entrenamiento de la red, se utilizaron 60,000 capturas que corresponden al aprendizaje de las neuronas y 10,000 realizaron las pruebas con el fin de comprobar si el adiestramiento fue satisfactorio. Además, por estandarización la librería te presenta las categorías que clasifica cada artículo en el que están ordenados de acuerdo a su listado del 0 al 9.

```
[4] datos_entrenamiento, datos_pruebas = datos['train'], datos['test']

[5] nombres_clases = metadatos.features['label'].names

[6] nombres_clases

['T-shirt/top',
 'Trouser',
 'Pullover',
 'Dress',
 'Coat',
 'Sandal',
 'Shirt',
 'Sneaker',
 'Bag',
 'Ankle boot']
```

Figura 6. Clasificación de las neuronas de salida. Fuente: Elaboración propia.

Cuando se está entrenando la red, es necesario normalizar los datos para que las entradas sean de 0 y 1, esto ayuda a mejorar el entrenamiento y los resultados. Por lo tanto, los valores que se reciben serían de 0 a 255

debido a que corresponden a cantidad de los pixeles, así se utilizó una función de normalización para poder dividir cada pixel entre 255 y pasar todos los datos de entrenamiento y prueba con dicha normalización.

```
def normalizar(imagenes, etiquetas):
  imagenes = tf.cast(imagenes, tf.float32)
  imagenes /= 255
  return imagenes, etiquetas

datos_entrenamiento = datos_entrenamiento.map(normalizar)
datos_pruebas = datos_pruebas.map(normalizar)

datos_entrenamiento = datos_entrenamiento.cache()
datos_pruebas = datos_pruebas.cache()
```

Figura 7. Normalización de los datos. Fuente: Elaboración propia.

E. Modelo de entrenamiento

El código representado en el modelo de entrenamiento de la red neuronal profunda usa la API de alto nivel Keras de TensorFlow. Específicamente, es un modelo secuencial, lo que significa que las capas se apilan secuencialmente una encima de otra. El modelo se define de la clase 'tf.keras.Sequential', que permite crear una secuencia lineal de capas (Gallego et al 2023).

```
modelo = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28,28,1)),
    tf.keras.layers.Dense(50, activation=tf.nn.relu),
    tf.keras.layers.Dense(50, activation=tf.nn.relu),
    tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
```

Figura 8. Código de modelo de entrenamiento con sus capas ocultas y su función de activación. Fuente: Elaboración propia.

En la primera capa del modelo se utilizó la función para aplanar la imagen de entrada. La red espera que tenga una forma de (28,28,1), lo que se refiere a la escala de la ilustración en grises con el tamaño de 28x28 pixeles. La capa 'Flatter' la convierte en 2D en un vector unidimensional de 784 elementos, que se utiliza como entrada para las capas densamente conectadas.

La segunda es una capa densa con 50 neuronas, en el que cada neurona de esta capa está conectada a todas las salidas de la capa anterior, lo que implica que hay 50 pesos diferentes que se ajustan el entrenamiento para aprender patrones en los datos. La función de activación ReLU, introduce la no linealidad en la red permitiéndole aprender representaciones más complejas de los datos. La tercera capa es otra capa densa con 50 neuronas y función de activación ReLU. Igual que la capa anterior, esta capa también aprende representaciones más abstractas de los datos.

Por último, la capa de salida del modelo es otra capa densa con 10 neuronas, que aplica la función de activación 'sofmax', que es apropiada para problemas de clasificación multiclase. La capa 'saftmax' toma los valores de salida y los convierte en una distribución de probabilidad sobre las 10 clases posibles, lo que permite que el modelo realice una clasificación precisa. Esto con el propósito de que, en la salida, donde le otorgará valores del 0 al 1 en las clasificaciones de acuerdo a lo que puede identificar el modelo y dando como sumatoria uno.

```
[ ]  modelo.compile(
        optimizer='adam',
        loss=tf.keras.losses.SparseCategoricalCrossentropy(),
        metrics=['accuracy']
    )
```

Figura 9. Compilación mediante el modelo 'adam'. Fuente: Elaboración propia.

Se utilizó 'adam' para compilar un modelo de red neuronal usando la API de alto nivel Keras de TensorFlow. La función 'compile()' se aplica para configurar el modelo en el proceso de entrenamiento.
Una vez que el modelo se compila con la configuración adecuada, está listo para ser entrenado ejecutado los datos de entrenamiento y validado en los datos de prueba. Durante el entrenamiento, el modelo ajusta sus pesos para minimizar la función de pérdida específica, aplicando el optimizador y actualizando los pesos en función de los gradientes calculados durante el proceso de retropropagación.

Para poder entrenar la red neuronal se realiza mediante la función 'fit' con cinco épocas importando la biblioteca estándar de matemáticas de Python. El código crea una variable que se utiliza para almacenar el historial las métricas de entrenamiento de un modelo de aprendizaje automático. Al aplicar el modelo fit con las cinco épocas, en cada vuelta muestra las pérdidas y la precisión se puede observar y analizar el modelo establecido.

```
[ ] import math

    historial = modelo.fit(datos_entrenamiento, epochs=5, steps_per_epoch=math.ceil(num_ej_entrenamiento/TAMANO_LOTE))

    Epoch 1/5
    1875/1875 [==============================] - 17s 3ms/step - loss: 0.5292 - accuracy: 0.8140
    Epoch 2/5
    1875/1875 [==============================] - 6s 3ms/step - loss: 0.3796 - accuracy: 0.8615
    Epoch 3/5
    1875/1875 [==============================] - 5s 3ms/step - loss: 0.3484 - accuracy: 0.8727
    Epoch 4/5
    1875/1875 [==============================] - 6s 3ms/step - loss: 0.3328 - accuracy: 0.8769
    Epoch 5/5
    1875/1875 [==============================] - 6s 3ms/step - loss: 0.3193 - accuracy: 0.8814
```

Figura 10. Entrenamiento de la red. Fuente: Elaboración propia.

Con la red ya entrenada, se aplica la biblioteca Matplotlib para crear un gráfico de la magnitud de pérdida en función del número de épocas, generando una línea que es útil para evaluar el rendimiento del modelo a medida que avanza el entrenamiento y detectar los posibles problemas de sobreajustes y subajustes.

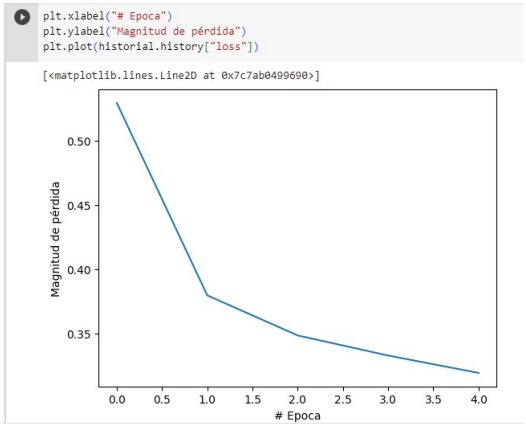

Figura 11. Gráfica de entrenamiento. Fuente: Elaboración propia.

F. Resultados del entrenamiento

Con las neuronas entrenadas en las funciones y capas que se aplicaron durante las evaluaciones en el modelo, se realizó una prueba general para intentar predecir el conjunto de imágenes con la red ya analizada. Se definieron con una franja de color azul las muestras donde el código haya acertado bien y una franja roja si se obtuviera un error estimado, prediciendo erróneamente en dos muestras.

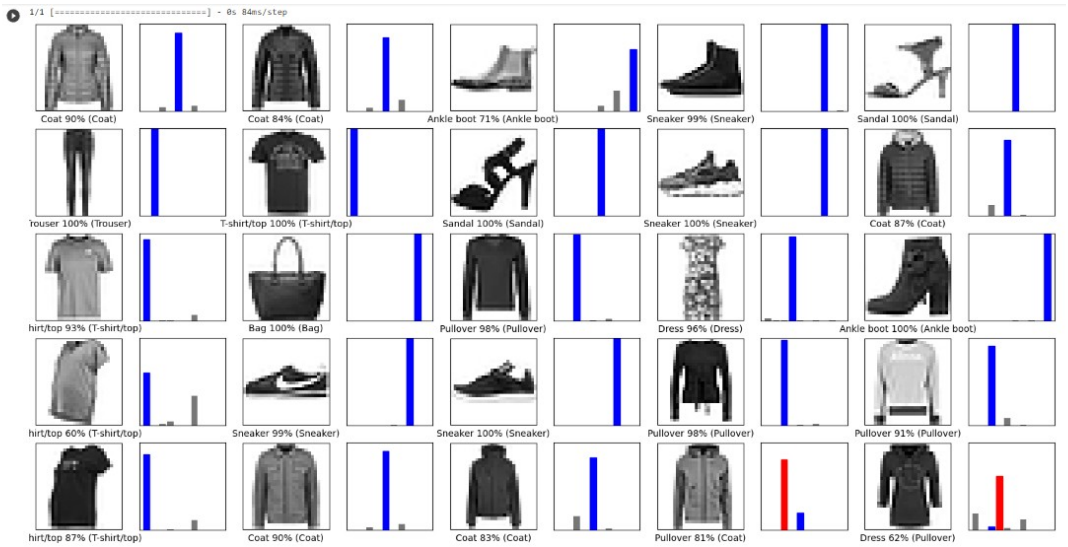

Figura 12. Prueba de predicción para la clasificación de imágenes. Fuente: Elaboración propia.

Debido a que para una prueba inicial se puede dar por alto dichos porcentajes de error, se prosiguió a aplicar el código para que pueda agarrar cualquier dato en el proceso e intente predecir por separado la función de la red, ofreciendo datos accesibles.

```
imagen = imagenes_prueba[30]
imagen = np.array([imagen])
prediccion = modelo.predict(imagen)

print("Predicción: " + nombres_clases[np.argmax(prediccion[0])])
```

```
1/1 [==============================] - 0s 64ms/step
Predicción: Sneaker
```

Figura 13. Resultado de predicción en modo individual. Fuente: Elaboración propia.

CONCLUSIÓN

Utilizar un modelo de aprendizaje automático para clasificar artículos de ropa basándose en su fotografía es una decisión muy ventajosa y prometedora en el campo del comercio electrónico y la industria de la moda. Permitiendo clasificar automáticamente una gran variedad de artículos de ropa, lo que es especialmente valioso para tiendas en línea con catálogos extensos.

Aunque en este caso el aprendizaje fue realizado mediante una red neuronal densa la experiencia puede ayudar desarrollar mejores modelos utilizando redes neuronales convolucionales (CNN) que demuestra ser eficiente para la clasificación de imágenes. Además, estos modelos son altamente escalables, lo que significa que pueden procesar grandes volúmenes de imágenes rápidamente, lo que resulta ideal para empresas y plataformas en línea o productos de grandes cantidades.

En general, el uso de modelos de aprendizaje automáticos para clasificar artículos de ropa basándose en imágenes ofrece numerosos beneficios y oportunidades para la industria de la moda en plataformas digitales, lo que mejora la experiencia del cliente y brinda ventajas competitivas a las empresas que lo implementan de manera efectiva.

REFERENCIAS

Aggarwal Charu C. (2018). Neural Networks and Deep Learning: A Textbook. Springer. ISBN- 13: 978-3319944623.

Gallego Sánchez A. J., Alfaro Contreras M. (2023) Deep Learning con Keras y PyTorch. Anaya Multimedia. ISBN: 978-8441548169.

Goodfellor, I. (2018). Deep Learning. Das umfassende Handbuch. MIT Press.

LeCun, Y, Bengio, Y, Hinton, G. (2015). "Deep Learning". Nature. DOI: 10.1038/nature14539.

Nielsen, M A. (2015). "Neural Networks and Deep Learning: A Textbook. Determination Press. ISB: 978-0615989306.

Patrick J. (2023). TensorFlow Developer Certification Guide. Gitforgits. ISBN: 978-8119177325

Raschka, S, Mirjalili, V. (2020). "Python Machine Learning". Packt Publishing. ISBN: 978-1789955750.

Trask, A. (2019). "Grokking Deep Learning". Manning Publications. ISBN: 978-1617293702.

Vettriselval R, Aarthy S. L, Sujatha R. (2021). Integrating Deep Learning Algorithms to Overcome Challenges in Big Data Analytics. CRC Press. ISBN: 978-1000454543.