ISBN: 978-628-95207-8-1. ISSN: 2414-6390. Digital Object Identifier: 10.18687/LACCEI2024.1.1.1293

Modelo Clasificador de prendas de vestir basado en Redes Neuronales

Antonio Arroyo-Paz[1], Leonid Aleman-Gonzales[2], Marga Ingaluque-Arapa[2], Pablo Tapia-Catacora[2], Adolfo Jimenez-Chura[2], Hugo Marca-Maquera[3], Cesar Rodriguez-Aburto[4]
[1] Universidad Tecnológica del Perú, Perú, c25921@utp.edu.pe
[2] Universidad Nacional del Altiplano, Perú, laleman@unap.edu.pe, miingaluque@unap.edu.pe, pctapia@unap.edu.pe, ajimenez@unap.edu.pe
[3] Universidad Nacional de Moquegua, Perú, hmarcam@unam.edu.pe
[4] Universidad Nacional del Callao, Perú, carodrigueza@unac.edu.pe

Abstract*– La clasificación de ropa es un proceso que revoluciona la organización de armarios y permite ofrecer una mejor experiencia de compra en los comercios electrónicos. Una de las alternativas tecnológicas para abordar este proceso son las Redes Neuronales Convolucionales (CNN) debido a su óptima comprensión de características particulares mediante una representación visual, como una imagen. El presente artículo tiene como finalidad implementar un modelo CNN que procese información de imágenes de prendas de vestir mediante capas ocultas para identificar patrones distintivos que permitan su categorización eficaz. Se utilizó una metodología basada en la construcción de un modelo clasificatorio mediante el lenguaje de programación Python, las librerías "TensorFlow" y "TensorFlow Datasets", la máquina virtual de Google Colaboraty y las imágenes tomadas del repositorio público "Fashion-MNIST", perteneciente a la tienda online de ropa "Zalando". Como resultado se obtuvo una CNN funcional con una precisión de 88.57%. Finalmente, se considera a este trabajo como un artículo de referencia para futuros trabajos que se enfoquen en las utilidades prácticas de una CNN en distintos campos.

Keywords*-- Red Neuronal Convolucional (CNN), Aprendizaje Profundo, Reconocimiento de ropa.

I. INTRODUCCION

En la actualidad, la categorización de las prendas de vestir destaca por su impacto en la eficiencia y sostenibilidad personal; este proceso revoluciona la organización de armarios y redefine la experiencia de compra, ofreciendo recomendaciones personalizadas en los comercios electrónicos. En este sentido, la aplicación de un clasificador de ropa mediante imágenes ayuda a la recolección y almacenamiento de datos, contribuye a la creación de un sistema de recomendación y simplifica la organización de prendas. Sin embargo, el etiquetado manual es una tarea difícil que consume mucho tiempo y disminuye la eficiencia de etiquetado debido a la influencia del criterio subjetivo del cliente o usuario [1].

En este contexto, la tecnología desempeña un rol crucial, especialmente en la clasificación de imágenes mediante la aplicación de un modelo compuesto por algoritmos basados en inteligencia artificial. Según San-Payo et al. [2], se puede adicionar algoritmos de aprendizaje automático para mejorar la precisión de la identificación de elementos visuales.

Por lo cual, es necesario identificar una herramienta que proporcione una capacidad optima de procesamiento y compresión de imágenes, Las Redes Neuronales Convolucionales (CNN) funcionan de manera similar al córtex visual del cerebro humano. Procesan sus capas de una forma que imita cómo el córtex visual identifica y reconoce diferentes características en la información de entrada. Así como nuestro cerebro analiza patrones y formas visuales, las CNN están diseñadas para detectar y reconocer distintos rasgos y características en los datos de entrada que reciben. De acuerdo con Artola [3], esto se consigue mediante varias capas especializadas y ocultas que contiene de forma jerárquica la CNN; Las capas iniciales de una Red Neuronal Convolucional tienen la función de identificar y reconocer características simples o propiedades básicas en los datos de entrada. A medida que se avanza hacia capas más profundas y especializadas dentro de la red, estas se vuelven más capacitadas para detectar y reconocer patrones y formas cada vez más complejas. Eventualmente, las capas finales alcanzan un nivel de sofisticación que les permite identificar elementos muy elaborados como rostros humanos o siluetas de objetos completos.

Desde el punto de vista de Massiris, Delrieux y Fernández [4], el propósito de una CNN es extraer todas las características de una imagen para ser utilizadas en la detección y clasificación categórica de las clases (objetos). Además, se indica que los filtros utilizados en las diferentes capas de una Red Neuronal Convolucional se ajustan y optimizan junto con los componentes encargados de la clasificación. El objetivo es minimizar al máximo el error total que se produce en la tarea de clasificación. Es decir, los parámetros de los filtros de las capas convolucionales se sincronizan y afinan en conjunto con los elementos que determinan la clasificación final, de modo que se reduzca al mínimo el margen de error en los resultados de clasificación.

Por lo tanto, en el presente artículo se pretende implementar un modelo de red neuronal convolucional que procese información de imágenes de prendas de vestir mediante capas ocultas para identificar patrones distintivos que permitan su categorización eficaz.

El documento está organizado de la siguiente manera. La sección 2, cubre las investigaciones relacionadas con el tema de este trabajo. La sección 3, Metodología, menciona los recursos que se utilizaron y describe los pasos para construir el modelo CNN. La sección 4, presenta de forma sintetizada los

resultados de clasificación del modelo. Finalmente, en la sección 5, se presenta las conclusiones del trabajo.

II. TRABAJOS RELACIONADOS

La clasificación de imágenes es ampliamente utilizada en el campo de visión por computadora y aprendizaje automático, contribuyendo a su progreso y avance. Se pueden identificar dos enfoques principales para la clasificación de imágenes de ropa: los algoritmos de aprendizaje automático convencionales y los modelos de aprendizaje profundo [5].

1. *Aplicación De Redes Neuronales Convolucionales En La Clasificacion De Ropa*

Las Redes Neuronales Convolucionales (CNN) han representado un enfoque crucial en la clasificación de prendas de vestir dentro del campo del análisis de visión por computadora, generando avances notables en términos de precisión y eficiencia [6].

La mayoría de los modelos empleados en la clasificación de imágenes, se estructuran mediante capas especializadas de Redes Neuronales Convolucionales (CNN), que procesan información de manera progresiva y jerárquica. Una investigación reciente, desarrollo un sistema capaz de clasificar la vestimenta del "grupo étnico She" proveniente de cinco regiones diferentes en el sureste de China. Este sistema emplea la extracción de histogramas de color y características de momento de color, integrando la optimización mediante el algoritmo de polinización de flores (FPA), para fusionar estas características cromáticas. Además, implementa el análisis de componentes principales del kernel junto con un marco de aprendizaje profundo de CNN [7].

El aprendizaje profundo es una tecnología en constante evolución que demuestra una notable eficacia en la clasificación de imágenes de prendas de vestir de manera eficiente. El enfoque reciente en la extracción de características mediante aprendizaje profundo ha impactado positivamente en la clasificación de imágenes, especialmente en prendas de vestir, al reducir el proceso de abstracción de datos [8]. Estos avances resaltan la importancia del aprendizaje profundo en lograr una clasificación precisa de prendas de vestir.

De igual manera, algunos estudios utilizan el aprendizaje profundo con CNN, lo que permite una alta precisión en la identificación y clasificación de caracteres en los modelos propuestos y/o desarrollados. Un estudio en Corea abordó la ineficiencia de un sistema de recolección de ropa usada, y propuso un método para clasificar imágenes de estas prendas mediante cámaras ubicadas en diferentes lugares. La aplicación de aprendizaje profundo con CNN permitió realizar clasificaciones en dos categorías (superior e inferior) o incluso en múltiples clases. Los resultados se almacenaron en la nube a través de la informática de punta, lo que posibilitó el análisis de datos en dispositivos antes de su transmisión [9].

Digital Object Identifier: (only for full papers, inserted by LACCEI).
ISSN, ISBN: (to be inserted by LACCEI).
DO NOT REMOVE

III. METODOLOGÍA

La Figura 1 representa la estructura de la metodología propuesta para clasificar las imágenes de prendas de vestir, divida en las siguientes partes:

1. Conjunto de datos

El conjunto de datos consta de 70,000 imágenes que representan una amplia variedad de prendas de vestir, distribuidas en 10 categorías diferentes (Tabla 1). Estas categorías incluyen camisetas, zapatos deportivos, bolsos y otros artículos similares. Las imágenes fueron recopiladas de Zalando, una empresa de moda que proporciona un conjunto de datos denominado Fashion-MNIST, el cual contiene artículos de la misma compañía. Puede acceder al repositorio de datos en GitHub mediante el siguiente enlace: https://github.com/zalandoresearch/fashion-mnist#fashion-mnist.

Es importante destacar que estas imágenes están organizadas y clasificadas por cada tipo de prenda.

TABLA I
TYPE SIZE FOR PAPERS

N°	Prenda
1	Camiseta / Top
2	Pantalón
3	Suéter
4	Vestido
5	Abrigo
6	Sandalia
7	Camisa
8	Zapatilla
9	Bolsa

Las imágenes suministradas en esta base de datos están en escala de grises y tienen un tamaño de 28 x 28 píxeles (Figura 2).

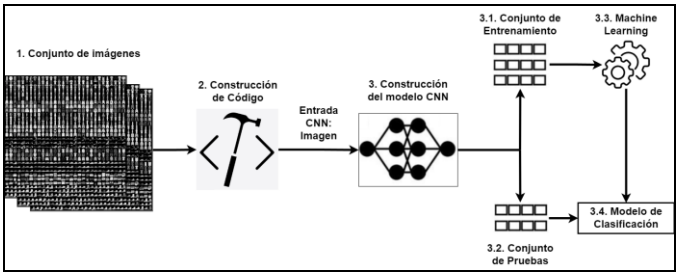

Figura 1. Metodología propuesta, elaborada a partir de [13].

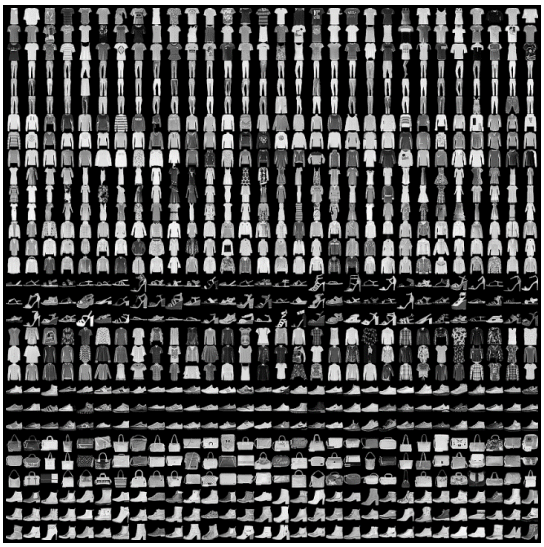

Figura 2. Imágenes del conjunto Fashion-MNIST.

Para la construcción del código, se optó por Google Colaboratory o Colab, una herramienta que facilita la escritura y ejecución de código Python [10] directamente desde el navegador. Esta plataforma no solo posibilita la creación de código, sino que también ofrece la opción de compartir el entorno de trabajo de manera colaborativa.

En primer lugar, se procede a importar las siguientes librerías para el manejo de datos (Código 1, 2):

- TensorFlow es una biblioteca de código abierto enfocada en el aprendizaje automático (Machine Learning), creada por Google para cubrir sus propias necesidades relacionadas con las redes neuronales artificiales. Por lo tanto, TensorFlow se fundamenta en los principios y técnicas del aprendizaje profundo (deep learning), que se basa en el uso de redes neuronales complejas y de múltiples capas [11]. Su implementación se lleva a cabo mediante la importación.

```
1  import tensorflow as tf
```

Código 1. Importación de TensorFlow para Desarrollo de Machine Learning.

- TensorFlow Datasets es una recopilación de conjuntos de datos preparados y listos para utilizar, no solo con TensorFlow, sino también con otros marcos de aprendizaje automático escritos en Python. Todos los conjuntos de datos se presentan en formato tf.data.Datasets, lo cual facilita la creación de canalizaciones (pipelines) de entrada de datos de alto rendimiento y de uso sencillo. [12].

```
2  import tensorflow_datasets as tfds
```

Código 2. Importación de TensorFlow DataSets para Desarrollo de Machine Learning.

Descargamos el set de datos "Fashion-MINIST" de Zalando para almacenar y utilizar las imágenes (Código 3).

```
1  datos, metadatos = tfds.load(
2      'fashion_mnist',
3      as_supervised=True,
4      with_info=True)
```

Código 3. Descarga del conjunto de datos.

La línea de código carga el conjunto de datos 'fashion_mnist' desde TensorFlow Datasets, incluyendo tanto los datos de las imágenes como sus etiquetas. Asimismo, recupera información adicional sobre el conjunto de datos almacenada en el objeto metadatos.

Antes de iniciar el trabajo con los datos obtenidos, se realizó una validación visual mostrando los datos mediante la impresión de una representación gráfica (Código 4).

```
1  metadatos
```

Código 4. Impresión de Metadatos.

Posteriormente, se dividió el conjunto de datos en dos partes: entrenamiento (60,000 imágenes) y pruebas (10,000 imágenes), asignados a variables separadas, 'datos-entrenamiento' y 'datos-pruebas', respectivamente (Código 5).

```
1  datos_entrenamiento, datos_pruebas = (
2      datos['train'], datos['test'])
```

Código 5. Separación de datos (Entrenamiento - Prueba)

Con respecto a las etiquetas, se generan 10 categorías posibles y se almacenan en 'nombres-clases', empleando la información proveniente de las características del metadato (Código 6).

```
1  nombres_clases = (
2      metadatos.features['label'].names)
```

Código 6. Etiquetado de categorías.

Luego de asignar las categorías posibles para representar la clasificación de datos en la variable 'nombres-clases', se procede a mostrarlas en la consola (Figura 3).

```
1  nombres_clases
```

Código 7. Imprimir por consola los nombres de las clases.

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

Figura 1. Impresión de categorías.

Posterior a ello, se define una función 'normalizar' para ajustar los valores de los pixeles de las imágenes de 0-255 a 0-1, para mejorar el aprendizaje de la red. Luego, se aplica esta función a los datos de entrenamiento y pruebas, utilizando 'map'. Además, se optimiza rendimiento al almacenar en caché los datos, favoreciendo un entrenamiento más rápido (Código 8, 9).

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

Código 8. Función de Normalización.

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

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

Código 9. Normalización de valores y almacenamiento en cache para acelerar el entrenamiento.

A continuación, se selecciona la primera imagen del conjunto de datos de entrenamiento para su visualización. Después de transformarla a una matriz NumPy y redimensionarla a 28x28 píxeles, se utiliza la librería Matplotlib para representar la imagen. La función 'imshow' muestra la imagen, mientras que 'colorbar' añade una barra de colores para representar los valores, y 'grid' desactiva la cuadrícula en el gráfico (Código 10).

```
for imagen, etiqueta in datos_entrenamiento.take(1):
  break
imagen = imagen.numpy().reshape((28,28))

import matplotlib.pyplot as plt

plt.figure()
plt.imshow(imagen, cmap=plt.cm.binary)
plt.colorbar()
plt.grid(False)
plt.show()
```

Código 10. Código para representar la imagen inicial de prueba.

Por último, 'show' presenta la imagen en una ventana gráfica (Figura 3).

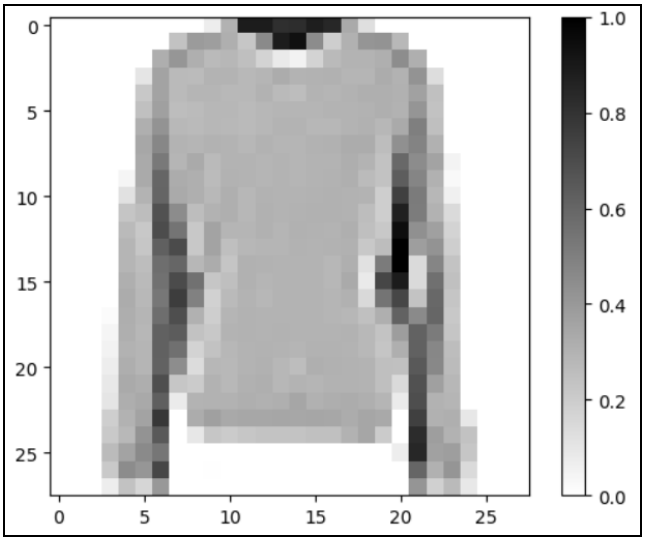

Figura 4. Imagen inicial de prueba.

Construcción del modelo CNN

En primer lugar, se creó una variable denominada “modelo” que hace referencia a una red secuencial. Para acceder al modelo, se define una capa de entrada manualmente utilizando el tipo “Flatten”, que aplanará la entrada de imágenes 28 x 28 píxeles en escalas de grises (blanco y negro) mediante un solo canal. La capa “Flatten” se encarga de convertir la imagen en una sola dimensión, con 784 neuronas, donde se recibirá cada píxel.

Posteriormente, se agregan dos capas ocultas densas con 50 neuronas cada una, con una función de activación ‘relu’, seguidas por una capa de salida con 10 neuronas. A esta capa de salida se le proporcionara la función de activación ‘Softmax’, ideal para redes de clasificación, que calcula la probabilidad de pertenencia a las 10 categorías. La función ‘Softmax’ garantiza que la suma de las neuronas de salida siempre sea igual a 1 (Código 11).

```
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)
])
```

Código 11. Creación del Modelo.

Luego de que el modelo está configurado, se compila usando el optimizador 'adam' y la función de pérdida 'SparseCategoricalCrossentropy', apropiada para clasificación con etiquetas dispersas, y registra la métrica de precisión 'accuracy' para evaluar el rendimiento (Código 12).

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

Código 12. Compilación del Modelo.

Luego, se obtiene la cantidad de ejemplos en los conjuntos de entrenamiento y pruebas del dataset. Los valores almacenados en 'num_ej_entrenamiento' y 'num_ej_pruebas' almacenan la cantidad de ejemplos en los respectivos conjuntos de datos de entrenamiento y pruebas (Código 13).

```
num_ej_entrenamiento = (
    metadatos.splits["train"].num_examples)
num_ej_pruebas = (
    metadatos.splits["test"].num_examples)
```

Código 13. Almacenamiento de datos.

Después, se utiliza la función 'print' para visualizar la en la consola la cantidad de datos, con la finalidad de confirmar los tamaños de los conjuntos de datos de entrenamiento y pruebas (Código 14).

```
print(num_ej_entrenamiento)
print(num_ej_pruebas)
```

Código 14. Imprimir cantidad de datos.

```
60000
10000
```

Figura 5. Cantidad de datos de Entrenamiento y Pruebas.

Por otro lado, para que la red entrene de manera más rápida, se estable un lote de tamaño 32. Luego, se manipulan los datos de entrenamiento para ser seleccionados aleatoriamente y no en un orden especifico, para ello se utiliza 'repeat' y 'shuffle', con el número de ejemplos de entrenamiento, agrupados en lotes del tamaño definido. Para los datos de prueba, se agrupan simplemente en lotes del tamaño especificado sin repetición o mezcla. Esta técnica optimiza el entrenamiento con grandes conjuntos de datos y evita patrones secuenciales en la red (Código 15).

```
TAMANO_LOTE = 32

datos_entrenamiento = (
    datos_entrenamiento.repeat()
    .shuffle(num_ej_entrenamiento)
    .batch(TAMANO_LOTE))
datos_pruebas = datos_pruebas.batch(TAMANO_LOTE)
```

Código 15. Aceleración del Entrenamiento.

Para entrenar el modelo, se emplea la función 'fit' con los datos de entrenamiento. Se especifica la cantidad de épocas deseadas para iterar sobre los datos, obteniendo así una precisión en cada una de estas iteraciones (Código 16).

```
import math

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

Código 16. Entrenamiento del Modelo.

De igual manera, se procede a ver la función de pérdida en cada época definida (Código 17), utilizando 'Matplotlib' para graficar la magnitud de la pérdida en función al número de épocas.

```
plt.xlabel("# Epoca")
plt.ylabel("Magnitud de pérdida")
plt.plot(historial.history["loss"])
```

Código 17. Representar Función de Pérdida.

Después, se emplea 'Matplotlib' para mostrar 25 imágenes del conjunto de pruebas y predecir sus etiquetas utilizando el modelo. Las predicciones correctas se muestran en azul, las incorrectas en rojo. Además, se visualiza la confianza del modelo mediante barras que representan la probabilidad asignada a cada clase (Código 18).

```python
import numpy as np

for imagenes_prueba, etiquetas_prueba in datos_pruebas.take(1):
  imagenes_prueba = imagenes_prueba.numpy()
  etiquetas_prueba = etiquetas_prueba.numpy()
  predicciones = modelo.predict(imagenes_prueba)

def graficar_imagen(i,
                    arr_predicciones,
                    etiquetas_reales, imagenes):
  arr_predicciones, etiqueta_real, img = (
    arr_predicciones[i], etiquetas_reales[i], imagenes[i])

  plt.grid(False)
  plt.xticks([])
  plt.yticks([])

  plt.imshow(img[...,0], cmap=plt.cm.binary)

  etiqueta_prediccion = np.argmax(arr_predicciones)
  if etiqueta_prediccion == etiqueta_real:
    color = 'blue'
  else:
    color = 'red'

  plt.xlabel("{} {:2.0f}% ({})"
             .format(nombres_clases[etiqueta_prediccion],
                                100*np.max(arr_predicciones),
                                nombres_clases[etiqueta_real]),
                                color=color)

def graficar_valor_arreglo(i, arr_predicciones, etiqueta_real):
  arr_predicciones, etiqueta_real = (
    arr_predicciones[i], etiqueta_real[i])

  plt.grid(False)
  plt.xticks([])
  plt.yticks([])
  grafica = plt.bar(range(10), arr_predicciones, color="#777777")
  plt.ylim([0, 1])
  etiqueta_prediccion = np.argmax(arr_predicciones)

  grafica[etiqueta_prediccion].set_color('red')
  grafica[etiqueta_real].set_color('blue')

filas = 5
columnas = 5
num_imagenes = filas*columnas
plt.figure(figsize=(2*2*columnas, 2*filas))
for i in range(num_imagenes):
  plt.subplot(filas, 2*columnas, 2*i+1)
  graficar_imagen(i, predicciones, etiquetas_prueba, imagenes_prueba)
  plt.subplot(filas, 2*columnas, 2*i+2)
  graficar_valor_arreglo(i, predicciones, etiquetas_prueba)
```

Código 18. Funciones para el Matplotlib.

RESULTADOS

Los resultados del rendimiento del modelo de Red Neuronal Convolucional (CNN) se generaron mediante la ejecución del código 18. Esto incluye la representación de las predicciones del modelo y las probabilidades asociadas con esas predicciones. Por lo que, mediante las representaciones de cada imagen se puede identificar visualmente cómo el modelo clasifica las prendas de vestir, tanto acertadamente como erróneamente (Figura 6).

Después de completar cinco épocas de entrenamiento, se recopilaron los resultados que se presentan en la Tabla 2. Estos resultados muestran que el modelo de CNN logró alcanzar una precisión del 90.42% al evaluarlo con el conjunto de datos de prueba.

TABLA II
MÉTRICAS DE PRECISIÓN, PÉRDIDA Y TIEMPO POR ÉPOCA

Época	Precisión	Pérdida	Tiempo (s)
1	0.8900	0.2964	5 s
2	0.8956	0.2809	6 s
3	0.8972	0.2742	5 s
4	0.8989	0.2687	7 s
5	0.9042	0.2550	5 s

Los registros presentados en la Tabla 2 ilustran la progresión del modelo a lo largo de las cinco épocas de entrenamiento. Se destaca un avance gradual en la precisión del modelo, partiendo de un 89.00% en la primera época y alcanzando un 90.42% en la quinta. Al mismo tiempo, se observa una reducción constante en la pérdida, disminuyendo de 0.2964 en la primera época a 0.2550 en la quinta, como se refleja también en el gráfico correspondiente (Figura 7).

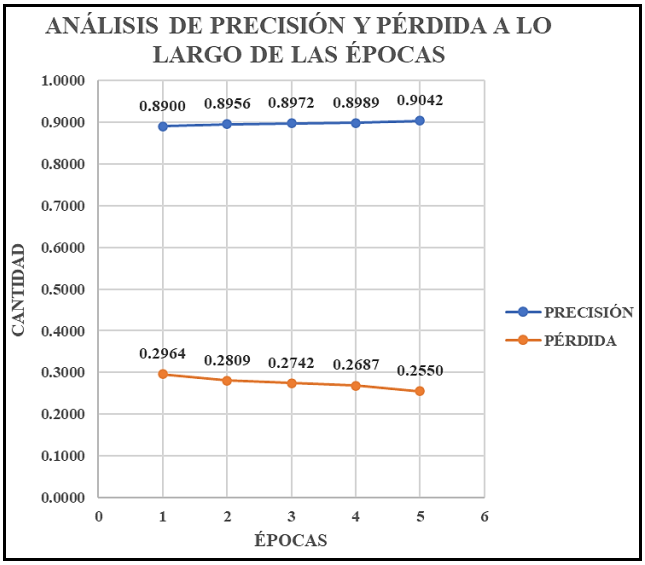

Figura 7: Gráfico de análisis de precisión y pérdida a lo largo de las épocas.

Cabe destacar que, a pesar del aumento en la precisión, el tiempo de ejecución no presenta una tendencia definida, fluctuando entre 5 y 7 segundos en diferentes épocas. Estos resultados indican una mejora significativa en la capacidad del modelo para adaptarse y ajustarse mejor a los datos de entrenamiento a medida que avanza a lo largo de las épocas.

De igual manera, la gráfica de pérdida (Figura 8) demostró una disminución gradual y constante durante el proceso de entrenamiento, esto nos indica la exitosa convergencia del modelo hacia un estado más preciso. Al examinar las predicciones realizadas sobre el conjunto de pruebas, se reveló un rendimiento altamente eficaz en la clasificación precisa de diversas prendas de vestir.

Figura 6. Clasificación de imágenes con nivel de precisión.

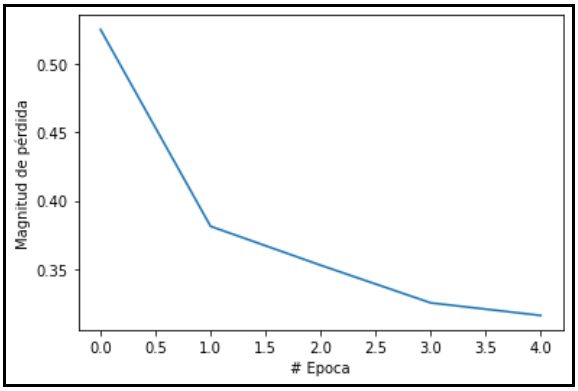

Figura 8: Gráfico de la función de pérdida del modelo.

Estos hallazgos subrayan la virtud del enfoque de Redes Neuronales Convolucionales para la identificación y clasificación de clases, como prendas de vestir a partir de imágenes, resaltando su capacidad para reconocer patrones complejos y su utilidad en aplicaciones de análisis de imágenes en el ámbito de la moda y áreas afines.

CONCLUSIONES

Las redes neuronales convolucionales tienen varios usos prácticos al ser capaces de identificar patrones complejos en grandes volúmenes de datos, el reconocimiento de imágenes es solo una de las tantas utilidades que tiene trabajar con sistemas de aprendizaje.

En este contexto, se optó por implementar un modelo de red neuronal convolucional para facilitar la identificación y clasificación precisa de diversas categorías de prendas de vestir. El modelo se caracteriza por emplear capas ocultas para identificar patrones distintivos en las imágenes de prendas de vestir, lo que permite una categorización efectiva.

Inicialmente, se desarrolló el código que destaca la importancia del preprocesamiento de datos en el aprendizaje automático. La normalización de los datos de imágenes de prendas de vestir (Fashion MNIST), escalando los valores de píxeles al rango de 0 a 1, resultó crucial para mejorar tanto la eficacia como la velocidad de convergencia del modelo de red neuronal. La arquitectura propuesta para el modelo consta de capas de aplanamiento, capas densas con activación ReLU y una capa final de clasificación con activación softmax, demostrando su efectividad en la clasificación multiclase de las diversas prendas de vestir. Esta estructura ofrece la adaptabilidad necesaria para capturar patrones complejos y realizar predicciones precisas.

Además, la elección del optimizador Adam y la función de pérdida de entropía cruzada categórica se reveló adecuada para este problema de clasificación. El entrenamiento con lotes de datos y su mezcla aleatoria contribuyeron a un aprendizaje más eficiente, evitando así el sobreajuste del modelo.

Los resultados del entrenamiento y la evaluación con el conjunto de prueba evidenciaron una mejora continua en la capacidad del modelo para clasificar con precisión las prendas de vestir. La alta precisión obtenida valida la eficacia de las

redes neuronales en la identificación precisa de las categorías de moda.

Por lo cual, se sugieren diversas aplicaciones potenciales en sistemas de recomendación de moda, catálogos digitales y reconocimiento de imágenes en comercio electrónico. La capacidad del modelo para diferenciar entre diversas prendas de vestir con alta precisión puede ser fundamental para mejorar la experiencia del usuario y la precisión en las recomendaciones.

REFERENCIAS

[1] Z. Gao and L. Han, "Clothing image classification based on random erasing and residual network," *Journal of Physics: Conference Series*, vol. 1634, no. 1, p. 012136, Sep. 2020, doi: 10.1088/1742-6596/1634/1/012136.

[2] G. San-Payo, J. C. Ferreira, P. Santos, and A. L. Martins, "Machine learning for quality control system," *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, no. 11, pp. 4491–4500, Nov. 2020, doi: 10.1007/s12652-019-01640-4.

[3] Á. Artola Moreno, "Clasificación de imágenes usando redes neuronales convolucionales en Python," Grado en Ingeniería de las Tecnologías de Telecomunicación, Universidad de Sevilla, Sevilla, 2019. [Online]. Available: https://idus.us.es/bitstream/handle/11441/89506/TFG-2402-ARTOLA.pdf?sequence=1&isAllowed=y

[4] M. Massiris, C. Delrieux, and J. Á. Fernández, "Detección de equipos de protección personal mediante red neuronal convolucional YOLO," in *Actas de las XXXIX Jornadas de Automática, Badajoz, 5-7 de septiembre de 2018*, Universidade da Coruña. Servizo de Publicacións, Mar. 2020, pp. 1022–1029. doi: 10.17979/spudc.9788497497565.1022.

[5] J. Xu, Y. Wei, A. Wang, H. Zhao, and D. Lefloch, "Analysis of Clothing Image Classification Models: A Comparison Study between Traditional Machine Learning and Deep Learning Models," *Fibres & Textiles in Eastern Europe*, vol. 30, no. 5, pp. 66–78, Oct. 2022, doi: 10.2478/ftee-2022-0046.

[6] A. Medina, J. I. Méndez, P. Ponce, T. Peffer, A. Meier, and A. Molina, "Using Deep Learning in Real-Time for Clothing Classification with Connected Thermostats," *Energies*, vol. 15, no. 5, p. 1811, Mar. 2022, doi: 10.3390/en15051811.

[7] X. Ding, T. Li, J. Chen, L. Ma, and F. Zou, "Research on the Clothing Classification of the She Ethnic Group in Different Regions Based on FPA-CNN," *Applied Sciences*, vol. 13, no. 17, p. 9676, Aug. 2023, doi: 10.3390/app13179676.

[8] Q.-Q. Li, Y.-Q. Zhong, and X. Wang, "Classification of Female Apparel using Convolutional Neural Network," *Journal of Fiber Bioengineering and Informatics*, vol. 11, no. 4, pp. 209–216, Dec. 2018, doi: 10.3993/jfbim00319.

[9] S.-K. Noh, "Deep Learning System for Recycled Clothing Classification Linked to Cloud and Edge Computing," *Computational Intelligence and Neuroscience*, vol. 2022, pp. 1–9, Nov. 2022, doi: 10.1155/2022/6854626.

[10] A. Marzal Varó, I. Gracia Luengo, and P. García Sevilla, *Introducción a la programación con Python 3*. Universitat Jaume I, 2014. doi: 10.6035/Sapientia93.

[11] F. de P. Quirós Pérez, "Reconocimiento de imágenes con TensorFlow desde R," TRABAJO FIN DE GRADO, Universidad de Sevilla, Sevilla, 2022. [Online]. Available: https://idus.us.es/bitstream/handle/11441/142916/DGME%20QUIROS%20PEREZ%2C%20FRANCISCO%20DE%20PAULA.pdf?sequence=1&isAllowed=y

[12] "TensorFlow Datasets: una colección de conjuntos de datos listos para usar." [Online]. Available: https://www.tensorflow.org/datasets?hl=es-419

[13] A. A. M. Teodoro *et al.*, "An Analysis of Image Features Extracted by CNNs to Design Classification Models for COVID-19 and Non-COVID-19," *Journal of Signal Processing Systems*, vol. 95, no. 2–3, pp. 101–113, Mar. 2023, doi: 10.1007/s11265-021-01714-7.