

Proyecto final.

Clasificador de frutas en imágenes y Clasificador de noticias.

Cesar Hormazabal- Felipe Farías - Lizette Arévalo

COM4402 – Introducción a Inteligencia Artificial
 Escuela de Ingeniería, Universidad de O'Higgins
 22, Diciembre, 2023

Abstract— En el siguiente documento se mostrará un análisis de los pasos del proceso al entrenar las bases de datos *Clasificación de frutas usando CNN* y *Clasificación de noticias textuales usando CNN Y RNN*. Aquí se describirán las teorías, metodologías, y resultados, para luego discutir y concluir al respecto, habiendo un enfoque práctico para la implementación de redes neuronales convolucionales, no se aplicará ningún Código adicional.

I. INTRODUCCIÓN

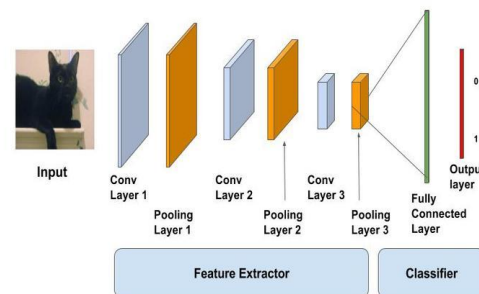
Este informe se centrará en explicar/ analizar, la implementación de Convolutional Neural Networks (CNN) Y Recurrent Neural Networks (RNN), en la clasificación de frutas y noticias.

Siendo el primer caso la *Clasificación de frutas al usar CNN*, en donde utilizando el Desarrollo de modelos de aprendizaje profundo se enfoca tanto en la clasificación de las frutas como en el control de calidad. Esto implica entrenar el modelo con un conjunto de datos que contiene imágenes de frutas etiquetadas, para que la red pueda aprender a reconocerla y clasificarla correctamente, mientras que por otro lado el objetivo principal es buscar ciertas características o defectos de las frutas (Respectivamente). A esto se le llama transferencia de aprendizaje, dado que se toma un modelo pre-entrenado y se adapta para realizar las tareas específicas, en este caso para clasificar.

En el segundo caso *Clasificación de noticias textuales usando CNN Y RNN*, dará el enfoque a que una red neuronal convolucional puede ser aplicada también a tareas de procesamiento de textos, específicamente en la capacidad de extracción de características mediante filtros convolucionales, los cuales aprenden patrones visuales a diferentes niveles.

II. MARCO TEÓRICO.

Clasificación de frutas al usar CNN



Arquitectura CNN

A. Red neuronal convolucional (CNN).

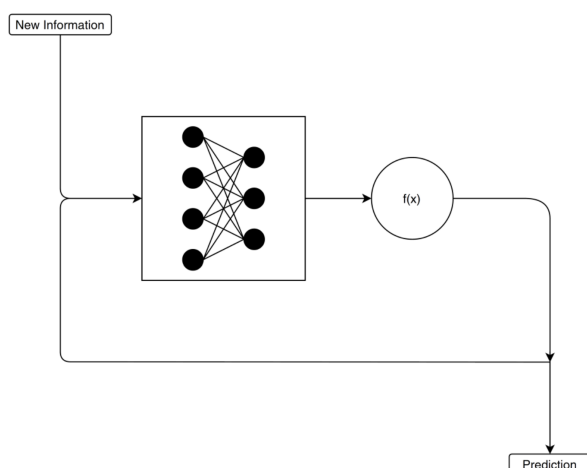
Es un tipo de red neuronal diseñada específicamente para el procesamiento de datos estructurados en forma de cuadrículas. Estas redes son altamente efectivas en tareas de visión por computadora, reconocimiento de patrones y procesamiento de imágenes.

- **Capa de entrada:** Se encarga de recibir los datos del mundo exterior y procesarlos antes de transmitirlos a la siguiente capa. Esta puede tener varias dimensiones, pero específicamente cuando son imágenes podría ser una matriz tridimensional.
- **Capa convolucional:** Extraen características de la entrada mediante operaciones de convolución. Pueden detectar patrones específicos gracias a sus filtros, como los bordes, texturas o formas.
- **Capa de aplanamiento:** Transforma la salida tridimensional de las capas en un vector unidimensional.
- **Capa completamente conectada:** Son neuronas conectadas entre sí, cada una ponderada a una capa anterior (Aplanamiento), ya que son las que realizan operaciones más complejas basadas en las características extraídas por las capas convolucionales.
- **Capa de salida:** Produce la salida final de la red neuronal, esta puede tener diferentes configuraciones, en este caso la clasificación de imágenes podría tener una

neurona por clase con una función de activación softmax para asignar probabilidades a cada clase.

- B. **Función Softmax:** transforma las salidas en una representación expresada como probabilidades, asegurando que la suma total de todas las probabilidades resultantes sea igual a 1.
- C. **cv2 (OpenCV):** OpenCV (Open Source Computer Vision) es una biblioteca para procesamiento de imágenes y visión por computadora. En este código, se utiliza para leer imágenes en escala de grises utilizando "cv2.imread" y realizar operaciones en las imágenes.
- D. **Momentum:** utiliza parámetros de momentos para crear la mayor optimización posible.
- E. **Dropout:** Dentro del código es utilizado para evitar un reajuste.
- F. **Activación ReLU:** Esta función transforma los valores anulando aquellos valores negativos del código y dejando los positivos tal como entran.
- G. **Tamaño de la imagen:** Tienen una resolución de 100x100 píxeles. Con 82213 imágenes de 120 clases.
- H. **Keras (tf.keras):** Interfaz de alto nivel para redes neuronales que facilita la construcción y entrenamiento de modelos. En este caso, se utiliza la implementación de Keras incluida en TensorFlow (tf.keras), que es la versión de Keras integrada en el paquete TensorFlow.
- I. **TensorFlow (tf):** Biblioteca de código abierto para aprendizaje automático y redes neuronales. Proporciona herramientas y recursos para construir y entrenar modelos de aprendizaje profundo. En este contexto, se utiliza la implementación específica de Keras incluida en TensorFlow (tf.keras) para definir y compilar el modelo de red neuronal convolucional (CNN).
- J. **Función de pérdida:** Es la que mide la discrepancia entre la salida de red y el resultado.

Clasificación de noticias textuales usando CNN Y RNN



Arquitectura RNN

- A. **CONV1D:** Capa convolucional unidimensional de redes neuronales, se adapta a datos secuenciales unidimensionales, en este caso el procesamiento de textos.
- B. **Kernel_size:** Tamaño del filtro que se desplaza a lo largo de la dimensión de la secuencia.
- C. **Capa TextVectorization de TensorFlow/Keras para vectorizar texto**
 - **max_features = 10000:** Establece el número máximo de tokens (palabras) a considerar al construir el vocabulario. En este caso, se ha fijado en 10,000.
 - **text_vectorizer = tf.keras.layers.TextVectorization(max_tokens=max_x_features):** Crea una capa de vectorización de texto utilizando Keras. Esta capa toma texto como entrada y realiza la vectorización, convirtiendo el texto en secuencias de números.
- D. **Modelo secuencial keras:**
 - Capa de vectorización:** El cual se encargará del preprocesamiento de los textos.
 - Capa de embedding:** Dentro del código sirve para representar los tokens en un vector de características entrenables en un espacio de alta dimensión.
 - A Conv1D:** Con este se procesan las secuencias
 - **Capa densa:** Es quien toma el vector y lo convierte en una única salida lógica.
- E. **Función Softmax:** transforma las salidas en una representación expresada como probabilidades, asegurando que la suma total de todas las probabilidades resultantes sea igual a 1.
- F. **Activación ReLU:** Esta función transforma los valores anulando aquellos valores negativos del código y dejando los positivos tal como entran.
- G. **Función de pérdida:** Es la que mide la discrepancia entre la salida de red y el resultado.
- H. **Redes neuronales recurrentes(RNN):** Tipo de arquitectura de redes neuronales diseñadas para trabajar con datos de secuencias

III. METODOLOGÍA

Clasificación de frutas al usar CNN

- 1) Cargamos las librerías correspondientes:

```
import os
import cv2
import random
import numpy as np
from matplotlib import pyplot as plt

from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, ConfusionMatrixDisplay

import tensorflow as tf
from tensorflow import keras

from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Dense, Conv2D, MaxPooling2D, Dropout, GlobalAveragePooling2D, Flatten
from tensorflow.keras.utils import to_categorical
from tensorflow.keras import optimizers
```

- 2) Luego el código carga el conjunto de datos de Fruit-360, el cual contiene 82213 imágenes de 100x100 píxeles con 120 clases de frutas y verduras.

- 3) Se establecen parámetros para escoger específicamente seis categorías del conjunto de datos guardando estos en “Fruits-Classifer”, lo anterior en un directorio “datasets/fruits-360”
- 4) El siguiente Código lee y carga las imágenes de entrenamiento (3074) de un conjunto de datos de las frutas, convirtiéndolas en escalas grises y almacenándolas en sus etiquetas de clases.

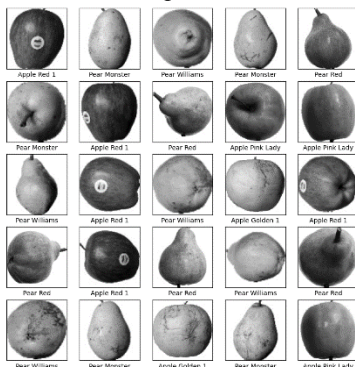
```

# Read training set
train_images = []
train_dir = os.path.join(base_dir, 'training/')

# set the training director

for category in CATEGORIES:
    path = os.path.join(train_dir, category)
    class_num = CATEGORIES.index(category)
    # Iterate to each category
    for image in os.listdir(path):
        # Iterate to each image in
        if (image.endswith('.jpg') and not image.startswith('.')):
            img_array = cv2.imread(os.path.join(path, image),
                                   cv2.IMREAD_GRAYSCALE)
            # read the image
            train_images.append([img_array, class_num])
            # save the image in trainin
print("Training images: ", len(train_images))
  
```

- 5) La biblioteca “os” se utiliza para realizar operaciones relacionadas con el Sistema operativo, como manipulación de rutas de directorios. CV2, se encarga de procesar las imágenes y visión por computadora. Por lo tanto lee las imágenes y realiza operaciones en las imágenes. Base_dir variable que almacena la ruta del directorio base del conjunto de datos de frutas, con categorías de clases.
- 6) Luego el Código selecciona aleatoriamente 25 imágenes de entrenamiento las cuales visualizará de 5x5 con sus etiquetas de clases.



- 7) Lo siguiente es que el Código baraja aleatoriamente el conjunto de datos entrenados para mejorar la precisión del modelo, para esto se hace uso de listas para almacenar las imágenes y etiquetas convirtiéndolas en matrices.
- 8) Se remodelan y normalizan las imágenes restando la media y dividiendo en 255, para finalmente imprimir la forma de los conjuntos de entrenamiento.

```

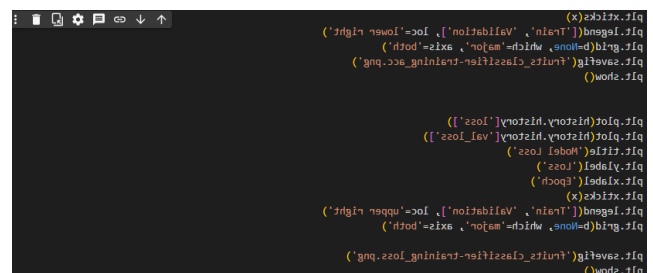
[ ] # reshape and normalize the data before training
x_train = x_train.reshape(-1, img_size, img_size, 1)
mean_train = np.mean(x_train, axis=0)
x_train = x_train - mean_train
x_train = x_train/255

x_test = x_test.reshape(-1, img_size, img_size, 1)
mean_test = np.mean(x_test, axis=0)
x_test = x_test - mean_test
x_test = x_test/255

y_train = np.asarray(y_train)
y_test = np.asarray(y_test)

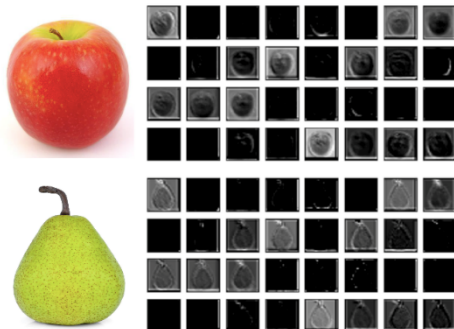
print(x_train.shape)
print(x_test.shape)
  
```

- 9) Además el Código establece hiper parámetros para la construcción de un modelo de red neuronal convolucional, haciendo uso de “Input_shape”, para la entrada de imágenes, “Filters_numbers” que representará al número de filtros por capa, “Filterssize” indicará el tamaño de dichos filtros, “Weight_decay” da el factor de decaimiento para la regularización L2, “Dropout” quien evita el sobreajuste, “Época” número de épocas de entrenamiento.
- 10) Luego el código define la arquitectura del CNN, utilizando la biblioteca Keras, esta incluye varias capas como de convolución, regulación, capa GlobalAveragePooling2D, Capa flatten, capa full connect, capa de salida, compilación del modelo y por último un resumen del modelo.
- 11) El código entrena el modelo y visualiza las curvas de entrenamiento, parte uno “Accuracy Plot” se grafica las curvas de precisión (Accuracy) del conjunto de entrenamiento y validación a lo largo de las épocas. “Loss Plot” se grafican las curvas de pérdida (loss) del conjunto de entrenamiento y validación a lo largo de las épocas, “Plt.Savefig” se encarga de guardar las visualizaciones de las imágenes.



- 12) Luego se testea el modelo imprimiendo los resultados.
- 13) Finalmente se genera y muestra una matriz de confusión normalizada para evaluar el rendimiento del modelo en el conjunto de prueba. Con “Confusion_matrix” el cual calcula la matriz de confusión de la function de scikit-learn, se normaliza usando el parámetro para mostrar tasas de clasificación en lugar de recuentos. “ConfusionMatrixDisplay” visualiza la matriz de

confusión, especificando las etiquetas de clase y la matriz normalizada. “Disp.plot” Graficar la matriz de confusión utilizando el objeto “disp”. Para obtener resultados, tal como se espera del ejemplo.



Clasificación de noticias textuales usando CNN Y RNN

Se divide en clasificación de noticias con CNN y la clasificación de texto combinando CNN y RNN

- 1) El conjunto de datos ag contiene 4 clases en un total de 120.000 muestras de noticias, mientras que el conjunto de prueba contiene 7.600 muestras, se tomarán solo el 10% de las noticias del conjunto de entrenamiento, cada clase contiene 30.000 muestras de entrenamiento y 1.900 muestras de prueba, y se irán pasando al conjunto de validación para aumentar un poco.
- 2) Se inicia la clasificación de noticias con CNN del conjunto de datos new_dataset, ag_news_subset. Este código utiliza TensorFlow Datasets para cargar el conjunto de los datos donde:
 - a) **tfds.load('ag_news_subset:1.0.0',):** Carga el conjunto de datos "ag_news_subset" en su versión 1.0.0. La función **tfds.load** es utilizada para cargar conjuntos de datos de TensorFlow Datasets.
 - b) **split con ['train[:90%]', 'train[90%:]']:** Divide el conjunto de datos en dos partes. El 90% se utiliza para entrenamiento (**train[:90%]**), y el 10% restante se divide entre datos de prueba y validación (**train[90%:] + test**).
 - c) **with_info=True:** Solicita que se devuelva información adicional sobre el conjunto de datos, incluyendo detalles sobre las etiquetas, características, etc.
 - d) **as_supervised=True:** Solicita que el conjunto de datos se cargue en un formato supervisado, proporcionando pares de

entrada y etiqueta. Esto facilita el entrenamiento de modelos supervisados.

- e) - **(train_data, val_data), info = ...:** Almacena los conjuntos de datos de entrenamiento y validación, así como la información adicional sobre el conjunto de datos en las variables

- 3) Lo siguiente es aplicar un código para entender la distribución de clases en el conjunto de datos y las categorías a las que pertenecen las noticias.

```
# Displaying the classes
class_names = info.features['label'].names
num_classes = info.features['label'].num_classes
print(f'The news are grouped into {num_classes} classes that are :{class_names}')
The news are grouped into 4 classes that are :['World', 'Sports', 'Business', 'Sci/Tech']
```

- 4) Se obtiene un número de ejemplos de entrenamiento y validación del conjunto de datos. Además se itera sobre las primeras 4 muestras del DataFrame “news_df” imprimiendo la información detallada de algunas noticias, incluyendo etiqueta, nombre de la clase y descripción de la noticia.

```
for i in range(0,4):
    print(f'Sample news {i}\n \
label: {news_df['label'][i]} {class_names[i]}\n \
Description: {news_df['description'][i]}\n-----\n')
```

- 5) Se establece un buffer_size de 1000 y un batch_size de 32. Para aplicar TextVectorizer, antes se configura el flujo de datos de entrenamiento y validación con “TensorFlowDatasets” Barajando los datos, para crear lotes con un tamaño específico y utilizando prefetching para optimizar el rendimiento del modelo.
- 6) Luego se convertirán los textos en tokens, eliminando puntuaciones y minúsculas. Se obtienen vocabularios en la lista de palabras individuales que componen una frase concreta. Se pasan algunas frases nuevas a “text_vectorizer”, y la secuencia vectorizadas se rellenan con el máximo de sentencias.
- 7) Se crea un modelo secuencial Keras que tomará los textos de entrada y salida de la clase de los textos de entrada, compuesta por capas: capas de vectorización de textos, capa de embedding, capa Conv1D y capa densa. Donde la capa Conv1D compone una capa de convolución y agrupación.

Así, el código define el modelo de red neuronal secuencial de TensorFlow/Keras para clasificar el texto. Aquí se combina tanto la técnica del procesamiento de textos, como convoluciones y redes completamente conectadas para realizar la clasificación de textos.

```
model = tf.keras.Sequential([
    text_vectorizer,
    tf.keras.layers.Embedding(input_dim=input_dim, output_dim=64, mask_zero=True),
    tf.keras.layers.Conv1D(64, 5, activation='relu'),
    tf.keras.layers.MaxPooling1D(),
    tf.keras.layers.Conv1D(64, 5, activation='relu'),
    tf.keras.layers.GlobalMaxPool1D(),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Dense(4, activation='softmax')
])
```


- 8) Luego la función “Plot_model” del módulo “tensorflow.keras.utils” visualiza el diagrama de la arquitectura del modelo en un formato gráfico. Este nos permitirá mostrar la disposición y conexión de las capas en el modelo de clasificación de texto. Se establecen los parámetros de cómo el modelo se evalúa y optimiza durante el proceso de aprendizaje. Con un batch_size de 32, calcula el número de pasos por época durante el entrenamiento y validación. Además se entrena el modelo utilizando el conjunto de datos de entrenamiento “train_data” y valida el conjunto de datos de validación con “val_data” con 25 épocas
- 9) Así, luego de compilar el modelo, visualizamos los resultados. La función “plot_acc_loss” visualiza las curvas de entrenamiento y validación de precisión (accuracy) y pérdida (loss) a lo largo de la época.

```
import matplotlib.pyplot as plt
# function to plot accuracy and loss
def plot_acc_loss(history):
    model_history = history.history
    acc = model_history['accuracy']
    val_acc = model_history['val_accuracy']
    loss = model_history['loss']
    val_loss = model_history['val_loss']
    epochs = history.epoch

    plt.figure(figsize=(10,5))
    plt.plot(epochs, acc, 'b', label='Training Accuracy')
    plt.plot(epochs, val_acc, 'r', label='Validation Accuracy')
    plt.title('Training and Validation Accuracy')
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy')
    plt.legend(loc=0)

    # create a new figure with plt.figure()
    plt.figure()
    plt.figure(figsize=(10,5))
    plt.plot(epochs, loss, 'b', label='Training Loss')
    plt.plot(epochs, val_loss, 'r', label='Validation Loss')
    plt.title('Training and Validation Loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend(loc=0)
    plt.show()
```

Clasificación de texto combinando CNN y RNN

1. Para la segunda parte de la clasificación de noticias. Se define un modelo de red neuronal secuencial en TensorFlow/Keras para la clasificación de textos, que combina capas convolucionales y una capa LSTM bidireccional, con las capas “text_vectorizer”, “Embedding” con una dimension de salida 64, dos capas de “Conv1D (64, FILTROS 5 en activación ReLU”, Dense(32 unidades y activación Softmax) y la capa de LSTM “Bidirectional”, etc.

```
conv_rnn_model = tf.keras.Sequential([
    text_vectorizer,
    tf.keras.layers.Embedding(input_dim=input_dim, output_dim=64, mask_zero=True),
    tf.keras.layers.Conv1D(64, 5, activation='relu'),
    tf.keras.layers.MaxPooling1D(),
    tf.keras.layers.Conv1D(64, 5, activation='relu'),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64)),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Dense(4, activation='softmax')
])
```

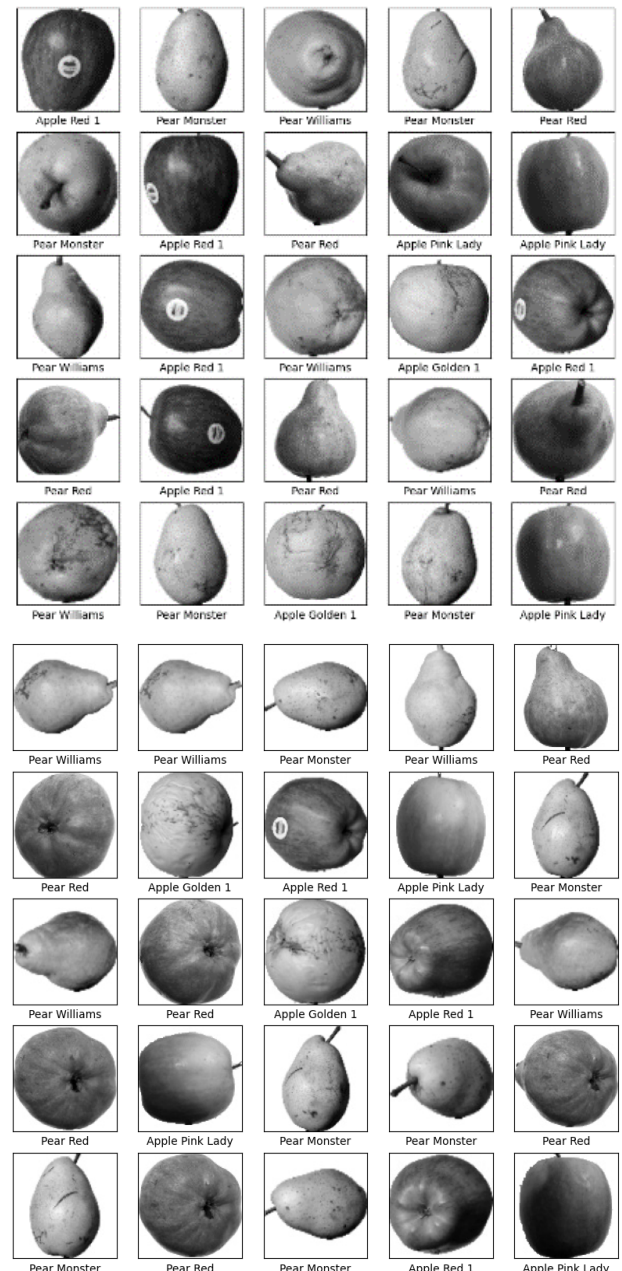
2. De manera análoga al proceso de clasificacion de texto solo con cnn, se entrena el modelo “conv_rnn_model” utilizando el conjunto de datos de entrenamiento “train_data” y validación de “val_data” con 25 épocas y batch_size=32. El historial capturará la información sobre la pérdida y presión en cada época, lo que permite analizar el rendimiento del modelo a lo largo del texto del tiempo. Se utiliza para visualizar curvas de

aprendizajes y diagnosticar el rendimiento del modelo.

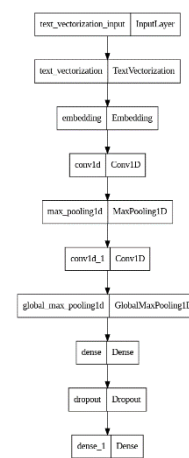
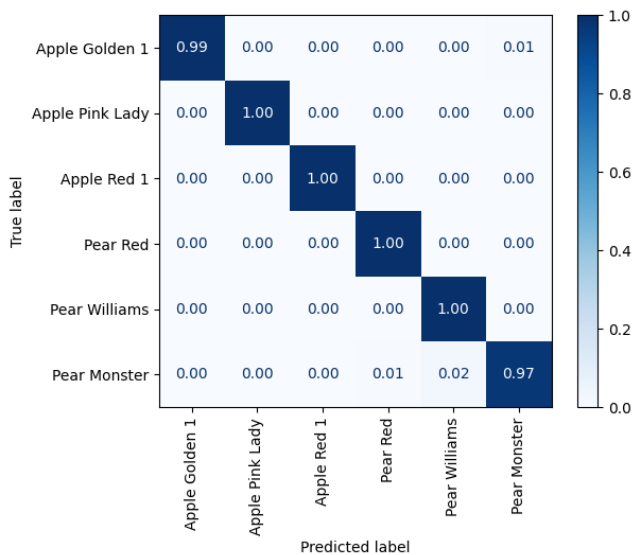
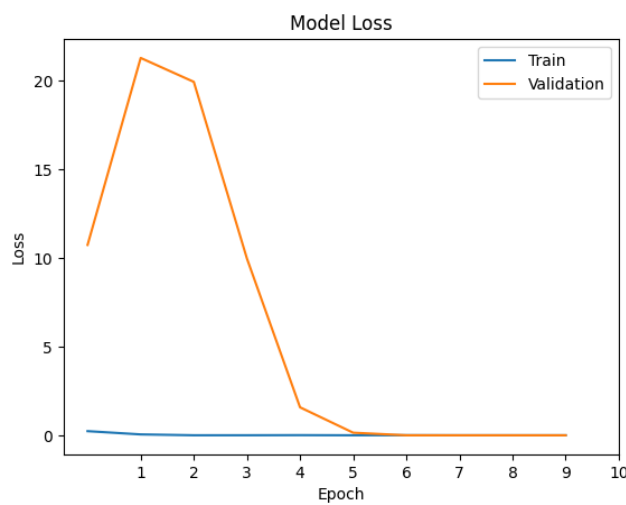
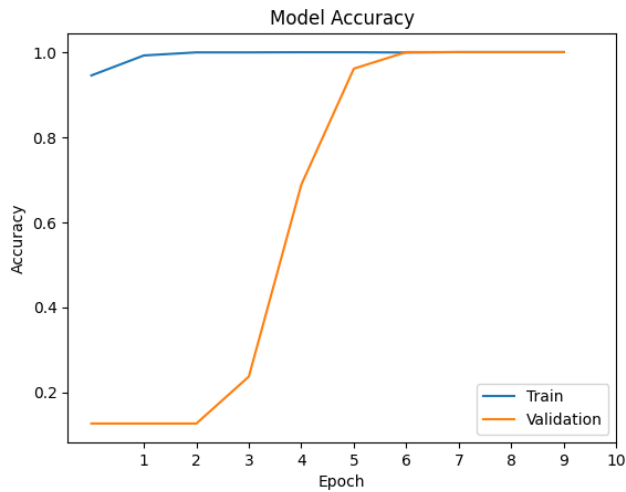
3. Finalmente se plotean los resultados de la accuracy y la pérdida para los conjuntos de entrenamiento y validación. Definiendo también una función predict() para probar la red con nuevos textos

IV. RESULTADOS.

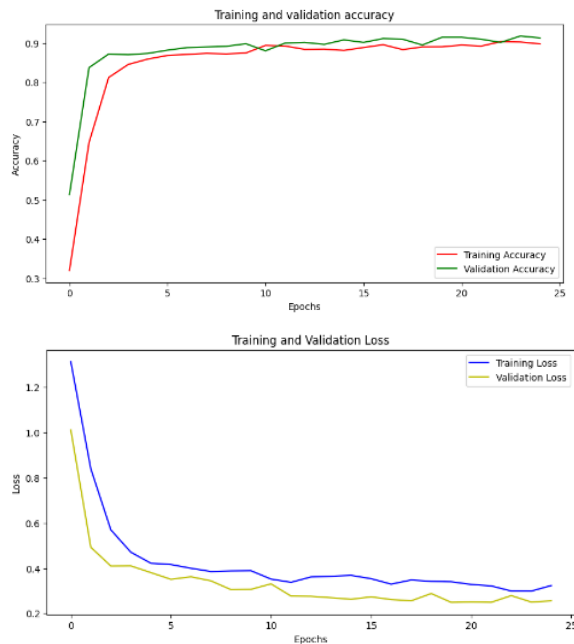
Clasificador de frutas con CNN.



Clasificación de noticias textuales usando solo CNN



Clasificación de noticias textuales usando CNN y RNN



V. ANÁLISIS.

En términos generales, la ejecución de los códigos muestra un desempeño sólido, proporcionando resultados precisos en tiempos de ejecución eficientes. Vamos a analizar detalladamente el rendimiento de cada modelo.

Clasificación de Frutas con CNN:

El clasificador de frutas, implementado mediante una red neuronal convolucional (CNN), destaca por su capacidad para clasificar con precisión las diversas cepas de frutas predefinidas. A pesar de recibir imágenes en escala de grises, el modelo logra un alto nivel de exactitud al categorizar los seis modelos de frutas contemplados.

Clasificación de Noticias con CNN y CNN+RNN:

En el análisis de noticias, el modelo basado en CNN para la clasificación de categorías muestra un rendimiento notable, logrando una alta precisión y una pérdida inicial baja. La introducción de una combinación de CNN y redes neuronales recurrentes (RNN) mejora aún más el rendimiento, aunque con una precisión ligeramente menor en comparación con la clasificación de frutas.

Se observa que la diferencia entre el uso exclusivo de CNN y la combinación de CNN+RNN es mínima al principio, pero la última supera rápidamente a la CNN sola con una mayor precisión en pocas épocas. Además, la combinación CNN+RNN demuestra ser más eficiente y menos costosa de entrenar en comparación con el modelo exclusivo de RNN.

En conclusión, ambos modelos demuestran ser eficaces en sus respectivas tareas, y la estrategia de combinar CNN y RNN destaca como una solución eficiente para mejorar la clasificación de noticias en comparación con el uso exclusivo de RNN. La implementación exitosa de estos modelos abre oportunidades para aplicaciones más amplias en el ámbito de la clasificación de imágenes y textos.

VI. CONCLUSIÓN Y DISCUSIÓN.

En líneas generales, las clasificaciones demostraron un alto nivel de precisión (accuracy), logrando clasificar eficazmente grandes volúmenes de datos de manera precisa y eficiente. Se destaca que los resultados, podrían haber variado al utilizar otros modelos o parámetros en las redes que utilizamos, lo cual refleja la flexibilidad y adaptabilidad de las redes y códigos implementados. Aunque en estos casos se eligió el modelo más eficiente para el desarrollo actual, la capacidad de adaptación permite explorar diferentes enfoques(modelos) según las necesidades específicas. Este enfoque estratégico en la elección de modelos contribuye a la eficiencia general del sistema, asegurando resultados precisos en un tiempo de ejecución optimizado

VII. REFERENCES

- [1] Ignacio Bugueño, (Noviembre- Diciembre), Clases finales.
- [2] Gavilan, I. (2020, May). *Catálogo de componentes de redes neuronales: Funciones de pérdida* , <https://ignaciogavilan.com/catalogo-de-componentes-de-redes-neuronales-iii-funciones-de-perdida/#:~:text=%C2%BFQu%C3%A9%20es%20una%20funci%C3%B3n%20de,es%20el%20correcto%20o%20deseado.>