Proyecto Semestral

Clasificador de frutas en imágenes y Clasificador de noticias

Luis Albandoz - Nicolás Barrera - Sebastián Garrido - José García

COM4402 – Introducción a Inteligencia Artificial

Escuela de Ingeniería, Universidad de O'Higgins

22 de diciembre, 2023

***Resumen*— El proyecto elegido es el proyecto 2 que consiste en clasificar frutas en imágenes utilizando Redes Neuronales Convolucionales (CNN) y en clasificar noticias utilizando CNN y Redes Neuronales Recurrentes (RNN).**

Para la clasificación de frutas en imágenes se utiliza el conjunto de datos Fruit-360 que contiene 82213 imágenes (100 x 100 pixeles) de frutas y verduras de 120 clases, ya subdivididas en conjunto de entrenamiento y prueba. Se eligen 6 categorías del conjunto de datos en el cual se observan 3 clases de manzanas y 3 categorías de peras: Apple Golden 1, Apple Pink Lady, Apple Red 1, Pear Red, Pear Williams y Pear Monster. Posteriormente se utiliza la matriz de confusión para medir la exactitud de los datos (*accuracy*).

Por otro lado, en el clasificador de noticias se utilizan el conjunto de datos news dataset, ag_news_subset que se encuentran disponibles en TensorFlow datasets, por lo que se requiere importar estas librerías. En el conjunto de entrenamiento hay 120000 muestras de noticias y el de validación 7600, por lo cual contiene 4 clases: world (0), sports (1), business (2) y Sci/Tech (3). Cada clase contiene 30000 muestras de entrenamiento y 1900 muestras de prueba. Posterior a ello se crea y se entrena el modelo en 25 épocas (*epochs*) y se visualiza a través de los gráficos el entrenamiento del conjunto de entrenamiento y de prueba en 10 épocas.

***Palabras claves*— Redes Neuronales Convolucionales (CNN), Redes Neuronales Recurrentes (RNN), conjunto de entrenamiento, conjunto de prueba, epochs.**

I. INTRODUCCIÓN

Las Redes Neuronales Convolucionales (CNN) son una clase de Red Neuronal en el aprendizaje profundo (Deep Learning) aplicando con mayor frecuencia en el análisis de imágenes visuales y en la clasificación de textos. También se conoce como redes neuronales artificiales invariantes al desplazamiento o espacio, basada en la arquitectura de los pesos compartidos de los filtros de convolución que se deslizan a lo largo de las características de entrada proporcionando respuestas equivariantes a la traslación, conocida como feature maps. Por su parte, las Redes Neuronales Recurrentes (RNN) se aplican principalmente en el reconocimiento de voz, traducción de idiomas y subtítulos en imágenes permitiendo procesar datos secuenciales.

II. MARCO TEÓRICO

Según Sotaquirá (2019) señala que "al igual que en las Redes Neuronales, las Redes Convolucionales también permiten detectar patrones en los datos de entrada, con la única diferencia de que en el caso de las Redes Convolucionales los datos de entrada son

imágenes" (párr. 4). A partir de la afirmación anterior se puede mostrar que las Redes Convolucionales logran el procesamiento de imágenes en las diferentes capas de la neurona, permitiendo transformar a valores numéricos.

Rapu (2023) señala que una convolución "consiste en deslizar un pequeño filtro o kernel sobre la imagen de entrada, calculando el producto escalar entre estos, para construir un mapa de características" (p.3). De esta expresión se puede afirmar que la convolución durante el procesamiento de la imagen permite obtener un valor escalar en el cual se almacena en un mapa, lo que permite obtener información relevante; por lo que el paso posterior a este será el procesamiento a las siguientes capas de la neurona.

Parada (2022) establece que "las capas convolucionales aplican distintos filtros sobre una imagen de entrada y crean nuevos volúmenes, por lo que las propiedades espaciales de la imagen se mantienen" (párr. 12). Con lo anterior se puede señalar, que las capas convolucionales pueden aplicar y modificar filtros en la imagen permitiendo mantener las propiedades iniciales de las imágenes antes de ser procesadas.

Sotaquirá (2019) afirma que las Redes Neuronales Recurrentes "son una de las principales arquitecturas del Machine Learning, muy usadas por ejemplo en los sistemas de reconocimiento de voz o en el análisis video, o en el procesamiento del lenguaje natural" (párr. 3). A partir de esta afirmación se puede demostrar que las Redes Recurrentes son usadas principalmente en aplicaciones de reconocimiento de voz, análisis y procesamiento de videos.

A través del portal Gamco (2021) señala que "las RNN tienen una estructura recurrente, que permite que la información fluya de una capa a otra a través de un estado oculto, que almacena información sobre los estados anteriores" (párr. 3). Según lo señalado se afirma que la estructura de una RNN presenta un estado oculto que va transmitiendo a la siguiente capa a partir de la capa anterior; esto permite que la Red Recurrente tenga memoria a largo plazo permitiendo reconocer y capturar patrones de secuencia de datos a lo largo del tiempo.

III. METODOLOGÍAS

Para la clasificación de frutas se diseña una Red Convolucional (CNN) con una arquitectura simple compuesta por una capa de entrada, tres capas convolucionales, una capa de aplanamiento, una capa completamente conectada y una capa de salida.

Se utiliza TensorFlow como biblioteca que permite entrenar y evaluar la Red Convolucional. Se utiliza la base de datos Fruit-360, cuyo origen es a través de la plataforma GitHub permitiendo analizar bajo la plataforma de Google Colab. Ya descargado el conjunto de datos se crea el parámetro "*CATEGORIES*" trabajando con las distintas clases de manzanas (Apple Golden 1, Apple Pink Lady, Apple Red 1) y peras (Pear Red, Pear Williams, Pear Monster) resultando 6 clases y en tamaño de 100 x 100 pixeles.

El siguiente procedimiento consiste en leer el conjunto de datos de entrenamiento que contienen imágenes bajo el formato JPG donde se convierten en escala de grises y se almacenan junto con las etiquetas de clase denominada "*train_images*", resultando 3074 elementos que pertenecen al conjunto de entrenamiento, mostrando en la Fig. 1

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

for category in CATEGORIES:
    path = os.path.join(train_dir, category)
    class_num = CATEGORIES.index(category)
    for image in os.listdir(path):
        if(image.endswith('jpg') and not image.startswith('.')):
            img_array = cv2.imread(os.path.join(path,image),
                                   cv2.IMREAD_GRAYSCALE)
            train_images.append([img_array, class_num])

print("Training images: ", len(train_images))
```

```
Training images:  3074
```

Fig. 1 Conjunto de datos de entrenamiento Fruit-360

En la Fig. 2 se muestra el mismo procedimiento anterior, para esta ocasión se trabaja con el conjunto de prueba donde las imágenes bajo el formato JPG son convertidas en escala de grises almacenando en la variable "*test_images*" resultando 1030 elementos.

```
# Read testing set
test_images = []
test_dir = os.path.join(base_dir, 'Test/')

for category in CATEGORIES:
    path = os.path.join(test_dir, category)
    class_num = CATEGORIES.index(category)
    for image in os.listdir(path):
        if(image.endswith('jpg') and not image.startswith('.')):
            img_array = cv2.imread(os.path.join(path,image),
                                   cv2.IMREAD_GRAYSCALE)
            test_images.append([img_array, class_num])

print("Testing images: ", len(test_images))
```

```
Testing images:  1030
```

Fig. 2 Conjunto de datos de prueba Fruit-360

Se crea un nuevo cuadro de código mostrado en la Fig. 3 que consiste en mezclar (*shuffle*) los conjuntos de datos de entrenamiento y prueba con el objetivo de mejorar la exactitud (*accuracy*) del modelo durante el entrenamiento organizando características y etiqueta en listas y matrices NumPy respectivamente, preparadas para utilizar en el entrenamiento y en la evaluación del modelo.

```
# Shuffle the dataset before training for better accuracy
x_train = []
y_train = []

random.shuffle(train_images)

for features, label in train_images:
  x_train.append(features)
  y_train.append(label)
x_train = np.array(x_train)

x_test = []
y_test = []

random.shuffle(test_images)

for features, label in test_images:
  x_test.append(features)
  y_test.append(label)
x_test = np.array(x_test)
```

Fig. 3 Mezcla (shuffle) entre los conjuntos de entrenamiento y prueba

Posteriormente se muestra en la Fig. 4 el preprocesamiento que consiste en preparar y normalizar los conjuntos de datos de entrenamiento y prueba antes de ser utilizados en el entrenamiento de la Red Convolucional.

```
# 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)
```

Fig. 4 Preprocesamiento conjunto de datos Fruit-360

Se comienza a diseñar la arquitectura de la Red Neuronal Convolucional utilizando la biblioteca Keras. La Red Neuronal presenta una capa de entrada, tres capas convolucionales completamente conectadas; cada capa convolucional utiliza la función de activación ReLU. Esta Red Neuronal utiliza la función de pérdida entropía cruzada categórica dispersa (*Sparse Categorical Crossentrophy*) y el optimizador SGD (*Stochastic Gradient Descent*) con momentum.

Con la Red Neuronal ya diseñada se comienza a entrenarse en un periodo de 10 épocas (*epochs*) y se crea el conjunto de validación definido en un 10% del conjunto de entrenamiento.

En la etapa posterior del entrenamiento se realizan los gráficos "*model accuracy*" y "model loss" en 10

épocas evaluando la precisión y la pérdida en los conjuntos de entrenamiento y validación.

El paso final en la clasificación de frutas, se realiza la matriz de confusión utilizando los conjuntos de validación y de prueba para medir y verificar que tan exactos son medidos los datos.

Para la clasificación de noticias se utilizan las bases de datos "*news dataset", "ag news subset"* que se encuentran en el conjunto de datos de TensorFlow, utilizando las Redes Convolucionales (CNN) y las Redes Recurrentes (RNN). Procesados los conjuntos de datos se muestran las clasificaciones a los diferentes tipos de noticias, encontrando 4 clases de noticias: mundo (*world: 0*), deportes (*sports: 1*), negocios (*business: 2*) y ciencias/tecnología (*Sci/Tech: 3*).

El siguiente paso es obtener el número de ejemplos en los conjuntos de entrenamiento y validación, por lo que se muestran la cantidad total de cada conjunto, 120000 muestras en el conjunto de entrenamiento y 7600 elementos por el conjunto de validación.

Al obtener ejemplos se visualizan las primeras 10 muestras de noticias como se muestran en la Fig. 5

	description	label
0	b'AMD #39;s new dual-core Opteron chip is desi...	3
1	b'Reuters - Major League Baseball\\Monday anno...	1
2	b'President Bush #39;s quot;revenue-neutral q...	2
3	b'Britain will run out of leading scientists u...	3
4	b'London, England (Sports Network) - England m...	1
5	b'TOKYO - Sony Corp. is banking on the \\$3 bi...	0
6	b'Giant pandas may well prefer bamboo to lapto...	3
7	b'VILNIUS, Lithuania - Lithuania #39;s main pa...	0
8	b'Witnesses in the trial of a US soldier charg...	0
9	b'Dan Olsen of Ponte Vedra Beach, Fla., shot a...	1

Fig. 5 Visualizaciones de las primeras 10 noticias

En la Fig. 6 se muestra el siguiente procedimiento que consiste en preparar los datos de entrenamiento y validación agrupando en lotes (*batch*) de máximo 32 y un tamaño buffer 1000, vectorizando los textos o convirtiéndose en secuencias o tokens numéricos utilizando la biblioteca *Keras Text Vectorization* mezclando el conjunto de datos de validación con los datos de prueba.

```
buffer_size = 1000
batch_size = 32

train_data = train_data.shuffle(buffer_size)
train_data = train_data.batch(batch_size).prefetch(1)
val_data = val_data.batch(batch_size).prefetch(1)
```

Fig. 6 Preparación de datos de entrenamiento y validación

Se define la Arquitectura del modelo de Red Neuronal utilizando la API '*Sequential*' de TensorFlow y Keras. La arquitectura (modelo) que aparece en la Fig. 7 está compuesto por una capa de vectorización que sirven preprocesar textos; una capa de embedding que representan los tokens en un vector de características entrenables de un espacio de alta dimensión; dos capas de convolución 1D, cada capa convolucional utiliza la función de activación ReLU con 64 filtros y una ventana (kernel) de tamaño 5; capas de max pooling 1D y capas densas. La primera capa densa presenta 32 unidades y utiliza la función ReLU, en cambio, la segunda capa densa es una capa de salida con 4 unidades (se asume debido a que hay 4 clases) y una activación '*softmax*' permitiendo obtener valores normalizados.

```
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')
])
```

Fig. 7 Arquitectura Red Neuronal secuencial Keras

El modelo es entrenado en un lapso de 25 épocas en el cual se utilizan los conjuntos de entrenamiento y de validación para evaluar la exactitud (accuracy) y la pérdida (loss).

En la Fig. 8 se muestra una nueva arquitectura de la Red Neuronal donde se combinan las CNNs y RNNs para volver a entrenar la Red Neuronal, pero en este modelo se presenta un pequeño ajuste que consiste en aplicar una capa LSTM bidireccional que permite manejar las características de las CNNs desde ambas direcciones.

```
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')
])
```

Fig. 8 Arquitectura Red Neuronal secuencial Keras con capa LSTM bidireccional

IV. RESULTADOS Y DISCUSIONES

Considerando la metodología anteriormente explicada, ahora consta exhibir los resultados y sus respectivas discusiones:

Código 1 (Clasificación de frutas):

Hay que tener en cuenta que, en este código, se diseña una CNN (Redes neuronales convolucionales), cuyo desarrollo se presta para diversos usos, en este caso en particular, para clasificar frutas y asignar un control de calidad.

En primer lugar, se cargan 2 sets de imágenes que se clasifican como conjuntos de entrenamiento (Fig. 9) y prueba (Fig. 10), del cual en el primero se guardan 3074 elementos pertenecientes al conjunto de entrenamiento, y en el conjunto de prueba, 1074 elementos.

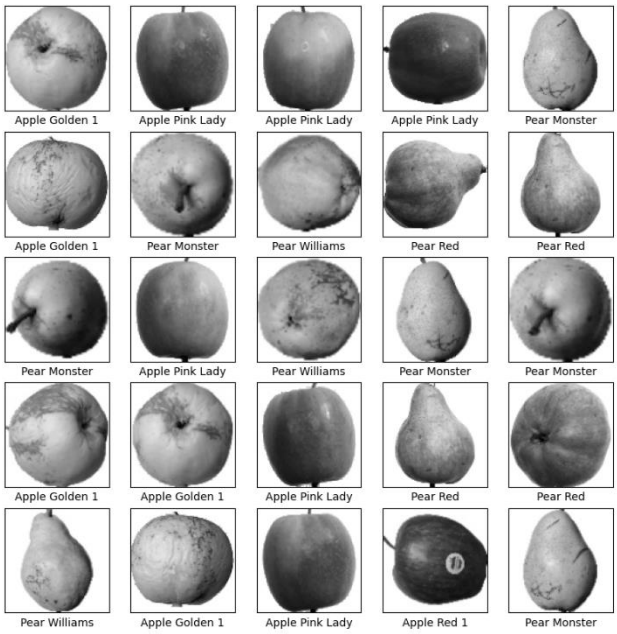

Fig. 9 Conjunto de entrenamiento Fruit-360

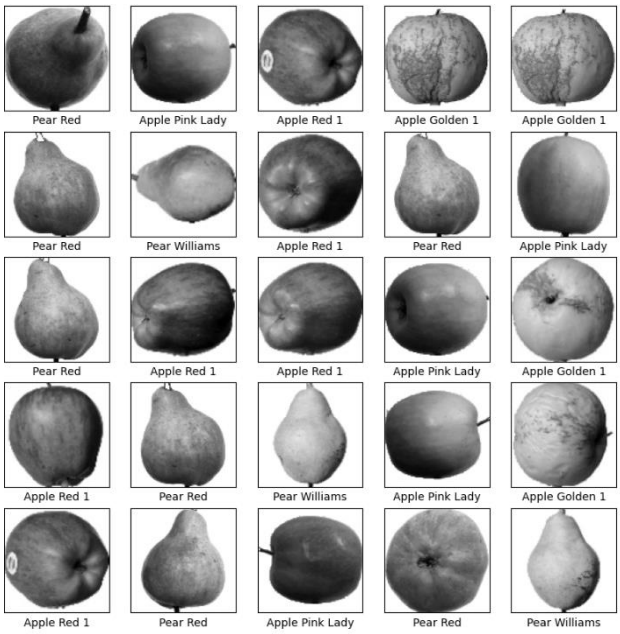

Fig. 10 Conjunto de prueba Fruit-360

Luego de aquel procedimiento, se procede a entrenar el mismo modelo, cuyos resultados (gráficos) que se muestran en la Fig. 11 evidencian

una precisión sostenida para ambas variables de entrenamiento y validación, cuyo último parámetro incrementa al máximo una vez llegada a la cuarta época. Por otro lado, la pérdida observada del modelo arroja para la variable de entrenamiento, una pérdida estable definida en 0 a partir de la primera época, y algo errática para la variable de validación, puesto que incrementa (y sostiene) una pérdida alta en las primeras 2 épocas, para luego disminuir abruptamente a 0 llegada la quinta iteración (época).

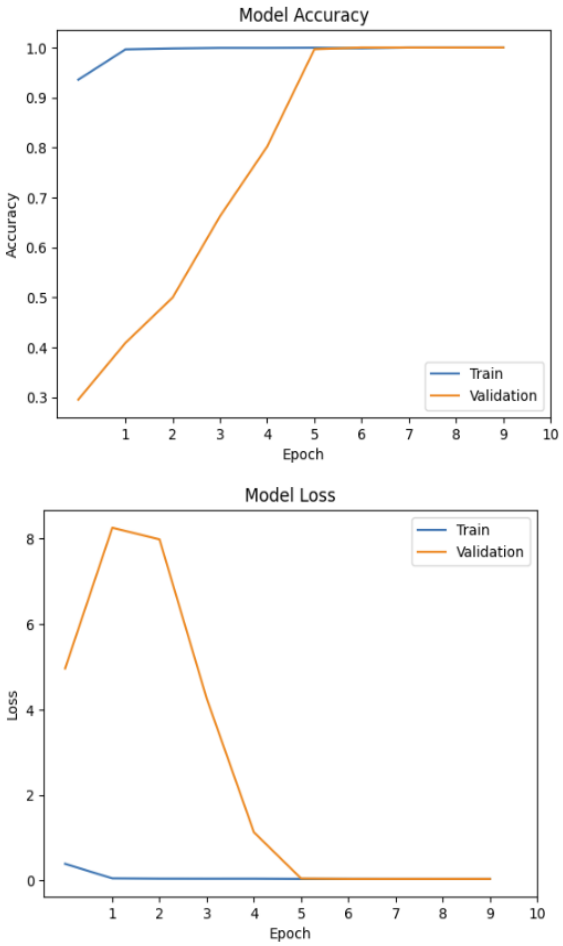

Fig. 11 Resultados modelo de accuracy y pérdida en conjuntos de entrenamiento y validación

Con esto en mente, se puede rescatar que los valores otorgados son efectivos y precisos, sobre todo al observar que el entrenamiento tuvo cifras positivas en su revisión. En la Fig. 12 se muestra el valor de la exactitud (accuracy) obteniendo un 0,98:

Accuracy 0.9893

Classification report

	precision	recall	f1-score	support
Apple Golden 1	1.00	1.00	1.00	160
Apple Pink Lady	1.00	0.94	0.97	152
Apple Red 1	1.00	1.00	1.00	164
Pear Red	0.96	1.00	0.98	222
Pear Williams	0.99	1.00	0.99	166
Pear Monster	1.00	0.99	0.99	166
accuracy			0.99	1030
macro avg	0.99	0.99	0.99	1030
weighted avg	0.99	0.99	0.99	1030

Fig. 12 Matriz de confusión conjunto entrenamiento

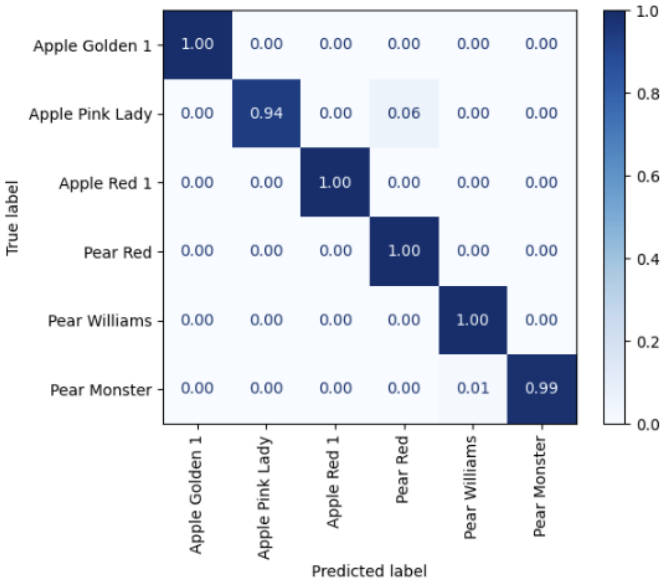

Fig. 13 Matriz de confusión

A partir de la matriz de confusión que aparece en la Fig. 13 se puede distinguir que la repetición de etiquetas en sus iteraciones para las frutas "*Apple Pink Lady*" y "*Pear Monster*" respectivamente, no resultan ser 1.00, sino que 0.94 y 0.99 en cada caso, indicando con esto una ligera imprecisión en el primer escenario.

Dicho esto, se puede determinar que el modelo efectivamente produce los resultados esperados de clasificación de frutas, siendo fiel a lo que se interpreta en el ejemplo que aparece en la Fig. 14

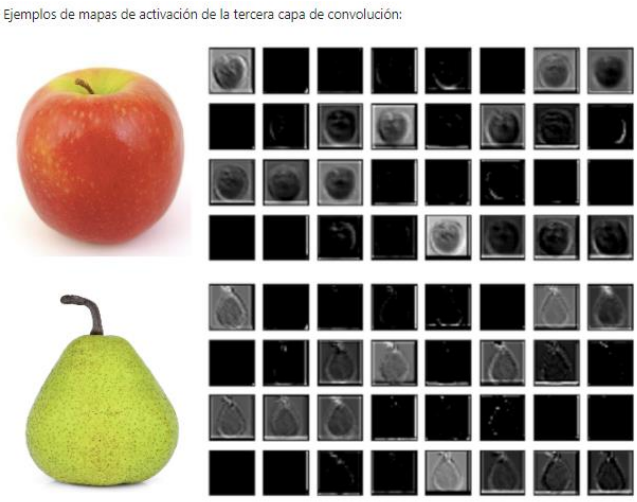

Fig. 14 Ejemplo de clasificación de frutas

Demostrando que la red convolucional permitirá clasificaciones confiables que, en lo gráfico y para todas las frutas involucradas con sus respectivas combinaciones, se destacarán las imágenes más cercanas y precisas a la figura de la fruta en cuestión.

Código 2 (Clasificación de noticias):

En un enfoque diferente se desarrolló paralelamente una CNN (Red neuronal convolucional) que se aplicaba a texto, específicamente noticias y la categoría específica a la que estas pertenecen sea mercado, deporte, tecnológicas e internacionales.

Primero cargamos el dataset un fragmento de una base extensa, con esta obtenemos nuestros conjuntos de entrenamiento, prueba y validación normalmente nuestro conjunto tiene 120.000 entradas de entrenamiento y 7600 de prueba, pero dedicaremos un 10% de entrenamiento para validación.

Pasamos a procesar nuestros datos vectorizando nuestros textos, dejándolos como tokens numéricos en secuencias.

Una vez efectuados estos pasos, empezamos la creación de nuestro modelo, en el cual encontramos buena precisión en el transcurso de las épocas, sobresalientemente para validación sobre entrenamiento está compartiendo mejoras significativas hasta la época 3, para luego ser graduales llegando a su mayor precisión durante la época 21, en tanto a la pérdida apreciamos como está inversamente disminuyendo gradualmente y oscilando entre resultados un poco mejores o peores en épocas terminales pero siempre existiendo.

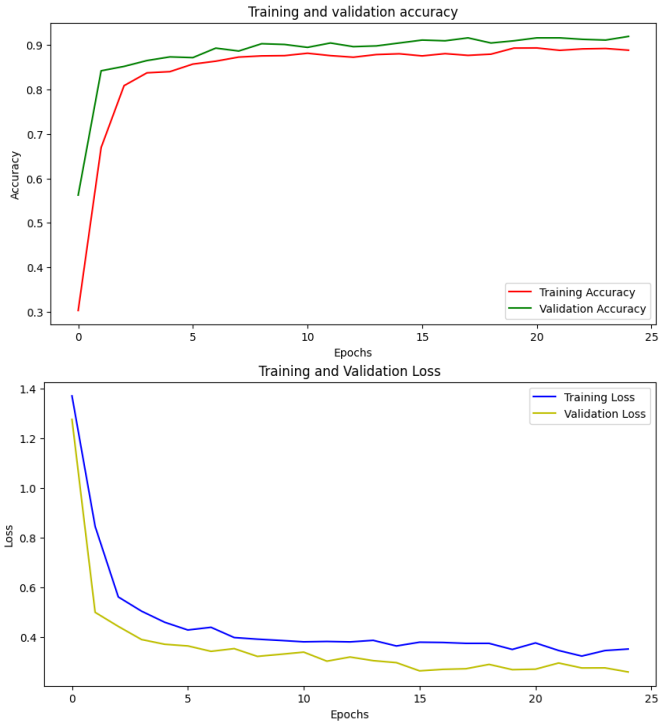

Fig. 15 Gráficos entrenamiento validación accuracy y validación loss

Ya con los gráficos que aparecen en la Fig.15, a pesar de que los resultados fueron aceptables creímos que existía un margen de mejora por lo que se aplicó RNN (Red Neuronal Recurrente) para ayudar a la CNN en el contexto de las secuencias, añadiendo como una capa extra, una vez hecho esto encontramos una ligera mejora general en todos los parámetros estos siguiendo los mismos patrones anteriores. En la Fig. 16 se muestran los siguientes gráficos de la validación de accuracy y loss verificando un éxito en torno a los parámetros

estudiados mostrando una mejor precisión y una pérdida aceptable.

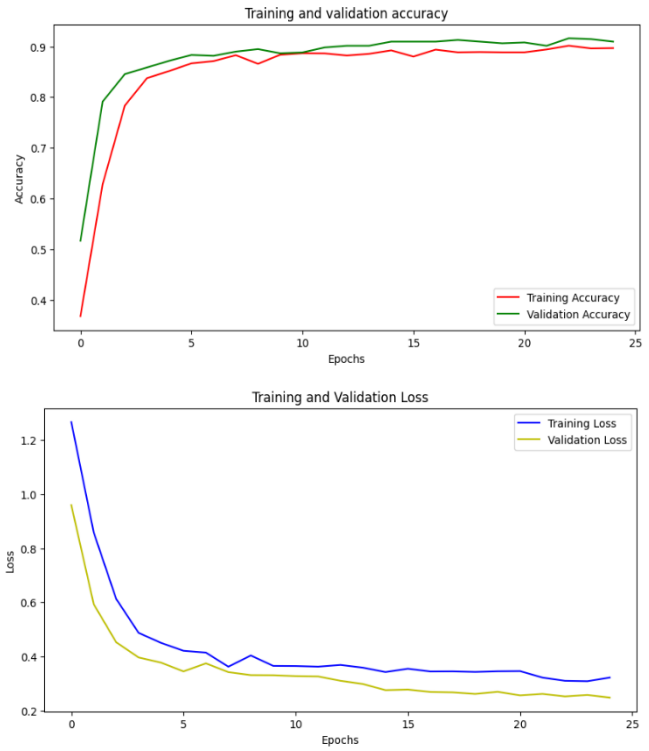

Fig. 16 Gráficos mejorados entrenamiento validación accuracy y validación loss

V. **CONCLUSIONES**

Las Redes Convolucionales (CNN) es una clase de Red Neuronal que tiene finalidad el procesamiento de datos, principalmente en el reconocimiento de imágenes y patrones que ayudan a aprender y extraer de manera automática las características relevantes que poseen las imágenes haciendo de este proceso una tarea fundamental en clasificar imágenes.

Por otro lado, las Redes Recurrentes (RNN) también pertenece a una clase de Red Neuronal, cuya función es mantener información a largo plazo, está diseñado para procesar información a través de secuencias de datos, donde la información es temporal. A diferencia de las CNN, las Redes Recurrentes presentan conexiones que forman bucles de retroalimentación que permiten recordar información anterior permitiendo que sea conservada durante un tiempo determinado.

Pero las Redes Recurrentes deben almacenar información a largo plazo debido al problema de la gradiente; para lograr abordar esta problemática se utilizan técnicas como las Redes LSTM que incorporan mecanismos para almacenar información a largo plazo.

En el código 1 que consiste en clasificar frutas en imágenes se utiliza la base de datos Frruit-360 en el cual se separan en 2 conjuntos: entrenamiento y prueba usando 6 clases, 3 clases de manzanas (*apple*) y 3 clases de peras (*pear*).

Al entrenar dicho modelo se puede establecer una alta medida en la medición de la exactitud (*accuracy*) y un menor grado de pérdida (*loss*) en el conjunto de entrenamiento. Pero en el conjunto de validación al aumentar de épocas se logra obtener una mayor exactitud y una menor pérdida ya que durante el reconocimiento de patrones los clasifica correctamente debido a que las muestras extraídas de las imágenes corresponden a su clase original; al evaluar el modelo se obtiene una exactitud de aproximadamente 98%.

En el código 2 que trata en clasificar noticias se utiliza la arquitectura bajo la secuencia *Keras* entrenando durante 25 épocas (*epochs*), en el cual se separan en grupos de entrenamiento (120000 muestras) y validación (7600 elementos) presentando 4 clases: internacional (*world*), deportes (*sports*), negocios (*business*) y ciencia/tecnología (*sci/tech*).

Antes de crear el modelo los datos son presentados a través de textos dejando en tokens numéricos a través de secuencias. Al entrenar el modelo se puede observar un rendimiento bastante bueno en cuanto a la precisión y una disminución en la pérdida, pero se pueden mejorar estas mediciones.

Se implementa una nueva arquitectura de Red Neuronal combinando CNN y RNN aplicando una

capa LSTM bidireccional logrando aumentar la precisión o exactitud y disminuír la tasa de error de la medición de los datos durante el entrenamiento.

BIBLIOGRAFÍA

[1] Sotaquirá, M. (2019). ¿Qué son las Redes Convolucionales?. Codificandobits. https://www.codificandobits.com/blog/redes-convolucionales-introduccion/

[2] Rapu, C. (2023). AyudantIA 5: Redes neuronales convolucionales (CNNs) e introducción a Tensor Flow. Universidad de O'Higgins. [PDF]

[3] Parada, P. (2022). Qué son las Redes Neuronales Convolucionales. IEBS. https://www.iebschool.com/blog/redes-neuronales-convolucionales-big-data/

[4] Sotaquirá, M. (2019). Introducción a las Redes Neuronales Recurrentes. Codificandobits. https://www.codificandobits.com/blog/introduccion-redes-neuronales-recurrentes/

[5] Gamco. (2021). Red Neuronal Recurrente Concepto y Definición. gamco.es . https://gamco.es/glosario/red-neuronal-recurrente