Proyecto semestral

Clasificador de frutas en Imágenes y Clasificador de Noticias

Nicolás Araya, Cristian Herrera, Cristóbal Lagos, Manuel Muñoz

COM4402-1 - Segundo Semestre 2023 – Introducción a la Inteligencia Artificial

Escuela de Ingeniería, Universidad de O'Higgins

22, 12, 2023

***Abstract*— Este trabajo se enfoca en el entrenamiento guiado de modelos de aprendizaje supervisado, específicamente en el uso de Redes Neuronales Convolucionales (CNN) para la clasificación de frutas. Se han seleccionado distintas clases de peras y manzanas del dataset Fruit-360. A su vez, también emplearemos Redes Neuronales Recurrentes (RNN) en conjunto con la CNN para clasificar diversos títulos de noticias extraídos del dataset AG's News de ComeToMyHead, que contiene más de 1 millón de artículos recopilados de más de 2000 fuentes.**

Se busca con este trabajo demostrar la efectividad de este enfoque, destacando la capacidad de escalabilidad para clasificaciones más amplias y no limitadas únicamente a un contexto acotado.

***Keywords*— CNN, RNN, Aprendizaje Supervisado, Redes Neuronales**

I. Introducción

La clasificación en deep learning se aplica a diversos contextos, como la clasificación de frutas y noticias. En el ámbito de las frutas, esta técnica permite identificar y categorizar automáticamente diferentes tipos de estas según características específicas, facilitando la gestión y selección en entornos como la producción agrícola y la cadena de suministro.

En el ámbito de la clasificación de noticias, la aplicación de deep learning resulta beneficiosa al automatizar la categorización de artículos o informes según su contenido, temática o relevancia. Este enfoque es valioso para organizar información en medios digitales, ofreciendo a los usuarios una experiencia más eficiente y personalizada al acceder a noticias específicas.

Para llevar en práctica la clasificación de estos dos tópicos importantes se utilizan dos datasets:

- El conjunto de datos Fruit-360 [1], contiene 82213 imágenes (100 × 100 píxeles) de frutas y verduras de 120 clases, ya subdivididas en conjuntos de entrenamiento y de prueba. Específicamente, se seleccionaron seis categorías del conjunto de datos (tres tipos de manzanas y peras): Apple Golden 1, Apple Pink Lady, Apple Red 1, Pear Red, Pear Williams y Pear Monster.
- AG's News [2] es un conjunto de datos proporcionado por ComeToMyHead de más de 1 millón de artículos de noticias recopilados de más de 2000 fuentes de noticias. Contiene las siguientes cuatro clases: World(0), Sports(1), Business(2), Sci/Tech(3).
  El número total de muestras de entrenamiento es de 120.000 y el de pruebas de 7.600. Cada clase contiene 30.000 muestras de entrenamiento y 1.900 muestras de prueba

Se espera que con el entrenamiento sea posible realizar la clasificación de frutas mediante una Red Neuronal Convolucional (CNN) y de texto a través de una Red Neuronal Recurrente (RNN) mejorada mediante el trabajo conjunto de una CNN que dota a la red de la capacidad de manejar secuencias y extraer palabras significativas en las frases de entrada.

II. Marco Teórico

1. Perceptrón Multicapa (MLP)

Es una red neuronal feedforward que consta de tres o más capas, incluyendo una capa de entrada, una o más capas ocultas y una capa de salida. Emplea funciones de activación no lineales.

2. Convolutional Neural Network

Es un tipo de red neuronal artificial diseñada específicamente para procesar datos de tipo malla, como imágenes. Utiliza capas convolucionales para detectar patrones locales en la entrada, compartiendo parámetros para mejorar la eficiencia y reducir la cantidad de datos necesarios.

- *Operación de convolución*
  Sea I la entrada de la imagen, K el kernel (filtro) y C la operación de convolución, la convolución de I y K se define como:
  $C(I,K) = \sum i,j\ I(i,j)\cdot K(i,j)$

  Esto implica a cada posición de la imagen I.

- *Capa de Pooling (MaxPooling)*
  La operación de Maxpooling en una pos $(i,\ j)$ con un tamaño de ventana $n\ x\ n$ se d como:
  $MaxPooling(I,n) = maxp,qI(p,q)$

- *Capa totalmente conectada*
  Sea x la salida de las capas anteriores, W y b los pesos y sesgos respectivamente, la operación en una capa totalmente conectada es:

  $$y = softmax(Wx + b)$$

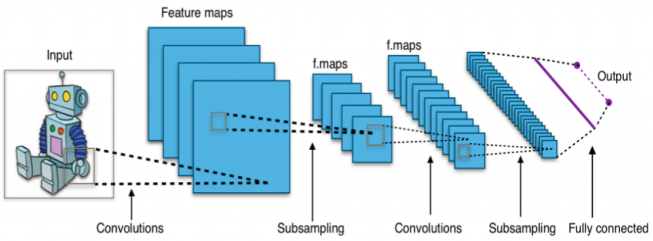

Fig. 1: Se muestra el modo de aprendizaje de una Convolutional Neural Network (CNN).

3. Recurrent Neural Network

Tipo de red neuronal que tiene conexiones de retroalimentación, permitiendo la persistencia de la información a lo largo del tiempo. Esto la hace adecuada para trabajar con secuencias de datos, como series temporales o texto.

- *Operación en una celda RNN*
  Dada una entrada x en un paso de tiempo t, la salida h_t de la celda RNN en el siguiente paso de tiempo se calcula como:

  $$h_t = tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$

  Donde W_{hh} y W_{xh} son matrices de pesos, h_{t-1} es la salida de la celda en el paso anterior, x_t es la entrada en el paso actual, b_h es el sesgo, y $tanh$ es la función tangente hiperbólica.

- *Salida de una celda RNN*
  La salida en el paso de tiempo t se calcula como una función de la salida de la celda h_t:

  $$y_t = softmax(W_{hy}h_t + b_y)$$

  Donde W_{hy} y b_y son pesos y sesgos respectivamente, y softmax es la función de activación softmax que convierte las salidas en una distribución de probabilidad.

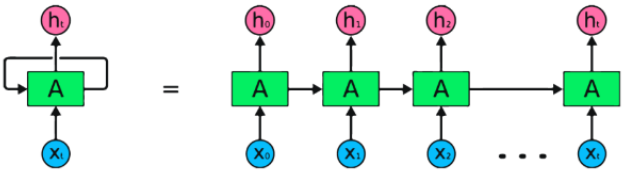

Fig. 2: Se muestra el modo de aprendizaje de una Recurrent Neural Network (RNN). Donde A corresponde al perceptrón, las distintas salidas de las cuales se retroalimenta.

4. Función de activación

La función ReLU, o Unidad Rectificadora Lineal,es una función común en las redes neuronales. Lo que hace es muy simple: si la entrada es positiva, deja la entrada sin cambios; si la entrada es negativa, la transforma en cero.

Esta función es popular porque es fácil de calcular y ayuda a resolver un problema en el entrenamiento de redes neuronales llamadas "desvanecimiento del gradiente". Además, su uso es común en muchas aplicaciones de aprendizaje profundo debido a su eficiencia y efectividad en la práctica.

La función sigmoide produce resultados que siempre están en el rango de 0 a 1. Debido a esta propiedad, normaliza la salida de cada neurona en una red neuronal. Se utiliza especialmente en modelos donde necesitamos predecir probabilidades como resultado. Esto se debe a que la probabilidad de cualquier evento siempre está en el rango de 0 a 1, por lo que la función sigmoide es una elección adecuada. Esta función también es conocida por su suave transición entre valores, evitando cambios bruscos en las salidas. Otra ventaja es que la función es diferenciable, lo que significa que podemos calcular la pendiente de la curva sigmoide en cualquier punto. Finalmente, esta función es efectiva para producir predicciones claras que tienden a acercarse a 1 o 0, lo que es útil en tareas de clasificación donde deseamos obtener resultados distintos y bien definidos.

5. Hiperparámetros

Épocas: Las épocas corresponden a un ciclo completo de entrenamiento a través de los datos de entrenamiento, lo que permite que la red ajuste sus pesos y mejore su capacidad para hacer predicciones precisas [4]. Este proceso comprende las siguientes etapas:

- **Forward Propagation**: Durante la propagación hacia adelante, los datos de entrenamiento se pasan a través de la red neuronal, capa por capa, desde la entrada hasta la salida. En esta etapa, se calculan las predicciones iniciales de la red en función de los pesos actuales.
- **Backward Propagation**: En esta etapa se calculan las derivadas parciales del error con respecto a los pesos de la red. Estas derivadas se utilizan para ajustar los pesos y mejorar el rendimiento del modelo.

Batch size: El hiperparámetro Batch Size se refiere al número de imágenes que se utilizan en la estimación de la gradiente durante el entrenamiento de un modelo. En cada iteración de entrenamiento, el modelo procesa un conjunto de datos de tamaño Batch Size antes de realizar una actualización de los pesos. Según Kandel [5], quien se centró en encontrar el valor óptimo de batch para entrenar clasificadores de imágenes, observó que existe una correlación significativa entre la tasa de aprendizaje y el tamaño del batch. En situaciones donde las tasas de aprendizaje son altas, un tamaño de batch grande tiende a desempeñarse mejor que con tasas de aprendizaje bajas. Por lo tanto, sugiere comenzar el entrenamiento de redes neuronales convolucionales (CNN) con un tamaño de batch más pequeño, resaltando la influencia de la relación entre la tasa de aprendizaje y el tamaño del batch en el rendimiento del modelo.

Para evaluar estos modelos existen distintas métricas, en particular estas nos sirven para cualquier tipo de aprendizaje supervisado. La métrica de accuracy, es la proporción de verdaderos positivos sobre la suma de falsos positivos y verdaderos negativos. Por ejemplo, en la detección de spam en correos electrónicos, si un correo legítimo se marca como spam por error, es un problema grave. Esto podría resultar en que los usuarios pierdan correos importantes. Así que, en tales casos, es crucial tener un alto nivel de accuracy para evitar estos errores costosos. La métrica accuracy es una medida que nos dice cuán seguido nuestro modelo acierta en sus predicciones. Para calcularla, simplemente dividimos la cantidad de predicciones correctas (todos los verdaderos positivos y verdaderos negativos) entre todas las predicciones realizadas. Es una forma de medir qué tan bien nuestro modelo está haciendo su trabajo. La métrica Recall evalúa cuántas veces un modelo de aprendizaje automático logra identificar correctamente los casos positivos (verdaderos positivos) de todos los casos positivos reales en el conjunto de datos. Se puede calcular dividiendo el número de verdaderos positivos entre el total de casos positivos. La matriz de confusión se trata de una métrica de evaluación de rendimiento utilizada en problemas de clasificación en el campo del aprendizaje automático, en los cuales la salida puede pertenecer a dos o más categorías.

III. Metodología

A) Clasificador de frutas en imágenes

El primer paso es importar todas las librerías necesarias para poder realizar el entrenamiento del modelo de aprendizaje supervisado CNN utilizado para esta tarea. Una vez importadas, se lee el conjunto de datos conservando las categorías de interés, las cuales son Apple Golden 1, Apple Pink Lady, Apple Red 1, Pear Red, Pear William y Pear Monster. Siendo las categorías anteriores variantes de las frutas pera y manzana utilizadas para la posterior clasificación.

Se separan los datos en conjunto de entrenamiento y de prueba (de un total de 4104 muestras, asignamos el 75% al conjunto de entrenamiento y un 25% al de prueba). Con el objetivo de tener una mayor accuracy a la hora de entrenar los modelos, se reordenan de forma aleatoria las muestras dentro de cada conjunto.

Se cambia la dimensión de los datos para que sea compatible con CNN, pues este modelo requiere de una dimensión extra, las imágenes correspondientes a los conjuntos de entrenamiento y prueba se cambian a una escala de grises y se realiza la normalización con el objetivo de ajustar las dimensiones. Además, de un ajuste a los hiperparámetros, entre los que destacan el número de épocas y la tasa de aprendizaje:

```
#Hiperparámetros
pool_size=(2, 2)
weight_decay = 5e-4
dropout = 0.6
lr = 0.001
momentum = 0.9
epochs = 10
batch_size = 32
```

Fig. 3: Hiperparámetros CNN.

El siguiente paso, antes de entrenar el modelo CNN, se especifica la arquitectura de la red neuronal profunda (ver Fig. 4). Este paso es importante, pues la red requiere de un diseño específico según el contexto en el que se llevará a cabo el entrenamiento.

Finalmente, debe entrenarse el modelo tal como se especifica anteriormente, realizando un total de 10 épocas.

B) Clasificador de noticias

Para comenzar, se importan las librerías a utilizar, siendo estas provenientes de tensorflow Keras y de ScikitLearn y cargando los datos. Inicialmente, el conjunto de entrenamiento contiene 120.000 muestras de noticias, mientras que el conjunto de prueba contiene 7600 muestras. Para armar el conjunto de validación se creó un conjunto a través del 10% de las noticias ubicadas en el conjunto de entrenamiento.

Una vez realizado esto, se prioriza la realización de un correcto preprocesamiento de los datos. Primero,

```
Model: "fruits-classifier"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 100, 100, 1)]     0

conv2d_3 (Conv2D)            (None, 100, 100, 16)      416

batch_normalization_3 (Bat   (None, 100, 100, 16)      64
chNormalization)

max_pooling2d_2 (MaxPoolin   (None, 50, 50, 16)        0
g2D)

conv2d_4 (Conv2D)            (None, 50, 50, 32)        8224

batch_normalization_4 (Bat   (None, 50, 50, 32)        128
chNormalization)

max_pooling2d_3 (MaxPoolin   (None, 25, 25, 32)        0
g2D)

conv2d_5 (Conv2D)            (None, 25, 25, 64)        18496

batch_normalization_5 (Bat   (None, 25, 25, 64)        256
chNormalization)

flatten_1 (Flatten)          (None, 40000)             0

dense_1 (Dense)              (None, 512)               20480512

dropout_1 (Dropout)          (None, 512)               0

id (Dense)                   (None, 6)                 3078

=================================================================
```

Fig 4: Estructura modelo clasificador de frutas.

Una vez realizado el paso anterior se procedió con la utilización de Keras con la función de TextVectorization layer que realiza una Tokenización, lo que significa que asigna un entero a cada palabra de la frase debido a que estos modelos no pueden trabajar con strings como tal. Después de crear la capa, se utiliza adapt para crear un vocabulario que será la lista de palabras individuales que componen una frase concreta a través del conjunto de datos.

```
[ ] sample_news = ['This weekend there is a sport match between Man U and Fc Barcelona',
                   'Tesla has unveiled its humanoid robot that appeared dancing during the show!']

    vectorized_news = text_vectorizer(sample_news)
    vectorized_news.numpy()

    array([[  40,  494,  186,   16,    3, 1567,  570,  159,  370,    1,    7,
            7486, 2556],
           [   1,   20,  876,   13,    1, 4845,   10, 1273,    1,  160,    2,
             532,    0]])
```

Fig 5: Vectorización de texto.

El siguiente paso corresponde a crear un modelo secuencial Keras que toma los textos de entrada y la salida de la clase de los textos de entrada (ver Fig. 6). El modelo va a estar compuesto por las siguientes capas:

1. **Capa de vectorización de texto:** Dedicada al preprocesamiento de los textos.
2. **Capa de embedding:** para representar los tokens en un vector de características entrenable en un espacio de alta dimensión.
3. **A Conv1D:** para procesar las secuencias.
4. **Capa densa:** para la clasificación.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 text_vectorization (TextVe  (None, None)              0
 ctorization)

 embedding (Embedding)       (None, None, 64)          640000

 conv1d (Conv1D)             (None, None, 64)          20544

 max_pooling1d (MaxPooling1  (None, None, 64)          0
 D)

 conv1d_1 (Conv1D)           (None, None, 64)          20544

 global_max_pooling1d (Glob  (None, 64)                0
 alMaxPooling1D)

 dense (Dense)               (None, 32)                2080

 dropout (Dropout)           (None, 32)                0

 dense_1 (Dense)             (None, 4)                 132

=================================================================
```

Fig 6: Estructura modelo secuencial.

Definida la estructura del modelo, se realiza el entrenamiento con un total de 25 épocas.

El paso siguiente corresponde a combinar las CNNs y LSTMs para ver si es posible mejorar los resultados. Así, Conv1D puede extraer palabras significativas en las frases de entrada, y las RNN pueden prever la secuencia contenida en las características de salida de las CNN. Sólo se añadirá una capa LSTM bidireccional para manejar las características de las CNNs desde ambas direcciones (ver Fig. 7).

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 text_vectorization (TextVe  (None, None)              0
 ctorization)

 embedding_1 (Embedding)     (None, None, 64)          640000

 conv1d_2 (Conv1D)           (None, None, 64)          20544

 max_pooling1d_1 (MaxPoolin  (None, None, 64)          0
 g1D)

 conv1d_3 (Conv1D)           (None, None, 64)          20544

 bidirectional (Bidirection  (None, 128)               66048
 al)

 dense_2 (Dense)             (None, 32)                4128

 dropout_1 (Dropout)         (None, 32)                0

 dense_3 (Dense)             (None, 4)                 132

=================================================================
```

Fig 7: Estructura modelo implementando LSTM.

IV. Resultados

A) Clasificados de frutas en imágenes

Una vez realizado el entrenamiento del modelo se obtienen las gráficas resultantes que muestran el rendimiento obtenido a través de la evolución del valor de la Accuracy tanto en el conjunto de entrenamiento como el de prueba a lo largo de las épocas para medir la curva de aprendizaje (ver Fig. 8) y la gráfica de la pérdida (ver Fig. 9).

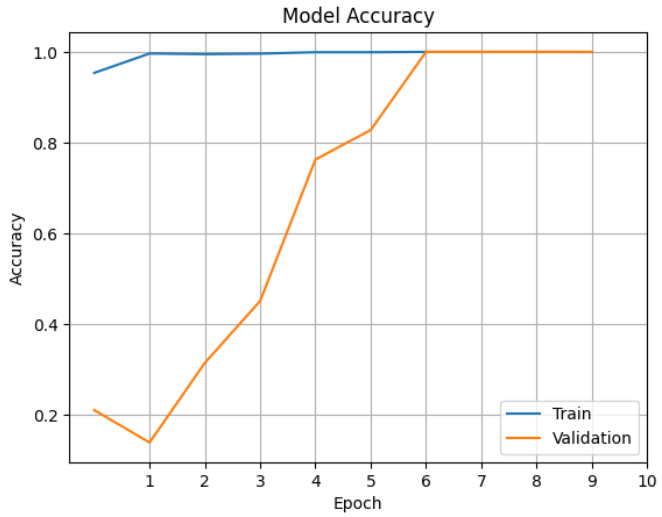

Fig 8: Accuracy del modelo.

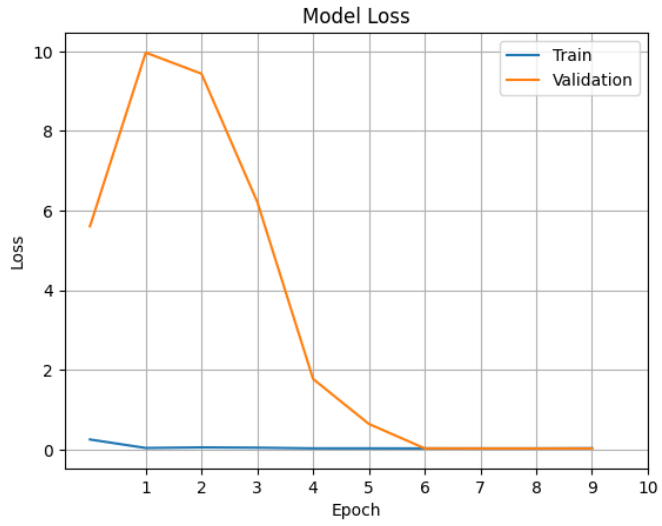

Fig 9: Loss del modelo.

Poniendo a prueba el modelo, se observan los mapas de activación de la tercera capa de convolución (ver Fig. 10):

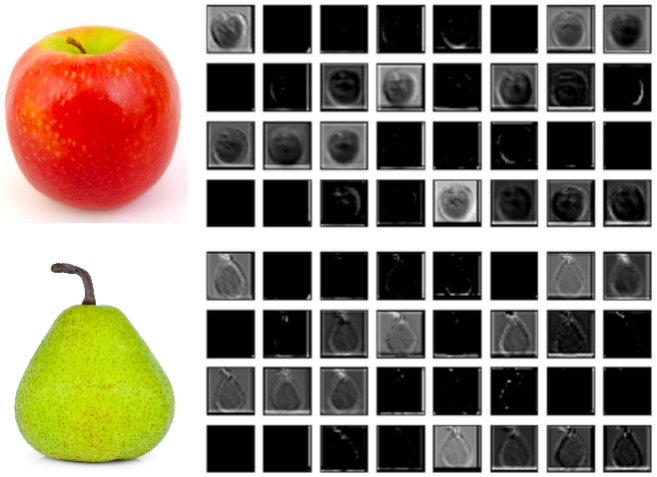

Fig 10: Mapa de activación tercera capa de convolución.

Por último, se realiza una medición de la matriz de confusión en el conjunto de prueba (ver Fig. 11):

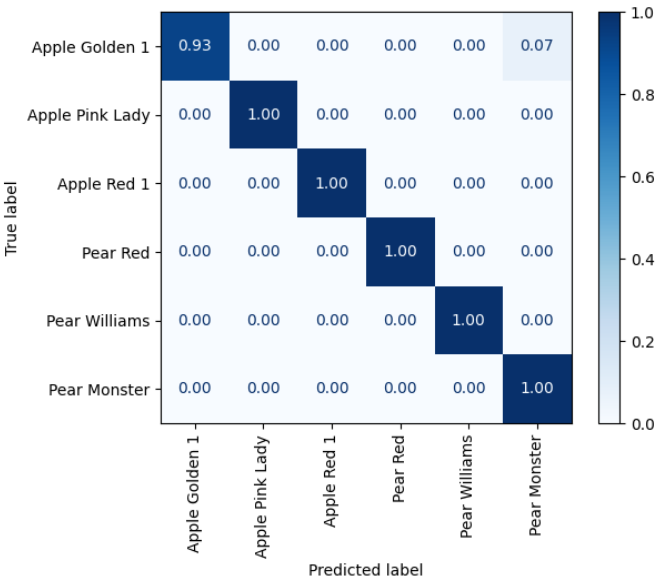

Fig 11: Matriz de confusión en el conjunto de prueba.

B) Clasificación de textos

Se crea un modelo secuencial Keras que toma los textos de entrada y la salida de la clase de los textos de entrada. Para analizar los errores, se traza la evolución del accuracy en el transcurso de las épocas (ver Fig. 12) y a su vez también se obtiene la gráfica de la pérdida (loss) (ver Fig. 13):

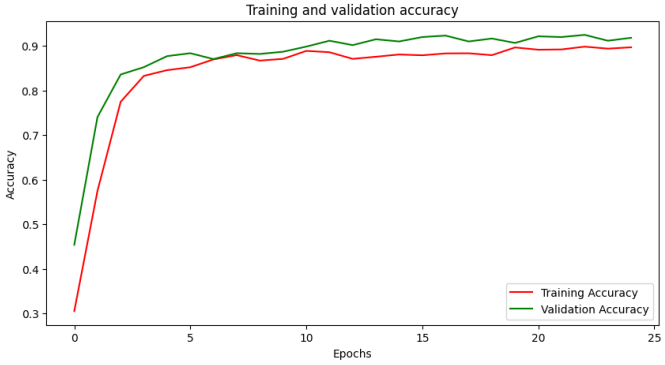

Fig 12: Accuracy vs épocas respecto al entrenamiento y validación

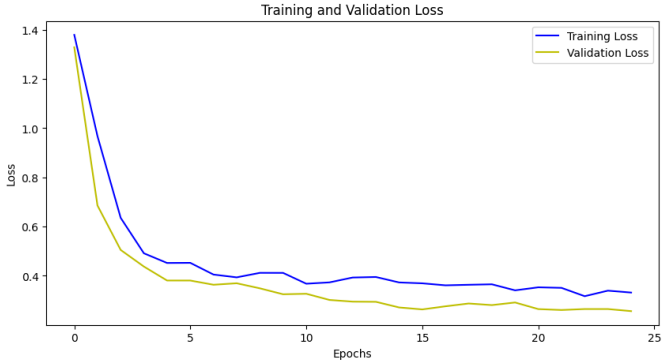

Fig 13: Loss vs épocas respecto al entrenamiento y validación.

Al usar CNN, se encuentra el inconveniente de que este modelo por sí solo no mantiene el orden de los pasos temporales y gracias a que RNN permite el reconocimiento de patrones a través de la retroalimentación en el aprendizaje (ver Fig. 2) .
Se añade una capa LSTM bidireccional para manejar las características de las CNN desde ambas direcciones (ver Fig.14 y 15).

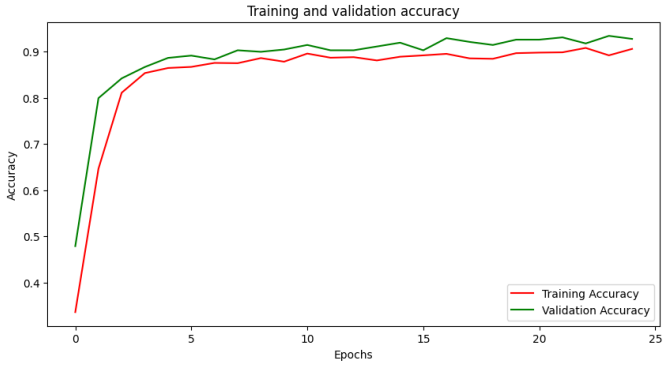

Fig 14: Accuracy vs épocas respecto al entrenamiento y validación

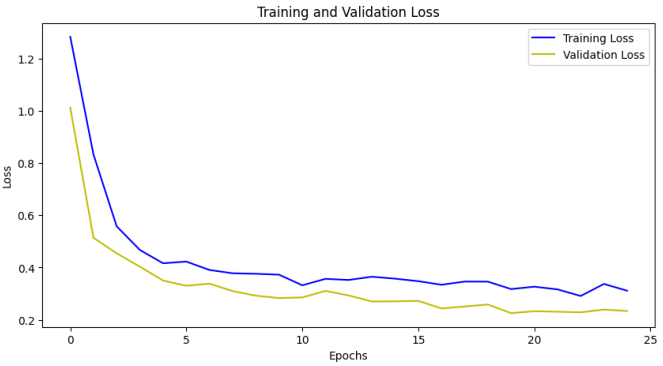

Fig 15: Loss vs Épocas respecto al entrenamiento y validación.

No hay mucha diferencia con el primero, pero los resultados que se obtienen combinando las Redes Neuronales Convolucionales (CNN) y las Redes Neuronales Recurrentes (RNN) tienen una superioridad notable a largo plazo en el entrenamiento del modelo..

Predicción con nuevos textos:

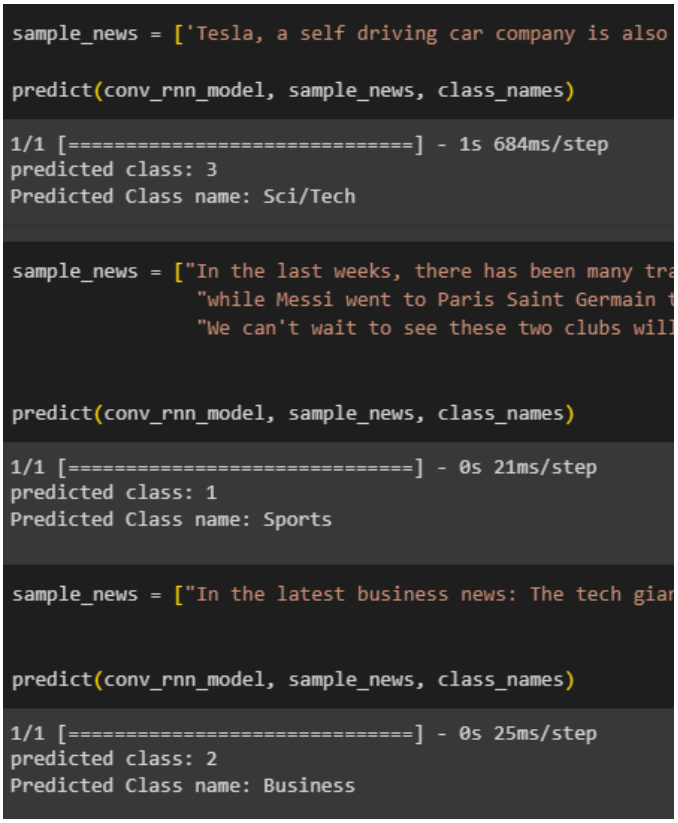

Fig 16: Inferencia de nuevos textos

V. Análisis

A) Clasificación de Frutas

La clasificación de frutas mediante el modelo obtiene un 90% de accuracy, lo cual significa que la clasificación es funcional. Esto se evidencia a través de la matriz de confusión mostrada en la figura 11. Solamente Apple Golden 1 (78%) y Pear Williams (67%) obtienen resultados más bajos, pero casi todas las demás tienen 100% de accuracy.

Al examinar las gráficas de Pérdida vs. Épocas, se aprecia que el modelo converge de manera consistente durante el proceso de entrenamiento en la figura 9 mostrando además, gracias a la métrica de pérdida en el conjunto de validación, que no existe overfitting como tal.

B) Clasificación de Noticias

Ahora analizando la clasificación de texto se aprecia en las gráficas que detallan el loss y el accuracy muestran que el entrenamiento realizado en el clasificador no contiene ningún signo de overfitting que esté afectando al accuracy, pues el loss se mantiene bajo y en un nivel adecuado. El entrenamiento resulta con un 90.46% de accuracy y 25.86% de loss para la última época del modelo CNN y los resultados obtenidos combinando ambos modelos son 91.61% de accuracy y 25.18% de loss. Como se puede ver, ambos modelos tienen un buen rendimiento. CNN con RNN es ligeramente superior, pero se tiene que considerar el mayor costo computacional debido al uso de capas RNN. Para despliegues en entornos con recursos limitados se debe tener en cuenta este factor importante, por lo tanto, si se trata de elegir el mejor modelo se puede determinar que solo CNN es superior. Obtiene valores de accuracy y loss prácticamente iguales con un menor costo computacional, ya que una RNN añade complejidad computacional debido a sus conexiones recurrentes y su procesamiento secuencial. La CNN, al no depender de conexiones recurrentes y ser más eficiente en términos de paralelización, logra un rendimiento comparable al de la CNN con RNN, pero no superior en entrenamientos más largos.

VI. Conclusiones Generales

A través del análisis se rescata que ambas arquitecturas de los modelos de aprendizaje supervisado logran el cometido, sin embargo, destaca la ligera superioridad de la colaboración de CNN con RNN que se evidenció en la clasificación de noticias.

Por el lado del clasificador de frutas, se demuestra que CNN es una herramienta bastante poderosa, pues el reconocimiento de las diversas variantes de peras y manzanas utilizadas como categorías demuestra la eficiencia en general para el reconocimiento de imágenes, abriendo paso a su implementación en nuevas formas de reconocimiento de patrones complejos en imágenes como en otras frutas u objetos de variantes como

los celulares y las marcas o detector de lenguaje de señas.

Por el lado del clasificador de texto, se demuestra que es posible realizar un trabajo conjunto con varios modelos a modo de “aprendizaje conjunto” que pueden ser entrenados para realizar tareas diferentes, pero con un objetivo en común.

Con lo anterior se agrega que puede ser utilizado no solamente en un problema en específico que tiene un modelo óptimo a utilizar, sino que también da la idea de incluir que 2 o más modelos convivan en armonía para dar la solución a un problema.

Aun así, es importante considerar que esto lleva consigo cierto trade-off, ya que aplicar más de 1 modelo repercute significativamente en un mayor costo computacional para procesar los datos, lo cual puede ser crítico en aplicaciones con grandes volúmenes de datos y en los que la eficiencia apremia en gran medida.

VII. Referencias

[1] Horea Muresan, Mihai Oltean, Fruit recognition from images using deep learning, Acta Univ. Sapientiae, Informatica Vol. 10, Issue 1, pp. 26-42, 2018.

[2] Xiang Zhang and Junbo Zhao and Yann LeCun, (2016) Character-level Convolutional Networks for Text Classification, arXiv:1509.01626 [cs.LG]

[3] I. Kandel and M. Castelli, The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset, ICT Express (2020), https://doi.org/10.1016/j.icte.2020.04.010.

[4] Araya, N., Bugueño, I. (2023) Tarea N2: Redes Neuronales https://github.com/NicolasAraya932/IAHOMEWORK/blob/main/Tarea2/InformeTarea2IANicolasAraya.pdf

Discusión:

Respecto al clasificador de las variantes de las peras y las manzanas se muestra con los resultados que el modelo creado es sólido gracias a el reconocimiento de patrones complejos en imágenes que permite identificar las diferencias de estas mismas.

Además, gracias a las gráficas se logra evidenciar que no existe sobreajuste en estos gracias a la métrica de pérdida evaluada con el conjunto de validación y una convergencia consistente.

Respecto al clasificador de noticias, se nota que la implementación de los dos modelos de red neuronal convolucional y recurrente son bastante útiles a modo de aprendizaje conjunto supervisado. Permitiendo que se clasifiquen los diversos tópicos ubicados en el conjunto de entrenamiento obtenido desde la base de datos de AG's News.
En la comparación realizada con el entrenamiento solo con CNN se descubre que, a pesar de que este modelo no mantiene el orden de los pasos temporales mostró buenos resultados en un entrenamiento de pocas épocas, ligeramente inferior al aprendizaje en conjunto. Esto puede evidenciar una posible mayor pérdida en entrenamientos de mayor cantidad de épocas en los que, seguramente el aprendizaje en conjunto será superior, ya que las redes neuronales recurrentes tendrán más tiempo para aprender las relaciones temporales.