UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

OCR CON REDES NEURONALES
CASO DE ESTUDIO: PROCESAMIENTO DE FACTURAS

Fredy Alejandro Velasco Castañeda

Matemáticas

Trabajo de fin de carrera presentado como requisito

para la obtención del título de

Matemático

Quito, 16 de febrero de 2024

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

HOJA DE CALIFICACIÓN

DE TRABAJO DE FIN DE CARRERA

USO DE OCR CON REDES NEURONALES PARA LA TRANSFORMACIÓN DIGITAL

Fredy Alejandro Velasco Castañeda

Carlos Jiménez, Ph.D. (c)

Antonio Di Teodoro, Ph.D.

.....

.....

Quito, 16 de febrero de 2024

© DERECHOS DE AUTOR

Por medio del presente documento certifico que he leído todas las Políticas y Manuales de la Universidad San Francisco de Quito USFQ, incluyendo la Política de Propiedad Intelectual USFQ, y estoy de acuerdo con su contenido, por lo que los derechos de propiedad intelectual del presente trabajo quedan sujetos a lo dispuesto en esas Políticas.

Asimismo, autorizo a la USFQ para que realice la digitalización y publicación de este trabajo en el repositorio virtual, de conformidad a lo dispuesto en la Ley Orgánica de Educación Superior del Ecuador.

Nombres y apellidos: Fredy Alejandro Velasco Castañeda

Código: 00126312

C.I.: 1719925917

Fecha: Quito, 16 de febrero de 2024

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

CAPÍTULO 1

INTRODUCCIÓN

La idea de incorporar funciones humanas a los sistemas digitales ha sido durante mucho tiempo un campo interesante y ampliamente investigado (Deepa y SS, 2014). Una de las funciones que se ha intentado incorporar es la capacidad de 'leer' e interpretar el texto impreso. En el siglo XIX, se llevaron a cabo varios trabajos de investigación en esta dirección, con el objetivo final de crear herramientas que faciliten la lectura para personas no videntes (Nagy, 2000). Sin embargo, posteriormente, con el creciente interés de la comunidad, los objetivos detrás de esta tarea se ampliarían.

Hoy en día, entre las aplicaciones más comunes se encuentran:

- **En el sector financiero**

  En esta área existe una gran demanda de extracción de información desde documentos impresos o digitales a texto que sea más fácil de manipular y almacenar. Esto se utiliza comúnmente en facturas y documentos legales.

- **Tráfico vehicular**

  Se utiliza para identificar las matrículas y señales de tránsito, principalmente para el registro y la detección de infracciones.

- **Identificación de identidades**

  En sistemas de seguridad, se usa para registrar y permitir o denegar el ingreso de individuos basados en su identificación.

  (Wang, Pan, Guo, Ji, y Deng, 2021)

En dirección a este objetivo, se han desarrollado algunos campos de investigación. En el presente trabajo, se ahonda sobre el campo llamado Optical Character Recognition (**OCR**) cuya

finalidad es extraer caracteres de un documento en formato de imagen que contenga escritura ya sea impresa o hecha a mano.

Si bien el interés y la investigación sobre el OCR no es algo nuevo y ya tiene décadas en proceso, sus capacidades se expandieron significativamente con el mayor entendimiento y facilidad de aplicación de las **redes neuronales** y el **machine learning**. Tanto es así que hoy en día se consideran un estándar para solucionar OCR, notablemente al entrenar los pesos en la suma ponderada (Smith, 2013).

Con el paso de los años, los OCR y las redes neuronales se han vuelto progresivamente más accesibles para la sociedad. En la década de 1950, estas tecnologías solo se encontraban en un puñado de empresas, lo que cambió en la década de 1980 con la llegada de los microprocesadores. Estos hicieron que los precios de estas tecnologías se redujeran en un factor de 10 (Nagy, 2000). En la actualidad, no solo la potencia de los dispositivos ha aumentado significativamente, sino también la capacidad de nuestra tecnología para conectarse a Internet. Esto, junto con la creciente popularidad del procesamiento de datos en la nube (Merenda, Porcaro, y Iero, 2020), permite que cualquier persona no solo tome una foto, sino también aplique procesos de OCR o redes neuronales a la misma.

En el presente trabajo, se profundiza en el OCR resuelto mediante métodos de aprendizaje automático desde una perspectiva matemática. Además, se incluye un estudio de caso en el que se integran las herramientas de programación disponibles en la actualidad para resolver estos planteamientos matemáticos. Se proporciona una introducción a los conceptos fundamentales, y se abre la puerta a las tecnologías en constante investigación en este campo de estudio

CAPÍTULO 2

MOTIVACIÓN

Si bien el uso de medios digitales para el almacenamiento de documentos ha venido en alza en las últimas décadas, el almacenamiento físico aún se mantiene como norma en varias instituciones. Este almacenamiento físico puede llegar a ocupar bodegas llenas de documentos, principalmente considerando que estos documentos pueden tener validez legal por varios años. Específicamente, en las entidades públicas del Ecuador, los documentos contables se guardan por siete años (Gobierno Nacional de la República del Ecuador, 2021), al igual que sus contrapartes digitales.

Es por esto que una práctica común es realizar la digitalización de estos documentos, prácticas que usualmente significan el escaneo de grandes bloques de documentos físicos. Sin embargo, si se requiere manipular o buscar con facilidad dentro de estos documentos, la digitalización no es suficiente, se necesita de caracteres manipulables. Esto nos lleva a la necesidad de una digitalización no solo por imágenes, sino también por caracteres. Realizar esta digitalización manualmente significaría un número considerable de horas y trabajadores, y en muchos casos sería inviable por tiempo o recursos. Esto hace que se genere la necesidad de una herramienta para facilitar esta tarea. Herramienta que será introducida y explorada en el presente trabajo.

CAPÍTULO 3

CONCEPTOS Y DEFINICIONES

3.1. OCR

El Reconocimiento Óptico de Caracteres, también llamado *OCR* por sus siglas en inglés, se refiere a un sistema computacional diseñado para la detección de caracteres en documentos digitales que contienen escritura a mano o impresa, con el fin de su conversión en documentos donde se puedan manipular los textos (Deepa y SS, 2014).

3.2. Redes Neuronales Artificiales

También llamadas solamente *Redes Neuronales*, son una técnica de Machine Learning que emula el funcionamiento de aprendizaje en los organismos biológicos, es decir, de las neuronas y dendritas (Aggarwal, 2018). Esta emulación es realizada matemáticamente con un modelo que toma un input de un vector $X = (X_1, X_2, ..., X_I)$ y construye una función no lineal $f(X)$ para predecir una respuesta Y (James y Witten, 2023), el vector X está compuesto de I muestras, con las cuales el modelo va a aproximarse lo mejor posible a una función f desconocida mediante nuestra respuesta Y (Lichtner-Bajjaoui, 2020).

Para realizar esta tarea, se usan 3 tipos de capas; la capa de inputs, que contiene la información que va a alimentar la red neuronal, luego las capas escondidas o hidden layer(s) que son una o varias capas que intentan encontrar el modelo que mejor se aproxime a mi función f desconocida y la última capa, el output layer que es el resultado de esta aproximación. Visualmente, este proceso tiene la siguiente representación:

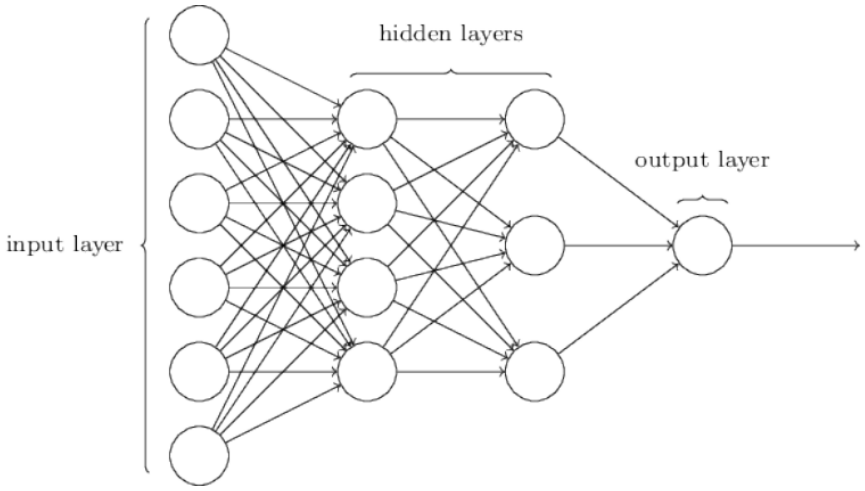

Figura 3.1: Representación de Red Neuronal Artificial

La Figura 3.1 muestra una imagen del libro(Nielsen, 2015).

Para nuestro input layer, sabemos que se trata de un vector con I entradas

$$X = (X_1, X_2, ..., X_I) \tag{3.1}$$

Que es tomado por cada uno de nuestros nodos en nuestro hidden layer para calcular una suma ponderada sobre estos, a cada nodo escondido, se lo llamará h, es decir, h es el índice del nodo en la hidden layer, al peso w y al total de la sumatoria se lo denomina a_h, o valor de activación de mi nodo h. Es decir tenemos la suma:

$$a_h = \sum_{i=1}^{I} w_{ih} x_i \tag{3.2}$$

Posteriormente, a esta sumatoria se le aplica una función localmente o globalmente diferenciable de activación θ_h tal que:

$$b_h = \theta_h (a_h) \tag{3.3}$$

Este proceso se repite para cada uno de los nodos escondidos h en el hidden layer, donde la función de activación depende del output deseado. A su vez, el output va a depender de cuál

es nuestro problema a resolver; si se está intentando resolver un problema de probabilidad, clasificación, regresión, etc. A este proceso se le llama también *Forward Pass*.

3.2.1. Función de Pérdida

Dado que la función objetivo es desconocida y las redes neuronales buscan hacer una estimación de la misma, existe un error asociado a las sumatorias realizadas. Este error, es calculado por la *función de pérdida* y denotado por $\mathcal{L}$, se calcula en función de los pesos de la red, los cuales se ajustan durante el entrenamiento mediante el gradiente.

En este trabajo, utilizaremos la función de pérdida conocida como *Cross-Entropy Loss*, también llamada *maximum likelihood*.

Para clasificaciones binarias, la función de pérdida es la siguiente:

$$\mathcal{L}(x, z) = -[z \cdot \log(y) + (1 - z) \cdot \log(1 - y)] \tag{3.4}$$

En el caso de redes neuronales con múltiples clases, la función de pérdida se expresa como:

$$\mathcal{L}(x, z) = -\sum_{k=1}^{K} z_k \cdot \log(y_k) \tag{3.5}$$

Donde z_k representa el k-ésimo elemento de la distribución real z, mientras que y_k es el k-ésimo elemento de la distribución predicha y.

Esta función de pérdida cuantifica la discrepancia entre las predicciones del modelo y las etiquetas reales, proporcionando una medida del error que la red neuronal busca minimizar durante el entrenamiento. La distribución real se toma de nuestros datos de entrenamiento o de test.

3.2.2. Backpropagation En Red Neuronales Con Múltiples Clases

Dado que los errores encontrados con nuestra función de pérdida pueden variar dependiendo de los parámetros que coloquemos, y nuestra intención es reducir el error tanto como sea posible, nos interesa encontrar los parámetros que minimizan esta función de pérdida.

Una técnica eficiente para calcular el gradiente y próximamente ir reduciendo el error es el *backpropagation* también conocida como el *backward pass* de la red.
Esta técnica es una repetición de la regla de la cadena en su versión para derivadas parciales. Se la realiza con el objetivo de saber cómo ir modificando mis parámetros, que inicialmente suelen ser aleatorios. Es decir, saber en que dirección tiene que ser mi modificación.

El proceso del backpropagation es el siguiente:

1. Derivada de la Pérdida respecto a la Salida de la Última Capa:

$$\frac{\partial \mathcal{L}(x,z)}{\partial a_k} = y_k - z_k \tag{3.6}$$

Esta derivada cuantifica cómo un cambio en la salida de la última capa afecta la función de pérdida.
Para simplificar, introducimos la notación:

$$\delta_h = \frac{\partial \mathcal{L}(x,z)}{\partial a_h} \tag{3.7}$$

Donde h es el índice de las unidades en la red.

2. Backpropagation del Error:

$$\frac{\partial \mathcal{L}(x,z)}{\partial z_i^{(l)}} = \sum_{j=1}^{J_{l+1}} \left(W_{ij}^{(l+1)} \cdot \frac{\partial \mathcal{L}(x,z)}{\partial z_j^{(l+1)}} \right) \cdot \theta'(z_i^{(l)}) \tag{3.8}$$

Aquí, calculamos cómo los errores en la capa siguiente afectan los errores en la capa actual, teniendo en cuenta los pesos (W) y la derivada de la función de activación θ'.

3. Error en los Nodos de la Capa Oculta (δ_h):

$$\delta_h = \theta'(a_h) \sum_{h' \in H_{l+1}} \delta_{h'} w_{hh'} \tag{3.9}$$

Aquí δ_h representa el error en el nodo h de la capa oculta, considerando la derivada de la función de activación y la suma ponderada de los errores en la capa siguiente.

4. Gradientes de los Pesos y Sesgos:

$$\frac{\partial \mathcal{L}}{\partial W_{ij}^{(l)}} = \delta_j \cdot a_j^{(l-1)} \tag{3.10}$$

$$\frac{\partial \mathcal{L}}{\partial b_i^{(l)}} = \delta_j \tag{3.11}$$

Aquí, calculamos las derivadas parciales de la función de pérdida respecto a los pesos (W) y sesgos (b) de la capa actual.

5. Actualización de Pesos y Sesgos:

Actualizamos los pesos y sesgos de la capa actual buscando minimizar el error. Dado que el proceso de backpropagation nos da la dirección de hacia donde tenemos que modificar nuestros parámetros, la magnitud de esta modificación se la atribuye a otro parámetro llamado *tasa de aprendizaje* y representado usualmente con α que es escogido al iniciar el proceso.

6. Repetición

Este proceso se repite en varias iteraciones, también llamadas *épocas*, hasta que se minimice el error tanto como sea posible. La convergencia se alcanza cuando los parámetros convergen a valores que reducen significativamente el resultado de la función de pérdida o después de un número predeterminado de épocas.

3.3. RNN

Recurrent Neural Networks, también llamadas RNN por sus siglas, son un subgrupo de redes neuronales usadas para procesar información secuencial (Goodfellow, Bengio, y Courville, 2016) lo que quiere decir que el output de la anterior iteración puede ser un input de la actual.

Dependiendo del número de inputs y outputs, existen cuatro tipos de RNN:

1. **Uno a uno**

   Es la más parecida a una red neuronal clásica, pues solo existe un input y un output.

2. **Uno a varios**

   Se trata de un solo input que genera varios outputs. El input y los outputs no necesariamente son del mismo tipo. Como ejemplo tendríamos una imagen que genera varios textos o un texto que genere varias imágenes.

3. **Varios a uno**

   En este caso tenemos varios inputs y un solo output. Ejemplos de este tipo de RNN serían problemas de clasificación con una clase o problemas con respuesta binaria.

4. **Varios a varios**

   Esta RNN posee varios inputs y varios outputs. No necesariamente poseen el mismo número de inputs que de outputs. Ejemplos de esto serían un corrector de errores ortográficos o un traductor.

Al igual que las redes neuronales clásicas, las RNN consisten de tres capas, el input, hidden layer(s) y el output. En cuanto al input, se mantiene con la misma forma, es decir el vector de entrada ya conocido (3.1). Las diferencias comienzan a aparecer en la hidden layer. Tomemos un input de tamaño T con I inputs y H nodos ocultos. Es decir, x_i^t es el valor del input i en el tiempo t. La sumatoria (3.2) es modificada a:

$$a_h^t = \sum_{i=1}^{I} w_{ih} x_i^t + \sum_{h'=1}^{H} w_{h'h} b_{h'}^{t-1} \tag{3.12}$$

Donde vemos la presencia de t, ya que para las RNN sí es relevante el tiempo. Además una segunda sumatoria que no existía en (3.2), esto se debe a que se necesita que las activaciones sucedan en el input actual y también en la activación del $t-1$, es decir, del paso anterior (Graves, 2012). Cabe recalcar que los pesos (w) son constantes en cada paso de tiempo en las RNN.

Las funciones de activación, al igual que en las redes neuronales tienen que ser diferenciables y son similares a (3.3) con la diferencia que sí se toma en cuenta el tiempo:

$$b_h^t = \theta_h \left(a_h^t \right) \tag{3.13}$$

Las RNN poseen varias unidades de funciones de activación. Cada unidad tiene una función de activación fija para cada paso $t \in T$ y también tiene un *paso escondido* que contiene la información pasada, es decir, la resultante desde $t = 0$ hasta el t actual. Este paso escondido se actualiza a cada instancia de t, y se realiza de forma recurrente, es decir, en base a $t - 1$. A este proceso (al igual que en las redes neuronales clásicas) también se le llama forward pass. El backward pass que se usa en las RNN es el BPTT.

Ejemplos del tipo de datos a los que se les aplica RNN son secuencias de texto, tiempo, etc. Las RNN son particularmente atractivas porque usan funciones iterativas para guardar información, teniendo la capacidad de aprender qué guardar y qué ignorar.

Si bien las RNN tienen las ventajas antes mencionadas sobre las Redes Neuronales clásicas, su problema es que no tienen la capacidad de retener información por largos periodos de tiempo (Graves, 2012).

3.4. BPTT

Asi como en las redes neuronales clásicas, en las RNN también existe el backward pass, con la diferencia de que no es llamado backpropagation, sino Backpropagation Through Time o BPTT por su siglas. Así como en las RN, se trata de un uso repetido de la regla de la cadena en su versión de derivadas parciales.

Similar a (3.7) introducimos la notación:

$$\delta_j^t = \frac{\partial \mathcal{L}}{\partial a_j^t} \tag{3.14}$$

Que representa el error de cada unidad j en la red neuronal en un paso de tiempo $t \in T$.

3.4.1. Error en los Nodos de la Capa Oculta

Tomando nuestro mencionado t, ahora añadimos que:

$$\delta_j^{T+1} = 0, \forall j \tag{3.15}$$

Donde T es el ultimo de mis pasos de tiempo.

Ahora tenemos:

$$\delta_h^t = \theta'\left(a_h^t\right)\left(\sum_{k=1}^{K}\delta_k^t w_{hk} + \sum_{h'=1}^{H}\delta_{h'}^{t+1} w_{hh'}\right) \tag{3.16}$$

Muy similar a (3.9) con la diferencia de que ahora el error no solo depende de la capa de salida, pero también de la capa escondida en el siguiente paso de tiempo. Esto sucede tomando en cuenta que la iteración comienza con $t = T$ y va decreciendo un paso de tiempo cada vez. Esta forma de iterar hace que el BPTT obtenga su nombre de *backwards* o *"hacia atrás"*.

3.4.2. Gradientes de Los Pesos y Sesgos

Tomando en cuenta el hecho de que en las RNN los pesos se mantienen durante cada paso de tiempo, los gradientes con respecto a los pesos son:

$$\frac{\partial \mathcal{L}}{\partial w_{ij}} = \sum_{t=1}^{T}\delta_j^t b_i^t \tag{3.17}$$

3.4.3. Actualización de Parámetros

Al igual que en el backpropagation clásico, los parámetros son ajustados buscando minimizar el error, tomando en cuenta que esos pesos se mantendrán constantes en cada paso de tiempo.

3.4.4. Repetición

La cantidad T de veces que repetimos el BPTT depende de la decisión del investigador, pero usualmente tiene que ver con el poder computacional disponible y de cuánto se desea que la RNN tenga acceso a los pasos más antiguos en la cadena. Por ejemplo, si se tiene una cadena de caracteres dentro de una historia, la magnitud de T va a dictar qué tanto la red va a tener presente el inicio de la historia al momento final de la misma.

3.5. El Problema Con los Gradientes

Recordemos que los procesos de Backpropagation y BPTT son usos repetidos de la regla de la cadena para derivadas parciales. Como visto anteriormente, esto se realiza usando el gradiente de la función de perdida. Si retomamos la ecuación de errores en los nodos de la capa oculta en nuestro BPTT:

$$\delta_h^t = \theta'\left(a_h^t\right)\left(\sum_{k=1}^{K}\delta_k^t w_{hk} + \sum_{h'=1}^{H}\delta_{h'}^{t+1} w_{hh'}\right) \tag{3.18}$$

Y deshacemos la simplificación previamente realizada de:

$$\delta_j^t = \frac{\partial \mathcal{L}}{\partial a_j^t} \tag{3.19}$$

Obtenemos la ecuación:

$$\frac{\partial \mathcal{L}}{\partial a_h^t} = \theta'\left(a_h^t\right)\left(\sum_{k=1}^{K}\frac{\partial \mathcal{L}}{\partial a_k^t} w_{hk} + \sum_{h'=1}^{H}\frac{\partial \mathcal{L}}{\partial a_{h'}^{t+1}} w_{hh'}\right) \tag{3.20}$$

Esta ecuación se compone de la suma de productos entre los pesos w y las derivadas parciales de la función de pérdida $\mathcal{L}$ respecto a los valores de activación a, multiplicadas posteriormente por la derivada de la función de activación θ'.

El resultado de esta suma depende significativamente de la magnitud de los valores presentes

en el gradiente y en los pesos. Dado que este proceso involucra múltiples pasos de tiempo, la multiplicación se repite y su resultado depende de los valores obtenidos en pasos de tiempo previos (Aggarwal, 2018). Esto es fundamental para comprender el problema del gradiente que puede explotar o desvanecer en redes neuronales recurrentes.

En este contexto van a haber tres casos posibles:

1. **El gradiente se mantiene estable:**

   Si nuestras multiplicaciones se mantienen muy cercanas a 1, la multiplicación sera estable.

2. **El gradiente desvanece:**

   Supongamos que el valor de la multiplicación entre el gradiente y el peso es $0 < p < 1$, en una red neuronal con r capas. Esto haría que al moverse por estas r capas, el valor de mi gradiente va acercándose a p^r, y que vaya haciéndose significativamente más pequeño a medida que avanza en las capas y pasos de tiempo. El gradiente entonces ajusta las primeras capas de manera significativa, pero apenas ajusta las últimas capas. A este problema se lo llama *desvanecimiento de gradiente* o *vanishing gradient*.

3. **El gradiente explota**:

   Cuando el valor p de la multiplicación entre el gradiente y el peso es mayor que 1, se produce un problema opuesto conocido como *explosión del gradiente* o *exploding gradient*. En este caso, los gradientes crecen exponencialmente a medida que avanzan en las capas y pasos de tiempo, pues p^r en esta ocasión es de una magnitud muy grande. Este problema puede causar divergencia en el proceso de entrenamiento y resultados incoherentes.

El caso 2 y 3 son más comunes de lo que se desearía y presentan un problema para las RNN principalmente. Estos problemas pueden surgir cuando se propaga el error a través de múltiples capas de una red neuronal, y pueden afectar significativamente la capacidad de la red para aprender y converger.

El problema nos lleva a un mal ajuste de nuestra red neuronal, y a que el uso de más capas, en vez de ayudar a la red, no tenga efecto alguno o incluso efecto negativo en su convergencia (Nielsen, 2015).

La elección adecuada de la función de activación y la implementación de técnicas que mitigan el gradiente desvaneciente (como ReLU o variantes) son estrategias clave para evitar esto.

3.6. LSTM

Long Short-Term Memory, también conocido como LSTM por sus siglas, es un rediseño de la arquitectura de las RNN enfocado en las celdas de memoria con el fin de mitigar el problema del gradiente desvaneciente. Este algoritmo fue introducido en 1997 por Hochreiter y Schimdhuber (Nielsen, 2015).

En estructura, la red neuronal es igual a una RNN como las que se presentó anteriormente, con la diferencia de que las unidades de suma Σ en la capa oculta (h) son reemplazadas por celdas de memoria (*memory cells*) (Graves, 2012).

Ademas de la arquitectura ya vista en las RNN, las LSTM tienen los siguientes elementos en sus bloques de memoria que controlan el flujo de la información:

1. **Puerta de entrada:**

$$i_t = \sigma\left(\sum_{i=1}^{I} W_{xi} x_i^t + \sum_{h'=1}^{H} W_{hi} h_{h'}^{t-1} + W_{ci} c_{t-1} + b_i\right) \tag{3.21}$$

Donde:

- x_i^t es la entrada i en el paso de tiempo t.
- $h_{h'}^{t-1}$ es el estado oculto en $t-1$ y la unidad h'.
- c_{t-1} es la celda de memoria en $t-1$.

La función de activación σ determina si la puerta se encuentra 'abierta' o 'cerrada', es decir, si está más cerca del 1 o del 0 respectivamente. Un ejemplo de función que resulta

efectiva para llevar a cabo esta tarea es la función sigmoide (σ). Cuando la puerta está cerrada (cerca de 0), significa que la celda de memoria no se actualizará significativamente con la información actual, permitiendo que la información previamente almacenada se mantenga durante más tiempo sin ser sobrescrita y que pueda ser accesada en etapas más avanzadas de la red (Graves, 2012).

2. **Puerta de olvido:**

$$f_t = \sigma \left(\sum_{i=1}^{I} W_{xf} x_i^t + \sum_{h'=1}^{H} W_{hf} h_{h'}^{t-1} + W_{cf} c_{t-1} + b_f \right) \tag{3.22}$$

La puerta de olvido es muy similar a la puerta de entrada, con la característica especial de que está conectada al mismo bloque de memoria. Su función principal es controlar cuánta información de la celda de memoria pasada c_{t-1} debe ser mantenida y cuánta debe ser olvidada en función de la entrada actual x_t y la salida anterior h_{t-1}.

La puerta de olvido desempeña un papel crucial en el funcionamiento de una celda LSTM al determinar qué información anterior debe mantenerse en la memoria a largo plazo y cuál debe ser descartada en función de las necesidades del problema que se quierer resolver.

3. **Puerta de salida:**

$$o_t = \sigma \left(\sum_{i=1}^{I} W_{xo} x_i^t + \sum_{h'=1}^{H} W_{ho} h_{h'}^{t-1} + W_{co} c_t + b_o \right) \tag{3.23}$$

La puerta de salida controla cuanta información va a pasarse a la siguiente etapa de la red.

4. **Cálculo de celda**

$$g_t = \tanh \left(\sum_{i=1}^{I} W_{xg} x_i^t + \sum_{h'=1}^{H} W_{hg} h_{h'}^{t-1} + b_g \right) \tag{3.24}$$

Su función es determinar si la celda de memoria c_t debe ser actualizada con los resultados obtenidos en el paso de tiempo actual y las salidas anteriores.

5. **Celda de memoria**

$$c_t = f_t \cdot c_{t-1} + i_t \cdot g_t \tag{3.25}$$

Donde:

- c_t representa la celda de memoria en el paso de tiempo t.
- f_t corresponde a la salida de la puerta de olvido en el paso de tiempo t.
- i_t representa la salida de la puerta de entrada en el paso de tiempo t.
- g_t es la salida de la función de cálculo de celda en el paso de tiempo t.

Cada bloque contiene al menos una celda de memoria conectada con el mismo bloque. La función de las celdas de memoria es almacenar y acceder a información en largos periodos de tiempo dentro de la red.

Es importante mantener la distinción; en este trabajo, a todo este sistema se le llama bloque de memoria, y dentro del mismo se tienen celdas de memoria, que no son lo mismo.

A continuación se presenta una representación gráfica de un bloque de memoria tomado de (Naseer y Zafar, 2018):

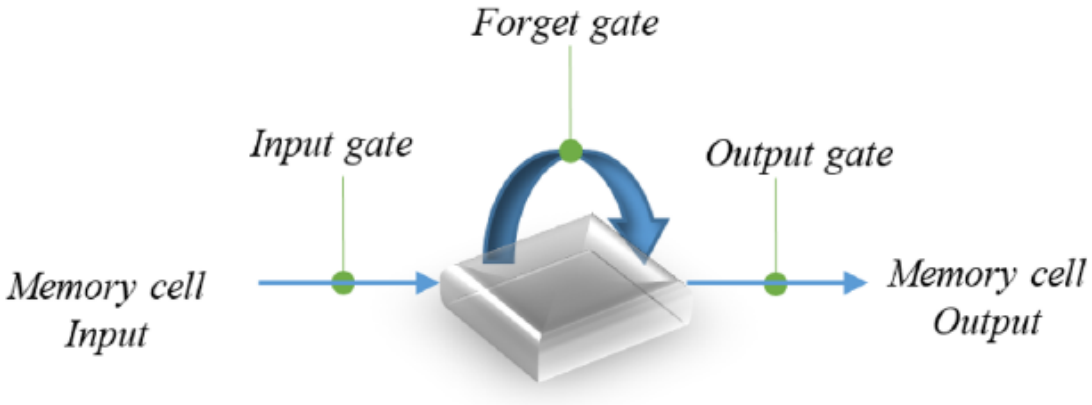

Figura 3.2: Bloque de memoria

Si bien las LSTM toman un largo tiempo para ser entrenados (James y Witten, 2023) se presentan como una posible solución a la falta de almacenamiento de las RNN, ya que se

pueden acceder a información en largos periodos de tiempo. Esto es posible gracias a que pueden distinguir qué información es relevante y debe ser mantenida, y qué información debe ser olvidada (Goodfellow y cols., 2016).

CAPÍTULO 4

CASO DE ESTUDIO

4.1. Caso de Estudio

Se desea realizar un caso de estudio para el reconocimiento de caracteres en facturas electrónicas emitidas en Ecuador para facilitar su manipulación futura. Para esto se usará OCR. Se debe escoger el modelo y programa específico para la tarea.

4.2. Facturas

Las facturas electrónicas serán tomadas de compras diversas realizadas en establecimientos ecuatorianos de distinta índole; supermercados, cines, etc. Es importante extraer de ellas principalmente el valor de los productos, el detalle y el numero de la factura en formato de texto. Las facturas vendrán inicialmente en formato PDF, pero no como texto editable, sino como una imagen.

4.3. Selección Del Modelo

Asma Naseer realizó un estudio en el 2018 (Naseer y Zafar, 2018) comparando la conjunción de LSTM y RNN contra un modelo con redes neuronales convolucionales (CNN) y un modelo correlacional, todos resolviendo el mismo problema de OCR. La comparación se realizó midiendo la precisión promedio del modelo, es decir, sus predicciones vs el grupo de datos de prueba. En el estudio, las precisiones de los modelos quedaron colocadas en el siguiente orden: LSTM-RNN (98.38 %), CNN (94.30 %), Modelo correlacional (90.95 %). Basado en esta información, el modelo que se escoge es LSTM-RNN.

4.4. Tipo de RNN

Una vez decidido que vamos a utilizar una RNN para llevar a cabo la tarea y que el input que vamos a recibir es de tipo PDF, debemos identificar qué tipo de RNN vamos a utilizar.

A pesar de que el input proviene de un archivo PDF, para nuestro programa será considerado como un único bloque de información. Por lo tanto, nuestra red será de tipo 'uno a varios', ya que ingresará un solo archivo y producirá como resultado una serie de palabras y números en forma de caracteres. Específicamente, se trata de una RNN 'uno a varios' con output de diferente tipo al input.

En este caso, utilizaremos el algoritmo de entrenamiento Backpropagation Through Time (BPTT), y el problema que estamos abordando es una tarea de clasificación con múltiples clases.

4.5. Función de Activación

Una vez que hemos identificado el tipo de problema que estamos tratando de resolver, es crucial seleccionar una función de activación que se adapte a nuestras necesidades. En este caso, buscamos una red neuronal recurrente capaz de transformar un PDF en varios caracteres con su información relevante. Dado que estamos abordando un problema de clasificación con múltiples clases, existen diversas funciones de activación que podemos considerar.

A continuación, se presentan algunas opciones que se ha tenido en cuenta para este trabajo debido a que funcionan en este contexto y a su amplia investigación en el campo:

4.5.1. Sigmoide

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{4.1}$$

En inglés, es conocida como *sigmoid function* y se utiliza con frecuencia en las unidades de puerta de las RNN, como es el caso de las LSTM, con el fin de controlar la activación de las celdas de memoria.

- Gráfica:

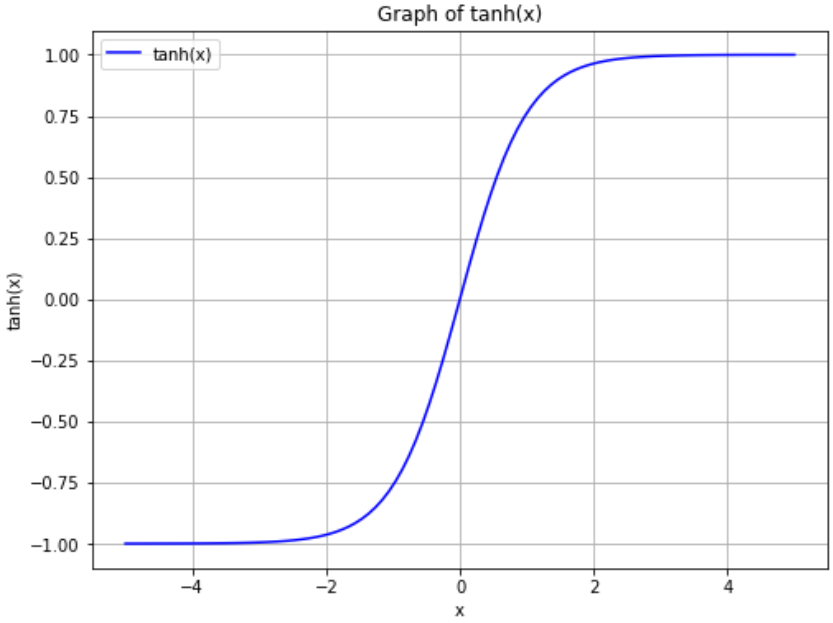

Figura 4.1: Función sigmoide

- Dominio:

$$-\infty < x < +\infty$$

- Rango:

$$0 \leq \sigma(x) \leq 1$$

- Gráfica de la derivada:

Figura 4.2: Derivada función sigmoide

Aquí se puede apreciar que el gradiente nunca sobrepasa los 0.25.

La salida o rango de valores de esta función estará comprendido entre 0 y 1. Esto la convierte en una opción muy utilizada cuando se requiere una salida que represente una probabilidad. A diferencia de la función de escalón ($stepfunction$), que es similar al sigmoide pero tiene un rango estricto de 0 o 1, la función sigmoide es suave. Sin embargo, el hecho de que su gradiente nunca sobrepase los 0.25 hace que esta función sea susceptible al problema del gradiente que se desvanece (Aggarwal, 2018).

4.5.2. Tangente Hiperbólica

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{4.2}$$

También conocida simplemente como '$tanh$', se utiliza en RNN y LSTM debido a su capacidad para controlar la activación de las celdas de memoria. Es similar a la función sigmoide, pero es importante destacar que tanto su función original como su derivada tienen rangos distintos a la función anterior.

- Gráfico:

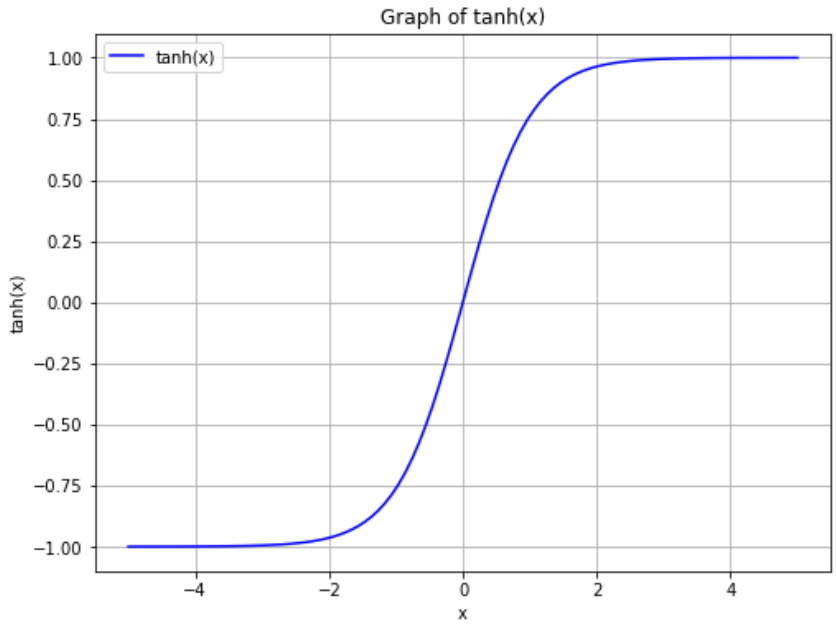

Figura 4.3: Función tanh

- Dominio:

$$-\infty < x < +\infty$$

- Rango:

$$-1 \leq \tanh(x) \leq 1$$

- Gráfico de la derivada:

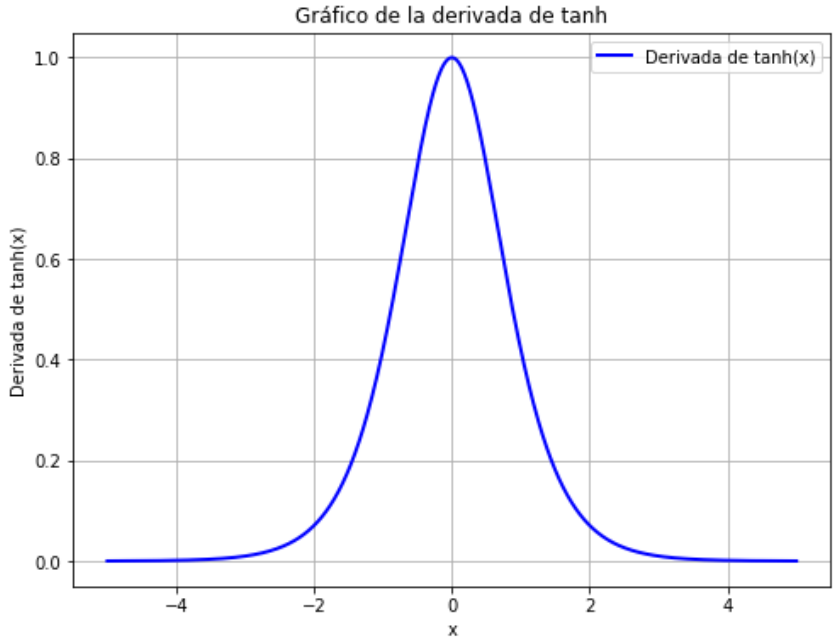

Figura 4.4: Derivada tanh

Como se mencionó anteriormente, no solo el rango de la función original difiere de la función sigmoide, sino que también el rango de su derivada es diferente. En este caso, la derivada de la función $tanh$ se encuentra en el rango de 0 a 1.

4.5.3. Relación Entre la Función Sigmoide y Tanh

Las dos funciones anteriormente mencionadas no solo son usadas en casos similares, sino también poseen una relación mediante la transformación lineal:

$$\tanh(x) = 2\sigma(2x) - 1 \tag{4.3}$$

Esto significa que cualquier red que utilice una capa oculta con una de estas funciones de activación puede cambiar a la otra mediante esta transformación. La razón por la cual se presentan como dos funciones diferentes es que tienen salidas distintas debido a sus rangos, por lo que la elección entre una u otra depende del problema a resolver (véase (Graves, 2012)).

4.5.4. ReLU

$$\text{ReLU}(x) = \text{máx}(0, x) \tag{4.4}$$

La Rectified Linear Unit, mejor conocida como ReLU, es una función que ha ganado popularidad en tiempos recientes. Esto se debe a que se calcula rápidamente, introduce no linealidad en la red y reduce la probabilidad de que ocurra el problema del gradiente que desvanece (Aggarwal, 2018).

- Gráfico:

Figura 4.5: Función ReLU

- Dominio:

$$-\infty < x < +\infty$$

- Rango:

$$0 \leq \text{ReLU}(x) \leq +\infty$$

- Grafico de la derivada:

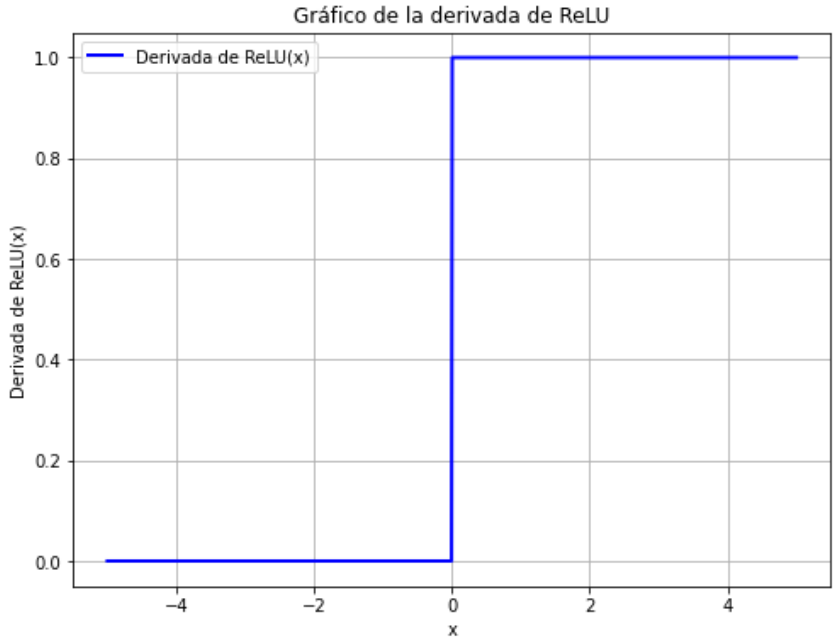

Figura 4.6: Derivada de ReLU

Como se puede observar, la derivada de cualquier argumento positivo es igual a 1. Esto la hace más resistente a los problemas de gradiente que desvanece o explota. Además, esta función se entrena más rápidamente debido a la simplicidad en el cálculo del gradiente. Es importante destacar que, aunque esta función no es suave en $x = 0$, es una función suave por partes. Este tipo de funciones han ganado popularidad en los años anteriores a este trabajo de investigación (Aggarwal, 2018).

4.5.5. Softmax

$$\text{Softmax}(x)i = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}} \tag{4.5}$$

La función softmax, también conocida como modelo logit multinomial, se utiliza frecuentemente en problemas que implican la clasificación en más de dos clases. Una característica fundamental de esta función es que garantiza que la suma de todas las activaciones sea igual a 1. Esta propiedad hace que la función sea especialmente útil y se asocie comúnmente con problemas que requieren salidas expresadas en forma de probabilidades (Nielsen, 2015).

- Gráfico:

  En el caso de softmax, dado que vamos a tomar clases mayores a dos, nuestro output no es graficable en dos dimensiones. A lo sumo podría hacerse una distribución de probabilidad que representen los outputs, pero no sería mas que un ejemplo, por lo que no se realizará.

- Dominio:

$$-\infty < x_i < +\infty$$

- Rango:

$$0 \leq \text{Softmax}(x)_i \leq 1$$

- Rango de la suma de las activaciones:

$$\sum_{i}^{K} \text{Softmax}(x)_i = 1$$

En el contexto del OCR, la función Softmax puede ser utilizada para clasificar y determinar a qué carácter en específico se refiere el que se está analizando, en función de una distribución de probabilidad. Se seleccionará aquel carácter que tenga la mayor probabilidad dentro de nuestro output de activaciones.

4.6. Herramientas de Software

Para la herramienta de OCR, es importante que el algoritmo que se use sea LSTM-RNN. Ademas, se decidió que sea open source y que sea compatible con Python. El paquete de Tesseract, desarrollado originalmente por HP a partir de 1985 y posteriormente liberada como

open-source en 2006(Smith, 2013) cumple con estas condiciones (J, 2021), además de ser un paquete popular en la actualidad, por lo que tiene basta información en la red. La librería de Python que contiene los paquetes de Tesseract es pytesseract (Ooms, 2023), que a la fecha tiene soporte de Google. Este wrapper además tiene la capacidad de reconocer caracteres alfanuméricos y en distintos idiomas, para posibles futuras expansiones al presente trabajo. Además de ser una herramienta ampliamente utilizada, existen autores, como (Singh y Bhushan, 2019), que afirman que Tesseract es la herramienta más precisa de OCR.

4.6.1. Funcionamiento Tesseract

Se sabe gracias a (Smith, 2013) en su texto 'An Overview of the Tesseract OCR Engine' bajo la aprobación y publicación de Google Inc. que los pasos que realiza Tesseract son los siguientes:

1. **Análisis de Componentes Conectados**

   Se trata de un análisis de píxeles para determinar cuáles están conectados entre sí, basado en algún criterio de vecindad. Esto permite separar texto de imágenes y distinguir texto perteneciente a un sector de otro.

2. **Organización Por Lineas de Texto**

   Los componentes conectados luego son organizados en *blobs* o Binary Large Objects, que son grupos de píxeles que comparten características como el color y tamaño, y que en el paso anterior se detectó que están conectados. Dado que estamos resolviendo OCR, los *blobs* luego se organizan en líneas de texto. Posteriormente, las líneas de texto se dividen en caracteres, teniendo en cuenta el espaciado entre ellos para intentar juntar los caracteres en palabras.

3. **Reconocimiento en Dos Pasadas**

   Ya que se está utilizando Redes Neuronales, las primeras partes del texto servirán como entrenamiento para la red. Es por esto que se necesitan dos pasadas para volver a analizar con una red suficientemente entrenada todas las partes del texto, no solo las finales. En la

primera pasada, se intentan reconocer todas las palabras posibles, las cuales luego entran en la red como datos de entrenamiento. En la segunda pasada, se analizan las palabras que no se lograron entender y se examinan con el contexto del nuevo aprendizaje.

4. **Resolución de Espacios Difusos y Altura de Letras**

   En una última fase, se revisan los espacios que no son tan claros para la herramienta en cuanto a dónde pertenecen. Con la información aprendida en las dos pasadas que se realizaron al documento, se revisan estos espacios y se les asigna una pertenencia. También se realizan hipótesis alternativas para la altura de las letras, con el fin de encontrar mayúsculas pequeñas (small-caps).

Si bien la herramienta ya lleva décadas en desarrollo, los detalles específicos dentro de cada paso de su funcionamiento no son públicos, es por esto que en el presente trabajo se detallan opciones posibles de su funcionamiento, incluyendo varias funciones de activación. Aunque, como se mencionó anteriormente, se sabe con certeza que el OCR de la herramienta usa LSTM-RNN para su resolución.

CAPÍTULO 5

RESOLUCIÓN

5.1. Condiciones Iniciales

Para este caso de estudio, se necesita que los archivos recibidos por el programa sean facturas electrónicas ecuatorianas con el formato estandarizado por el SRI (Servicio de Rentas Internas) y en formato PDF o JPG. Todos los archivos deben estar en una misma carpeta y debe escogerse una carpeta de salida para la salida de los archivos en JPG.

5.2. Cambio de Formato

Una condición previa de nuestra herramienta usada para el OCR, pytesseract, es que los archivos de input deben ser imágenes. El formato usual de las facturas electrónicas ecuatorianas es PDF, por lo que se necesita realizar un cambio de formato. Para esto, se usara el paquete Poppler y pdf2image, que van a ayudarnos a pasar de PDF a imagen, para que posteriormente sea usada por nuestro modelo con RNN-LSTM para extraer el texto de la imagen.

5.3. Preprocesamiento

Antes de aplicar el OCR, se realiza un preprocesamiento que consiste en hacer modificaciones a las imágenes para que la herramienta pueda distinguir con mayor facilidad los caracteres dentro de la misma. Para el presente trabajo, se llevaron a cabo los siguientes pasos de preprocesamiento con el paquete PIL/PILLOW (Python Imaging Library):

1. **Convertir a Escala de Grises**

Simplifica la imagen reduciendo la variabilidad de color, lo cual es útil ya que las facturas pueden presentarse en diversos colores, y para muchos tipos de análisis posterior, solo la intensidad de la luz es relevante.

2. **Binarización**

   Transforma la imagen en escala de grises a blanco y negro, facilitando la distinción entre el texto y el fondo, lo que es especialmente útil para mejorar la precisión del OCR.

3. **Mejora de Contraste**

   Incrementa la diferencia entre las áreas claras y oscuras de la imagen, resaltando los detalles y mejorando la legibilidad de los caracteres.

4. **Eliminación de Ruido**

   Reduce o elimina variaciones de brillo o color no deseadas (ruido), que pueden distorsionar la información importante, especialmente común en imágenes de documentos escaneados o de baja calidad.

5. **Ajuste de Brillo**

   Modifica la luminosidad general de la imagen para corregir problemas de sobreexposición o subexposición, asegurando que los detalles importantes sean visibles y no se pierdan en áreas demasiado claras u oscuras.

5.4. Extraer el Texto

Una vez que se ha realizado la preparación de los documentos para su tratamiento de OCR, se aplica la herramienta de pytesseract, la cual sabemos por 4.6.1 que se conforma por 4 pasos.

- **Pasos 1 y 2**

  Para el primer y el segundo paso, que consisten en el análisis de componentes conectados y la organización por líneas de texto, una función de activación muy posible de ser usada es la ReLU 4.5.4, no solo por la introducción de no linealidad al modelo y su rapidez de

tiempo de computación, sino también porque en estos dos primeros pasos, nuestro interés está en distinguir los distintos elementos y posteriormente blobs y líneas de texto entre sí. La función ReLU, al activar únicamente los valores positivos, ayuda a esta distinción con facilidad.

- **Paso 3**

  El tercer paso consiste en el reconocimiento en dos pasadas. Para esta tarea, las funciones tangente hiperbólica (tanh) 4.5.2 y sigmoide 4.5.1 son las más apropiadas.

  La función tanh tiene un rango de valores entre -1 y 1, lo que permite regular la información que pasa a través de la Red Neuronal Recurrente (RNN) y mantener gradientes fuertes. Se utiliza principalmente en la puerta de salida en los bloques de memoria de las LSTM.

  Por otro lado, la función sigmoide tiene un rango entre 0 y 1, siendo ideal para utilizar en la puerta de olvido y de entrada en nuestra LSTM. En estas puertas, la sigmoide decide qué información es relevante y debe permanecer en las siguientes generaciones de la red, y qué información debe ser olvidada. Esta decisión se realiza naturalmente considerando que 0 y 1 representan el 0 % y el 100 % de probabilidad de importancia, respectivamente.

- **Paso 4**

  El último paso es la resolución de espacios difusos y altura de letras. Para esta tarea, se requiere calcular probabilidades entre las diferentes opciones de espacios y alturas para determinar cuál de ellas escoger. La función de activación que mejor resuelve esto es la softmax 4.5.5, la cual es una función comúnmente utilizada para representar probabilidades entre distintas clases, asegurando que la suma total sea igual a 100 %.

  En este punto del reconocimiento óptico de caracteres (OCR), la red ya se encuentra entrenada, por lo que las decisiones se toman con conocimiento y aprendizaje previo. Es más probable que softmax nos dé la respuesta correcta en este contexto.

5.5. Almacenamiento del texto

Una vez realizado el OCR, los resultados se guardarán en un archivo con formato JSON, que es un formato editable y estructurado.

CAPÍTULO 6

RESULTADOS

Para comprobar la precisión del algoritmo, se realizo un test sobre 200 facturas electrónicas ecuatorianas, evaluando si el OCR lograba identificar exitosamente los parámetros:

- Valor Total
- Subtotal
- RUC
- la palabra FACTURA

Las facturas tuvieron como máximo dos hojas, aunque en su gran mayoría consistían solo de una. Cada archivo tomó alrededor de 5.79 segundos en ser procesado y plasmado en el archivo JSON. El resultado de el test de precisión fue el siguiente:

	Acertados	**Equivocados**	**Precisión**
RUC	151	49	75.5 %
Palabra Factura	151	49	75.5 %
Valor Total	173	27	86.5 %
Subtotal	172	28	86.0 %

Cuadro 6.1: Test de precisión.

Como se puede ver, se tiene una mayor precisión en el reconocimiento del valor total y el subtotal en comparación con el RUC y la palabra 'factura'. De hecho, tienen un porcentaje similar entre el RUC y la palabra 'factura', así como entre el valor total y el subtotal. En los 4 casos, la precisión supera el 70 %. Es importante notar que el RUC, el Valor Total y el Subtotal son valores numéricos, mientras que la palabra 'FACTURA' son caracteres de texto, pero que RUC y FACTURA suelen estar al inicio del documento y el valor total y subtotal a la mitad o al

final del mismo.

A continuación se presenta un ejemplo de una factura y de su resultado una vez aplicado el OCR:

GRUPO
LA RAMPA

FACTURA **No.001-001-000005513**

Número de Autorización:
1401202301179319430900120010010000055130000577516

Fecha y hora de Autorización:
15/01/2023 08:36:02

Ambiente: PRODUCCION
Emisión: NORMAL
Clave de Acceso:
1401202301179319430900120010010000055130000577516

Emisor: LA RAMPA S.A.S.
RUC: 1793194309001
Matriz: PICHINCHA / QUITO / QUITO DISTRITO METROPOLITANO / AVENIDA ORELLANA Y AMAZONAS
Correo: caja.guayacacumbaya@gmail.com
Teléfono: 024016112
Obligado a llevar contabilidad: SI

Razón Social: VISION WIDE SOC. CIVIL
Dirección: xxxxxx
Fecha Emisión: 14/01/2023

RUC/CI: 1792886376001
Teléfono:
Correo:

Código Principal	Cantidad	Descripción	Detalles Adicionales	Precio Unitario	Descuento	Total
CEV057	1.00	GUAYACO CHOLO		10.980000	$0.00	$10.98
CEV047	1.00	GUAYACO CRIOLLO		10.980000	$0.00	$10.98
BEB016	1.00	LIMONADA		2.670000	$0.00	$2.67
BEB017	1.00	LIMONADA IMPERIAL		3.130000	$0.00	$3.13

Información Adicional

Descripción VENTA PUNTO DE VENTA

Formas de pago

Tarjeta de crédito $31.09 0 días

Subtotal Sin Impuestos:	$27.76
Subtotal 12%:	$27.76
Subtotal 0%:	$0.00
Subtotal No Objeto IVA:	$0.00
Descuentos:	$0.00
ICE:	$0.00
IVA 12%:	$3.33
Servicio %:	$0.00
Valor Total:	$31.09

Figura 6.1: Ejemplo Factura

```
ORUPO FACTURA No.001-001-000005513
Número de Autorización:
1401202301179319430900120010010000055130000577516
Fecha y hora de Autorización:
15/01/2023 08:36:02
Emisor: LA RAMPA S.A.S.
RUC: 1793194309001 Ambiente: PRODUCCION
Matriz: PICHINCHA / QUITO / QUITO DISTRITO Emisión: NORMAL
METROPOLITANO / AVENIDA ORELLANA Y .
AMAZONAS Clave de Acceso:
Correo: caja.guayacacumbaya(QOgmail.com
Teléfono: 024016112
Obligado a llevar contabilidad: Sl
1401202301179319430900120010010000055130000577516
Razón Social: VISION WIDE SOC. CIVIL RUC/CI: 1792886376001
Dirección: xxxxxx Teléfono:
Fecha Emisión: 14/01/2023 Correo:
Código Cantidad Descripción Detalles Precio Descuento Total
Principal Adicionales Unitario
CEVO57 1.00 GUAYACO CHOLO 10.980000 $0.00 $10.98
CEVO47 1.00 GUAYACO CRIOLLO 10.980000 $0.00 $10.98
BEBO16 1.00 LIMONADA 2.670000 $0.00 $2.67
BEB017 1.00 LIMONADA IMPERIAL 3.130000 $0.00 $3.13
Información Adicional Subtotal Sin Impuestos: $27.76
Subtotal 12%: $27.76
Descripción VENTA PUNTO DE VENTA Subtotal 0%: $0.00
Subtotal No Objeto IVA: $0.00
Formas de pago
Descuentos: $0.00
Tarjeta de crédito $31.09 O días ICE: $0.00
IVA 12%: $3.33
Servicio %: $0.00
Valor Total: $31.09
```

Figura 6.2: Ejemplo Resultado OCR

CAPÍTULO 7

CONCLUSIONES

El trabajo presentado demuestra un enfoque matemático y computacional para el reconocimiento de texto en documentos PDF, con un enfoque particular en la extracción de información de facturas electrónicas. A través de la implementación de Redes Neuronales, específicamente Redes Neuronales Recurrentes (RNN) y Long Short-Term Memory (LSTM), se destaca su eficacia para abordar el problema de OCR, que, a pesar de los avances, sigue siendo un desafío en el campo de la visión por computadora. La elección de estas arquitecturas es relevante debido a su capacidad para procesar secuencias de datos y manejar dependencias de largo plazo, aspectos cruciales en el reconocimiento de texto.

El análisis realizado en una computadora de gama media, logrando procesar cada archivo en aproximadamente 5.79 segundos, y su capacidad para manejar el procesamiento por carpetas sugiere un rendimiento prometedor para aplicaciones a gran escala. Sin embargo, para proyectos que requieran el procesamiento de un volumen significativamente mayor de documentos o archivos de mayor complejidad, se recomienda la implementación en servidores, servicios web o clusters de alta capacidad para asegurar la escalabilidad y eficiencia del sistema.

La información disponible actualmente sobre Tesseract apunta al uso exclusivo de redes neuronales RNN y LSTM. Por lo tanto, nos centramos únicamente en estas arquitecturas. Sin embargo, para trabajos futuros, se sugiere explorar el uso de Redes Neuronales Convolucionales (CNN) en los primeros pasos del OCR, especialmente para la segmentación de imágenes.

En conclusión, este trabajo subraya a las Redes Neuronales Recurrentes (RNN) y las Long Short-Term Memory (LSTM) como una posible solución al reconocimiento óptico de caracteres,

ofreciendo un caso de estudio sobre las facturas electrónicas. Al mismo tiempo, reconoce la necesidad de adaptabilidad y escalabilidad en aplicaciones de mayor dimensión, sugiriendo el uso de infraestructuras computacionales más robustas para proyectos de mayor escala. Asimismo, reconoce un potencial para estos posibles proyectos, principalmente tomando en cuenta la accesibilidad que existe sobre herramientas tecnológicas como lo son cámaras y procesadores.

Referencias

Aggarwal, C. C. (2018). *Neural networks and deep learning*. Springer.

Deepa, B., y SS, K. (2014). Optical character recognition: an overview and an insight. *2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*, 1361–1365.

Gobierno Nacional de la República del Ecuador. (2021). *Código orgánico de planificación y finanzas públicas.* (Recuperado de https://biblioteca.defensoria.gob.ec/handle/37000/3401)

Goodfellow, I., Bengio, Y., y Courville, A. (2016). *Deep learning*. MIT Press. (`http://www.deeplearningbook.org`)

Graves, A. (2012). *Supervised sequence labelling with recurring neural networks*. Studies in Computational Intelligence.

J, K. (2021). Which ocr toolset is good and why? a comparative study. *Kuwait Journal of Science*.

James, G., y Witten, D. (2023). *An introduction to statistical learning*. Springer.

Lichtner-Bajjaoui, A. (2020). *A mathematical introduction to neural networks* (Tesis de Master). Universitat de Barcelona.

Merenda, M., Porcaro, C., y Iero, D. (2020). Edge machine learning for ai-enabled iot devices: A review. *Sensors*, *20*(9), 2533.

Nagy, G. (2000). Twenty years of document image analysis in pami. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, *22*(01), 38–62.

Naseer, A., y Zafar, K. (2018). Meta features-based scale invariant ocr decision making using lstm-rnn. *Computational and Mathematical Organization Theory*, *25*, 165–183.

Nielsen, M. A. (2015). *Neural networks and deep learning* (Vol. 25). Determination press San Francisco, CA, USA.

Ooms, J. (2023). tesseract: Open source ocr engine [Manual de software informático]. Descargado de `https://docs.ropensci.org/tesseract/(website)` `https://github.com/ropensci/tesseract(devel)` (R package version 5.2.1https://docs.ropensci.org/tesseract/ (website) https://github.com/ropensci/tesseract (devel))

Singh, J., y Bhushan, B. (2019). Real time indian license plate detection using deep neural networks and optical character recognition using lstm tesseract. En *2019 international conference on computing, communication, and intelligent systems (icccis)* (pp. 347–352).

Smith, R. W. (2013). History of the tesseract ocr engine: what worked and what didn't. En *Document recognition and retrieval xx* (Vol. 8658, p. 865802).

Wang, H., Pan, C., Guo, X., Ji, C., y Deng, K. (2021). From object detection to text detection and recognition: A brief evolution history of optical character recognition. *Wiley Interdisciplinary Reviews: Computational Statistics*, *13*(5), e1547.