UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Grado en Ingeniería Informática
Especialidad de Computación

Mejora de la segmentación de documentos digitales usando nuevas arquitecturas de redes neuronales

Daniel Cano Carascosa

Director: Jordi Torres Viñals
Codirector: Juan Luis Dominguez

Resumen

En los últimos años se han podido ver grandes avances en el área de las redes neuronales en diferentes áreas que engloban tanto lo visual como detección de objetos en imágenes, como análisis o generación de textos, siendo así capaces de generar historias creíbles a tiempo real.

El objetivo de este proyecto es hacer uso de ambos tipos de arquitecturas, tanto visuales como textuales y aplicarlas en el problema de segmentación de documentos donde dada una lista de páginas, la red tendrá que dividir la lista de páginas para decidir cuando comienza y acaba un documento, el cual puede estar formado por una o más páginas.

Para lograr esta tarea se seleccionan datasets públicos ya conocidos y se analizan, encontrando y corrigiendo fallos que pueden afectar a nuestro proyecto. Gracias al uso y unión de redes dedicadas imágenes y texto, podremos ver como los modelos propuestos no solo igualan a propuestas previas que se consideran estado del arte en esta área, sino que en algunos casos obtienen resultados superiores a los mostrados en investigaciones anteriores.

Resum

En els darrers anys s'han pogut veure grans avenços en l'àrea de les xarxes neuronals en diferents àrees que engloben tant allò visual com detecció d'objectes en imatges, com anàlisi o generació de textos, i així són capaços de generar històries creïbles a temps real.

L'objectiu d'aquest projecte és fer ús d'ambdós tipus d'arquitectures, tant visuals com textuals i aplicar-les al problema de segmentació de documents on donada una llista de pàgines, la xarxa haurà de dividir la llista de pàgines per decidir quan comença i acaba un document, el qual pot estar format per una o més pàgines.

Per aconseguir aquesta tasca se seleccionen datasets públics ja coneguts i s'analitzen, trobant i corregint errors que poden afectar el nostre projecte. Gràcies a l'ús i unió de xarxes dedicades imatges i text, podrem veure com els models proposats no només igualen propostes prèvies que es consideren estat de l'art en aquesta àrea, sinó que en alguns casos obtenen resultats superiors als mostrats en investigacions anteriors.

Abstract

In recent years, great advances have been seen in the area of neural networks in different areas that encompass both the visual and the detection of objects in images, such as analysis or text generation, thus being able to generate credible stories in real time.

The objective of this project is to make use of both types of architectures, both visual and textual, and apply them to the document segmentation problem where given a list of pages, the

network will have to divide the list of pages to decide when a document begins and ends, which can consist of one or more pages.

To achieve this task, already known public datasets are selected and analyzed, finding and correcting faults that may affect our project. Thanks to the use and union of dedicated images and text networks, we will be able to see how the proposed models not only match previous proposals that are considered state of the art in this area, but also in some cases obtain superior results than those shown in previous research.

Agradecimiento

Realizar un proyecto de esta envergadura supone un esfuerzo supone una inversión de tiempo muy notable, por lo que en primer lugar quiero agradecer a mi familia, la cual me ha soportado estos 4 meses en los que no he estado con ellos debido a que estaba ocupado con este proyecto, pero que aun así me han dado todo su apoyo en todo momento

También me gustaría agradecer al grupo BSC [1] y a mi director y codirector de proyecto, ya que son gracias a ellos dos que ha sido posible llevar a cabo este proyecto, ya que en momentos donde nos encontrábamos con un reto a enfrentar siempre me han sido una guía, bien dándome consejos, conocimientos o experiencias que de otra forma hubiera o bien hubiera tardado demasiado en adquirir y por ello el proyecto no se hubiera finalizado a tiempo o bien no hubiera sido capaz de adquirir en la ventana de tiempo que teníamos disponible.

Sin el soporte y paciencia, ya que a veces puedo tener momentos en los que me cuesta comprender algún concepto, de mis directores y familiares, este proyecto no hubiera sido posible, por ello les expreso mis más sinceros agradecimientos.

Índice de Contenidos

Índice de tablas

Índice de figuras

1 Introducción

Hace aproximadamente 15 años las redes neuronales no jugaban un papel muy importante o destacado en ningún sector más allá del de la investigación, aunque como todos hemos podido ver hoy en día ya no es el caso, ya que actualmente hemos visto un resurgir de estas técnicas las cuales han pasado de no tener un papel muy importante a tener o bien un papel fundamental e incluso indispensables en sectores como reconocimiento de objetos en imágenes [2], detector de poses de personas dada una imagen [3] o traductores [4].

Como la mayoría de nosotros hemos podido observar, los grandes avances de los últimos años han sido en el ámbito de las redes relacionadas con imágenes, ya que son estas las que visualmente más impactan y por lo tanto las más conocidas. No obstante en los últimos 2 años también se han estado observando importantes avances en el ámbito de las redes que tratan con secuencias temporales, es decir datos que guardan relación en el tiempo, algunos ejemplos podrían ser los valores de la bolsa para el oro, un texto, ya que cada palabra depende temporalmente de la anterior o sonidos como una conversación donde el tono que vocalizamos a cada momento depende del anterior para formar un significado conjunto como por ejemplo una palabra.

En este TFG se pretende desarrollar un modelo que haciendo uso de los últimos avances en análisis de imágenes y texto con redes neuronales, dada una secuencia de N imágenes, donde cada imagen será una página escaneada, decidir cuando termina un documento y cuando empieza el siguiente.

1.1 Contexto

Dentro de la facultad de informática de Barcelona existen diferentes menciones. Este trabajo de fin de grado pertenece a la mención de computación, concretamente al de Inteligencia Artificial y se ha realizado con la ayuda y cooperación del BSC [1].

El proyecto parte de la necesidad de diferentes organismos en la actualidad a gestionar una gran cantidad de documentos de forma diaria, como lo pueden ser por ejemplo bancos o universidades.

Una vez completado el proyecto, cualquier organización o particular será capaz de dividir en documentos un conjunto de páginas, según la relación entre estas, lo cual permitirá automatizar este proceso.

1.2 Identificación del problema

En este trabajo se busca encontrar cuando termina y cuando acaba un documento, tarea que para nosotros puede parecer simple, pero cuando te encuentras en una entidad bancaria, por ejemplo, y te llegan cerca de cientos de miles imágenes de documentos sin dividir cada día, entonces nos encontramos con un problema bastante importante de logística. Por lo que es en estas situaciones donde es de vital importancia generar un sistema automático que se encargue de este proceso. No

obstante para resolver este problema se deben lidiar tanto con la imagen de los documentos como con el texto de los mismos, ya que de no hacerlo se pueden crear situaciones donde el rendimiento obtenido disminuya.

Por ejemplo en el contexto donde dado una página de un documento solo tenemos acceso a la imagen de esta, la red no será capaz de identificar el contexto del texto asociado.

Por otro lado, si solo le informamos a la red de la información visual, es posible que alguna página de un documento solo contenga una gráfica, pero el modelo al no tener acceso a esta información podría entender que es la última página del documento.

Por lo que nuestro modelo en cuestión hará uso de ambas fuentes de información para intentar resolver este problema.

1.3 Agentes implicados

La herramienta que se va a desarrollar en este proyecto ofrece un gran beneficio a grandes entidades como bancos o cualquier organismo que tenga algún tipo de trámite que implique diversos documentos, ya que estos en múltiples ocasiones son bien o escaneados como un solo documento o enviados por los usuarios a través de sistemas informáticos como un solo fichero, por lo que en frente de esta situación donde se pueden recibir millones de documentos mensuales es necesaria una herramienta automatizada.

1.4 Alcance

A continuación se definen los objetivos, sub-objetivos, los requerimientos funcionales y no funcionales y riesgos posibles debido a que como el tiempo disponible para realizar el proyecto es limitado tenemos que tener todos estos factores presentes a la hora de distribuir nuestro tiempo.

1.4.1 Objetivos

Los objetivos principales de este proyecto son los siguientes:

1. Ser capaz de construir un modelo que haciendo uso de Transformers y redes convolucionales combinadas sea capaz de resolver el problema con un mínimo de un 70% de *accuracy*.
2. Superar o igualar resultados de los otros trabajos de investigación presentados, demostrando así que nuestro enfoque es o bien una alternativa válida y por lo tanto una alternativa con potencial para ser ampliada o bien una mejor alternativa que demuestre que la combinación de redes convolucionales y Transformers supera con creces los anteriores métodos propuestos.
3. Usar métodos de paralelización para reducir tiempos de entrenamiento para que se pueda obtener la red deseada en un tiempo mínimo.

1.5 Requerimientos

1.5.1 Requerimientos funcionales

A continuación se describen los requisitos funcionales para llevar a cabo este proyecto y también para su posterior uso en caso de que este se quisiera usar en una empresa.

1. Compatibilidad con cualquier dispositivo informático actual para que su ejecución sea posible en cualquier máquina, siempre que esta disponga de una GPU.

1.5.2 Requerimientos no funcionales

A continuación se describen los requisitos no funcionales, es decir aquellos requerimientos que no tienen relación directa con el funcionamiento del sistema, pero que se deben de tener en cuenta.

1. Rapidez: Como todos sabemos hoy en día las aplicaciones que usamos tienen unos tiempos de respuesta muy pequeños, ya que hay situaciones en donde esta velocidad es necesaria. En este proyecto se entiende que puede no parecer de vital importancia procesar rápidamente las páginas, pero si tardáramos 100ms para procesar cada par de páginas, en una situación donde tenemos 1.000.000 de páginas, una situación no muy difícil de imaginar como ya he propuesto en otros apartados, estaríamos hablando de 70 días para procesar toda esta información. Por lo que el apartado de rapidez no se debe ignorar.
2. Usabilidad: El modelo debe ser sencillo de usar para que alguien que no tenga conocimientos avanzados en esta área sea capaz de utilizar el modelo dentro de un ámbito de empresa.
3. Reusabilidad: Dado que en esta área existe una técnica llamada *transfer learning*, donde el modelo a partir de conocimiento ya adquirido aprende nuevas tareas, este modelo podría ser entrenado documentos bancarios y posteriormente ser entrenado en alguna otra tarea de las propuestas al inicio de este documento, como división de capítulos dentro de un libro, ya que como se ha visto en otras investigaciones *transfer learning* nos permite obtener mejores resultados de los que obtendríamos con la misma red entrenando des de cero [5].

2 Metodologia

2.1 Conceptos

El área del Deep Learning es muy extensa y compleja, donde los más mínimos detalles pueden marcar la diferencia no solo en rendimiento, sino en resultados, es por ello que en esta sección, aunque se definan los conceptos necesarios para comprender este proyecto solo estaremos definiendo una pequeña parte de esta área tan extensa.

A continuación se definen los conceptos clave para el proyecto, por lo que a partir de este momento usaré libremente los siguientes términos en múltiples ocasiones.

Tensor

Un tensor se puede entender como objetos que almacenan valores numéricos y que pueden tener distintas dimensiones, permitiendo así expresar conceptos como orden temporal entre otros. Por lo que un tensor de 0 Dimensiones es un escalar, 1 Dimensión es un vector, de 2D una matriz y de 3D un cubo.

Hiperparámetro

Se llama hiperparámetro a todos los parámetros que influyen en el entrenamiento del modelo y, por lo tanto, en su rendimiento final y que son establecidos antes del comienzo del mismo.

Capa Densa

Una capa densa [6] es una capa de red neuronal donde todas las neuronas que forman esta capa están densamente conectadas con todas las neuronas de la capa anterior, lo que significa que cada neurona de la capa densa recibe información de todas las neuronas de la capa anterior.

Capa convolucional

Las capa convolucionales son el pilar de las redes convolucionales, y estas están compuestas por uno o más filtros llamados habitualmente Kernels, cuyos parámetros son modificados en el proceso de entrenamiento. Todos los filtros realizan una convolución con la imagen, haciendo que el tamaño de la imagen original sea reducida y el número de operaciones necesarias para la siguiente capa.

Red Convolucional

Una red convolucional es un tipo de red neuronal formada por capas convolucionales la cual se especializa en procesar imágenes [7].

Existen excepciones donde se pueden usar capas convolucionales que no se encargan de procesar imágenes si no texto [8]. No obstante como en este caso no aplicaremos esta aproximación no tendremos en cuenta estos casos.

Red Recurrente

Una red recurrente [9] está formada por neuronas conectadas en forma de grafo dirigido, siendo así capaces de mostrar un comportamiento adaptable a dependencias temporales en datos, permitiendo que sean usadas en tareas como reconocimiento de voz o de análisis texto.

Transformers

Los transformers[10] son una arquitectura de Deep Learning que aplican el mecanismo de atención a los datos de entrada, el cual consiste en ponderar cada dato de entrada según su importancia y relevancia respecto los otros datos de entrada.

Estas redes no solo han demostrado un alto rendimiento tratando con imágenes [11] sino también con textos [12].

Vector Latente

En muchas situaciones me referiré a la información que se encuentra entre la entrada y la salida de una red neuronal, es decir aquella información que solo la red entiende y nosotros no podríamos entender, como vector latente.

Un vector latente se puede entender como una secuencia de números flotantes que codifican, en el caso de una imagen, el tipo de imagen, cuantas personas hay en la imagen, si la imagen tiene tonos claros, si tiene una situación de peligro o cualquier información que la red considere importante guardar.

Modelo Pre-entrenado

Un modelo pre-entrenado es aquel que ya se ha sometido a un proceso de aprendizaje y que, por lo tanto, ha aprendido a abstraer y procesar información de su entrada siempre que esté relacionada con lo que aprendió en el entrenamiento, ya que en caso de un modelo que aprenda a analizar textos en inglés solo tendrá conocimiento de este lenguaje por lo que toda entrada deberá estar en inglés.

Transfer learning

Se denomina *transfer learning* a la capacidad de que un modelo ya pre-entrenado, sea capaz de aprender una nueva tarea, haciendo uso de los conocimientos previos obtenidos. Por lo que si se seleccionara un modelo entrenado en inglés y se le entrenara con portugués, este aprendería más rápido, ya que podría hacer uso tanto de gramática como de estructura sintáctica que aprendió de los textos en inglés.

Loss function

La *Loss function* o también conocida como la función de error, es la función que determina para cada salida de nuestra modelo, el error que se ha cometido respecto el valor ideal deseado.

Learning Rate

Se define *Learning Rate* el parámetro del algoritmo de optimización llamado *Gradient Descent*[13], que determina el tamaño del paso en cada iteración hacia el punto mínimo en la función de *Loss*. Dado que este parámetro influye en el tamaño del paso, se debe tener en cuenta que un *Learning Rate* muy pequeño tendrá el efecto de un proceso de entrenamiento lento, mientras que uno demasiado elevado podría evitar alcanzar el punto mínimo debido a que si la longitud de estos pasos fueran demasiado grandes, podría ocurrir que nunca convergiera en el punto mínimo.

Batch

Batch se puede entender como cuantos datos van a ser suministrados a la imagen en una sola llamada, por lo que cuanto mayor sea este valor, mayor será la velocidad de entrenamiento.

Epoch

Epoch se puede entender como el número de veces que nuestro modelo aprenderá de todo nuestro conjunto de datos de entrenamiento y el número de veces que el modelo se evaluará con los datos de validación.

2.2 Soluciones existentes

Por supuesto como ya he mencionado este problema no solo existe actualmente, sino que supone un gran reto para muchísimas empresas que generan miles o millones de documentos de forma diaria y de forma constante, por lo que como se puede suponer ya se han hecho investigaciones e implementaciones que intentan solucionar este problema de la mejor forma posible.

A continuación expondré los últimos trabajos publicados que miran de resolver este problema en cuestión y los métodos que utilizan.

2.2.1 *Leveraging effectiveness and efficiency in Page Stream DeepSegmentation*

El paper que actualmente es el estado del arte [14] hace únicamente uso de imágenes con redes convolucionales, siendo la red seleccionada VGG16 [15].

Para lograr esta tarea los autores del paper emplean dos técnicas. La primera es donde únicamente hacen empleo de dos páginas siendo estas consecutivas y decidiendo si pertenecen o no al mismo documento, y otro método donde hacen uso de 3 páginas, y nuevamente deciden si pertenecen o no al mismo documento. Por lo que en su implementación como VGG16 solo puede procesar una imagen a la vez llaman al modelo tres veces, extraen la información del modelo previa a las últimas capas Densas y finalmente la concatenan y la pasan por una última capa Densa.

Los resultados que obtienen son bastante prometedores, llegando a obtener hasta un 92% de *accuracy* en un dataset de benchmark llamado Tobacco800 y 91.1% en un dataset propio que ellos mismos han generado manualmente.

2.2.2 *Page Stream Segmentation with Convolutional Neural Nets Combining Textual and Visual Features*

Ya ha habido previamente equipos que han intentado esta misma tarea haciendo uso únicamente de redes convolucionales, pero procesando imagen y texto [16]. Su estrategia fue la de, dada una situación donde tenemos N páginas, coger las páginas k_{-1} y k_0, y procesar las imágenes y el texto por separado, pero ambos elementos con redes convolucionales.

En primer lugar, se entrenaban dos redes, una intentando separar los documentos únicamente con imágenes y la otra únicamente con texto, haciendo que las dos redes, aunque no tuvieran resultados muy prometedores, estuvieran pre entrenadas.

La forma de entrenar estas redes serian coger, en el caso de la red que recibe las imágenes k_{-1} y k_0, introducirlas por la red convolucional, obtener dos vectores latentes, que posteriormente concatenaría e introduciría a una red Densa para obtener el resultado, y lo mismo para el texto.

Y posteriormente y una vez entrenadas por separado, se agruparían por lo que obtendríamos 4 vectores latentes que contendrían información sobre estas páginas tanto en imagen como en texto, las cuales se introducen por una capa densa para obtener finalmente un número que se situara entre el 0 y el 1 para darnos un porcentaje de como de fiable cree el modelo que las dos páginas en cuestión pertenecen a dos documentos diferentes.

Los resultados que obtiene este paper son muy cercanos al estado del arte, logrando un *accuracy* de 91% y un score de Kappa de 81.6% en Tobacco800.

2.2.3 *Use of language models for document stream segmentation*

Por supuesto al igual que hay quien ha hecho uso únicamente de convolucionales también hay quien ha intentado abordar el problema haciendo uso de redes recurrentes [17], las cuales como ya hemos mencionado, son capaces de abstraer propiedades temporales entre los datos, aunque son menos potentes que los Transformers como ya se ha demostrado previamente en diferentes investigaciones.

Los resultados que obtienen, pese a no estar dentro del estado del arte se encuentran bastante cerca, teniendo un *accuracy* del 90% en Tobacco800 y un *accuracy* de 96% con un dataset llamado READ-CORPUS, el cual tal y como ellos mencionan es privado.

2.3 Evaluación de modelos

Para evaluar nuestro modelo necesitamos de tres elementos.

1. El primero debe ser unos datos de entrenamiento, con los que entrenar a nuestro modelo. Este conjunto de datos lo llamaremos datos de entrenamiento.
2. El segundo debe ser unos datos de evaluación, donde ver como de bien funciona nuestro modelo, ya que si en estos datos nuestro modelo no funciona correctamente esto querrá decir que no ha aprendido de los datos de entrenamiento, sino que los ha memorizado. Este conjunto de datos lo llamaremos datos de validación.
3. En tercer lugar, tenemos los datos de test los cuales nos sirven para emular como funcionaría el modelo en un entorno real. Ya que si solo nos basáramos en los datos de validación estaríamos suponiendo que los datos de validación representan el mundo real, es decir desconocido, por lo que este tercer grupo no solo tiene como objetivo evaluar al modelo, sino evitar nuestra influencia sobre el resultado.

Un ejemplo visual y fácil de entender se puede visualizar en la figura 1, donde con únicamente un dataset generamos estos tres grupos independientes.

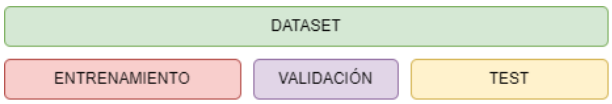


Figure 1: Segmentación del Dataset en Entrenamiento, validación y Test.

Además de esto, por nuestra parte también aleatorizamos los datos dentro de cada grupo, es decir eliminamos todo orden posible, para así evitar que el mismo orden de los datos genere algún beneficio oculto en el entrenamiento, esta aleatorización se aplica únicamente en las subsecciones generadas de entrenamiento, validación y test.

No se aplica en el conjunto global, antes de hacer la división, debido a que en la propia web donde se suministra el dataset que se va a usar, se adjunta un csv indicando los datos que van a ir a la

sección de entrenamiento, validación y test. Esta división se ha respetado, ya que como los mismos autores del dataset explican esta división está pensada para que en entrenamiento, validación y test, el número de documentos de que pertenezcan a una misma clase, sea proporcionalmente el mismo a los de los otros subconjuntos. Por lo que si en entrenamiento un 10% de los documentos pertenecen a ciencia, en validación y entrenamiento también habrá un 10% de documentos científicos respecto el total.

Finalmente, para medir el rendimiento de nuestros modelos, se toman 3 medidas diferentes a usar, aunque para entender estos conceptos de una forma más fácil nos basaremos en la figura 2:

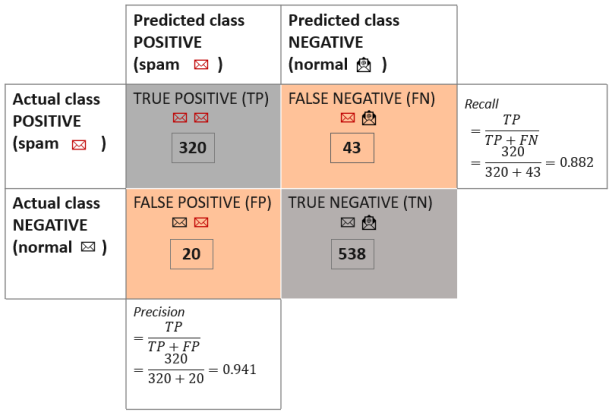


Figure 2: Matriz de confusión.

1. *Accuracy*: La *accuracy* medirá el porcentaje de aciertos respecto el total, pero esta métrica presenta un problema en el problema con el que estamos tratando, ya que en un caso donde tenemos 2 documentos, de 10 páginas cada uno, si nuestro modelo decidiera que hay dos documentos de 5 y 15 páginas, el *accuracy* sería del 85%, ya que de todas las veces que se le ha pedido identificar si dos páginas son diferentes documentos o no, ha tenido éxito en un 85% de las ocasiones, por lo que pese a lo que nosotros consideraríamos un error de importancia el modelo considera que ha tenido una gran tasa de acierto. La fórmula para obtener este valor seria $Accuracy = \frac{TP+TN}{TP+FP+FN+TN}$
2. F1 Score: Para lidiar en parte con lo que hemos observado que puede suceder con el *Accuracy* también hemos implementado F1 Score, aunque para entenderlo antes se deben entender dos conceptos:
   - Precisión: Cuando hablamos de la precisión de un modelo estamos respondiendo a la pregunta "¿De todos las páginas etiquetadas como inicios de documento, cuantas de estas son realmente inicio de documentos?" Por lo que cogiendo como ejemplo el anterior caso con 2 documentos de 10 páginas obtendríamos una precisión de 0, ya que ninguna de las páginas marcadas como inicio de documento realmente lo es. La fórmula para obtener este valor es $Precision = \frac{TP}{TP+FP}$. El problema de esta métrica por si sola es que ignora los falsos negativos, es decir no tiene en cuenta cuantas veces fallamos diciendo que dos páginas pertenecen al mismo documento, solo cuando decimos que pertenecen a diferentes documentos esta métrica entra en juego.

- Recall: Esta métrica responde a la pregunta "¿De todas las páginas que eran comienzo de documento, cuantas de ellas hemos etiquetado?". Con esta métrica se busca detectar la sensibilidad de nuestro modelo, es decir que tan bien es capaz de detectar. No obstante esta métrica por sola tiene el problema que en un caso de etiquetar todo como nuevo documento tendríamos un Recall del 100%, ya que no tiene en cuenta la precisión sino el número de falsos negativos que nuestro modelo ha dado. La fórmula para calcularlo es $Recall = \frac{TP}{TP+FN}$.

  Por lo que F1 Score se basa en coger estos dos conceptos y combinarlos2222222222222, haciendo que los puntos débiles de una se vean compensados por la otra y la forma que tiene de unirlas es mediante la fórmula $F1\_Score = \frac{2*(Recall*Precision)}{(Recall+Precision)}$.

3. Kappa: Esta métrica mide de manera cuantitativa la cantidad de veces que un observador aleatorio y nuestro modelo, coincidirán en que páginas pertenecen al inicio de un nuevo documento. Que dicho con en otras palabras, Kappa nos dirá cuantitativamente la probabilidad, de que dos sujetos, o modelos, piensen lo mismo delante de unos datos, es decir que haya ambos puntos de vista se pongan de acuerdo. Esto nos sirve para, en caso de que viniera un tercer observador (otro modelo), este debería de "estar de acuerdo" con nuestro modelo.

Por lo que con estos datos presentes, en todos los experimentos se hará una ronda de entrenamiento y una de validación, donde obtendrá el *Accuracy*, F1 Score y Kappa, y se seleccionará el modelo con más F1 Score, en caso de empate el modelo con un *Accuracy* más elevado y en caso de empate el modelo con Kappa más elevado. Una vez seleccionado el modelo con los parámetros más prometedores, se realizará una prueba de Test, que es la que finalmente presentaremos en la tabla de resultados.

3 Datos

3.1 Obtención de los datos

El dataset utilizado para entrenar a la red es BigTobacco (también llamado RVL-CDIP) el cual es un dataset público [18], que contiene 400,000 documentos. Estos documentos se clasifican en 16 clases diferentes y sus dimensiones no superan los 1000 pixeles. Hay 25,000 documentos para cada clase por lo que el dataset está balanceado

En la propia página se ofrece unos ficheros .csv que nos distribuyen los datos para así obtener 320.000 documentos de entrenamiento, 40.000 de validación y 40.000 de test respetando el número de diferentes clases de documentos en cada sección.

3.2 Generación de Texto

Para generar los datos se ha hecho uso de una herramienta llamada pytesseract[19]. Esta librería nos permite analizar una imagen, y extraer el texto que se encuentra en esta, ya que en los datos que disponemos no contienen el texto de las imágenes de los documentos.

Para el entrenamiento de los modelos se hará empleo de los procesadores *IBM Power9 8335-GTH @ 2.4GHz*, pero para poder hacer uso de esta librería de la forma más eficiente se ha tenido que usar procesadores *AMD EPYC 7742 @ 2.250GHz*, los cuales tienen incorporadas operaciones vectoriales, que la librería de pytesseract usa para generar el texto. Gracias a esto el tiempo para generar el texto es 23 días, que es un speed-up del 2.6 respectos los procesadores de IBM.

Para acelerar aún más el proceso se aplica una estrategia de paralelización, donde se lanzarán 32 procesos, y cada proceso será responsable de 12.500 documentos. Por lo que para finalmente generar todos los datos, serán necesarios 3 días y 12 horas. No obstante como el entorno en el que trabajamos una tarea solo puede estar en ejecución durante 48h, se decide dividir en 4 tareas.

Por lo que al finalmente se lanzarán 4 tareas, donde cada tarea procesará 100.000 documentos y cada proceso generará el texto de 3.125 documentos, lo que nos deja con una estimada de tiempo de 21h 30min por cada tarea, que si observamos no es el speed up ideal, pero tras una detallada observación se ha observado que la gestión y sincronización de 32 procesos supone una gran carga en tiempo para el entorno en el que nos encontramos.

Para conservar la información cada proceso conservará el texto extraído en ficheros JSON que posteriormente serán usados para la generación de ficheros H5DF.

3.3 Generar ficheros H5DF

Para conservar los datos y consultarlos de forma eficiente se ha decidido utilizar ficheros en formato H5DF. El formato H5DF es un formato de archivo de código abierto que admite datos grandes,

complejos y heterogéneos. HDF5 utiliza una estructura similar a un árbol N-ario, que le permite organizar los datos dentro del archivo de muchas maneras estructuradas diferentes, de manera eficiente, evitando la fragmentación de memoria y permitiendo que los datos se puedan acceder eficientemente en todo momento gracias a su estructura, lo cual es esencial para que la lectura de los datos no resulte ser un cuello de botella en el entrenamiento.

Haciendo uso de los ficheros CSV proporcionados, aplicamos la división de los documentos en las categorías de *train*, validación y *test*, donde un documento se compone de 1 a N imágenes y sus respectivos OCRs. Seguidamente, se generan las *labels* para cada página de cada documento, las cuales solo tendrán dos valores, siendo estos 0 y 1, donde 0 indicará que la página no es la primera página de un documento, y 1 indicará que sí que es la primera página de un documento. Debido a que el dataset BigTobacco nos ofrece los documentos ordenados, podemos asignar sin problemas estas *labels* a cada página.

A la hora de generar los ficheros H5DF, se plantearon dos estrategias, ya que se debía tener en cuenta que los datos almacenados en estos ficheros serán cargados en RAM, por lo que se debe elegir un método que no sobrecargue estos ficheros con información repetida, pero tampoco provocar que la manipulación de los datos, de ser necesario, resulte en un cuello de botella.

- La primera de ellas es que la estructura fuera la de [$imagen_k$, $imagen_{k+1}$, ocr_k, ocr_{k+1}, $max(label_k, label_{k+1})$]. Con esta implementación se pretendía que cada entrada fuera compuesta por dos páginas, por lo que cada entrada de los ficheros sería directamente introducida al modelo.
- La segunda de ellas es que cada elemento tuviera la estructura [$imagen_k$, ocr_k, $label_k$]. Este enfoque nos obliga a acceder a la siguiente entrada para poder generar un ejemplo de entrada para nuestro modelo, aunque reducimos significativamente el número de ejemplos que podemos introducir por fichero.

Después de pruebas en tiempos se observó que la mejor opción era la segunda, ya que esta no supone un tiempo elevado de carga en memoria RAM y la unión de datos manual no resulta en un cuello de botella.

Debido a este estudio del tamaño de los ficheros y de su impacto en la memoria RAM, se ha decidido no generan un único fichero para la sección de train, sino generar múltiples ficheros de tamaño más reducido, lo que nos permite cargar los datos en memoria RAM sin problemas, siendo 3000 el número de páginas contenidas en cada fichero. Aunque debido a esta división se ha tenido que añadir al final de cada fichero H5DF el primer ejemplo del siguiente fichero, para conservar la continuidad sin necesidad de abrir 2 ficheros simultáneamente.

3.4 Selección de dataset para realizar un benchmark

Una vez ya tenemos el dataset principal que vamos a usar para medir nuestro rendimiento, necesitamos de otro dataset de uso común que se haya usado en otros proyectos de investigación que

nos permita obtener métricas de *accuracy, F1 Score* y *Kappa* y así por nuestra parte, podernos comparar con los resultados obtenidos de otras fuentes con el fin de saber si nuestros resultados han sido mejores o peores que otros trabajos previos.

Para poder realizar esta prueba en nuestro utilizaremos Tobacco800, que ya ha sido ampliamente utilizado como dataset de benchmark no solo en tareas como segmentación de documentos sino también en otras tareas como clasificación de documentos, por lo que es bastante conocido. Este dataset consta de 1290 páginas, las cuales conforman 742 documentos y en nuestro caso estaremos utilizando todo el dataset como únicamente un dataset de pruebas, en ningún momento se entrenará en este dataset. El motivo de esta decisión es cuando se ha estudiado los otros proyectos[14] se ha observado que en ninguno deja constancia de que división de entrenamiento, validación y test que se ha empleado, y no únicamente eso sino que como en uno de estos se estudia[14] la división de las clases afecta en gran medida a los resultados obtenidos, además de que hay riesgo de *data leak*, que significa que es hay páginas de documentos tan similares que en un caso donde dividimos el documento en las secciones de *train, validación* y *test* si dos páginas casi idénticas terminaran una en la sección de *train* y la otra en la sección de *test*, por ejemplo, nuestro modelo se podría ver beneficiado debido a esta similitud, debido a que hay diferentes páginas en este dataset que son prácticamente idénticas y se encuentran en diferentes conjuntos que deberían ser independientes, por lo que para evitar esta situación nosotros no segmentaremos en ningún momento el dataset ni entrenaremos en él.

3.5 Problemas en el balanceo de clases

3.5.1 Origen del problema

Una vez realizado seleccionado el dataset que vamos a utilizar como benchmark, procedemos a analizar este dataset para ver su distribución de clases y compararlo con la distribución de clases de la porción de entrenamiento de RVL-CDIP (BigTobacco), ya que como exclusivamente es la sección de entrenamiento la que influye en el entrenamiento de nuestros modelos es esta la única que puede jugar un papel destacable en como se desarrolla el aprendizaje de los modelos, puesto que nuestra intención es visualizar si hay algún desbalance de clases que pueda favorecer a nuestros modelos en frente de otros no porque nuestros modelos sean más precisos, y los resultados los podemos visualizar en la figura 3.

Lo que estamos visualizando es como en Tobacco las clases se encuentran balanceadas, por lo que no hay ningún problema, en cambio, cuando miramos BigTobacco nos encontramos con un serio problema, el cual es que cerca del 77% de las páginas pertenecen a un mismo documento, por lo que si nosotros generásemos un modelo que diera todo el rato como salida ceros, tendríamos un 77% de precisión en la sección de entrenamiento, aunque esto no sería un problema si las secciones de validación y test mantienen un balance correcto, ya que nos percataríamos de esto, por lo que a causa de esta situación procedemos a analizar los conjuntos de validación y de test de BigTobacco, por el hecho de que de presentarse el mismo desbalance, podríamos estar arriesgándonos a que nuestro modelo estuviera viéndose beneficiado de este factor a la hora de obtener métricas de *accuracy*, *F1 Score* y *Kappa*.

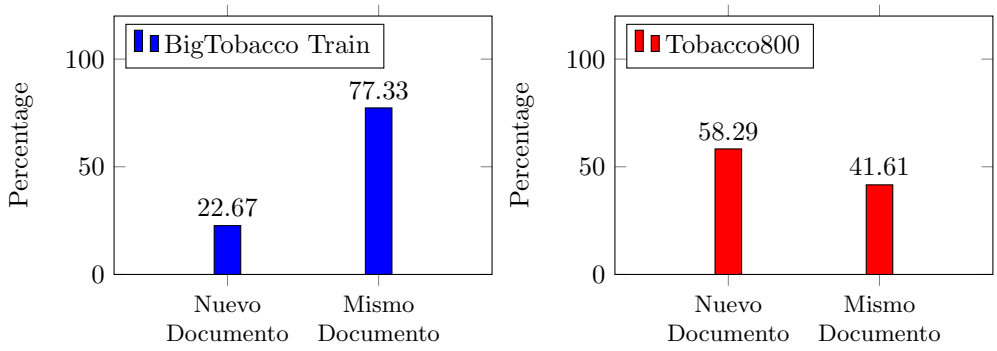


Figure 3: Distribuciones de clases en BigTobacco y en Tobacco800

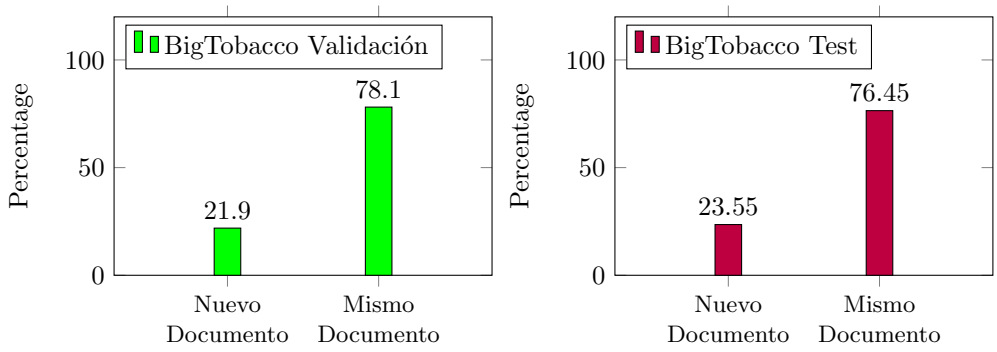


Figure 4: Distribuciones de clases en BigTobacco validación y Test

Como se puede observar en la figura 4, tanto la sección de validación como de test, presentan la misma distribución de clases que la sección de entrenamiento, lo cual es un problema porque existe la posibilidad de que nuestros entrenamientos estén siendo sesgados, por lo que en vista de esto, tenemos que tomar la decisión de aplicar un filtrado a nuestro dataset para corregir este desbalanceo de clases, aunque este filtrado solo se aplicará a la sección de entrenamiento, pues es de esta de la cual el modelo aprende y, por lo tanto, de la cual se puede ver influenciado

3.5.2 Filtrado de datos de entrenamiento

En primer lugar, para observar la situación en la que nos encontramos generamos una gráfica que nos agrupe los documentos en grupos, siendo estos grupos definidos por el número de páginas que contiene cada documento.

El resultado de esta visualización se puede observar en la figura 5, donde el extremo izquierdo hay el número de documentos que se conforman de una única página y a la derecha el número de documentos que se conforman de 1942 páginas, además debido al extremo desbalance en esta gráfica se ha tenido que escalar el eje que nos da información sobre el número de documentos a una

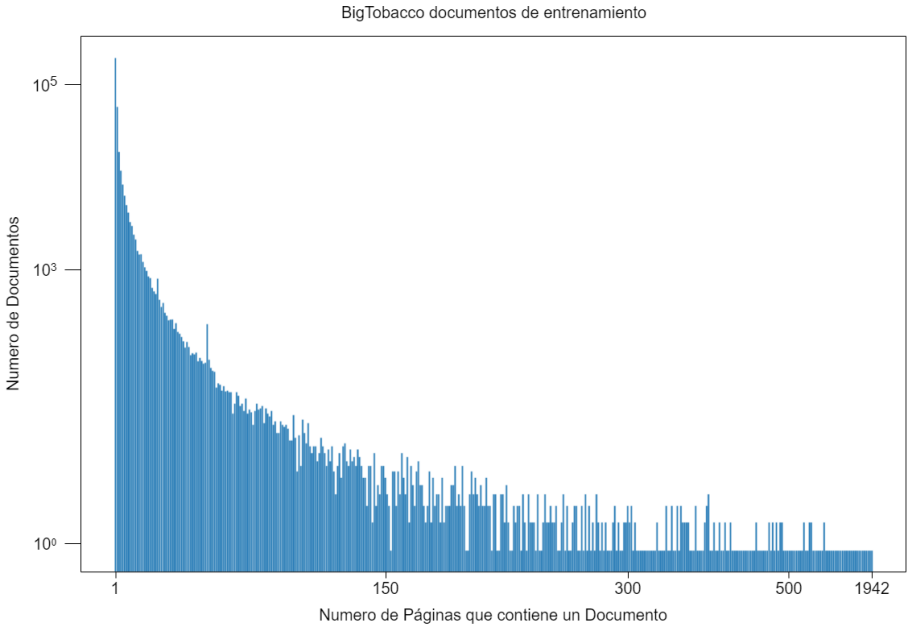


Figure 5: Numero de documentos segun el numero de páginas en la sección de BigTobacco Train.

escala logarítmica para que este sea visible.

Entonces una vez tenemos delante nuestra situación hay dos alternativas:

1. En primer lugar podemos acortar el número de documentos seleccionados de forma exponencial a medida que estos tienen más páginas (por lo que si hubiera 4 documentos de 2 páginas y 16 de 3 páginas solo seleccionaríamos aleatoriamente 2 documentos de 2 páginas y 4 documentos de 3 páginas), por lo que estaríamos acortando el número de ocurrencias de páginas en un mismo documento, y al mismo tiempo descartamos todos los documentos que contengan más de K páginas, ya que de no hacerlo debido a la increíble cantidad de documentos de 1 sola página se vuelve muy complejo encontrar un equilibrio. El resultado de esta aproximación se puede observar en la figura 6.
2. En segundo lugar podemos descartar todos los documentos que contengan más de K páginas, siendo K un número entre 1 y 1942, siendo esta la opción más sencilla. El resultado de aplicar este enfoque se puede ver en la figura 7.

Después de diversas pruebas encontramos que si escogíamos la primera opción, debíamos limitar el número de páginas por documento a 58, mientras que si escogíamos la segunda el número de páginas debía ser limitado a 10, ya que al no reducir el número de documentos hay muchas más ocurrencias de "Mismo documento".

Después de un análisis se decidió optar por la segunda opción, puesto que esta, a pesar de

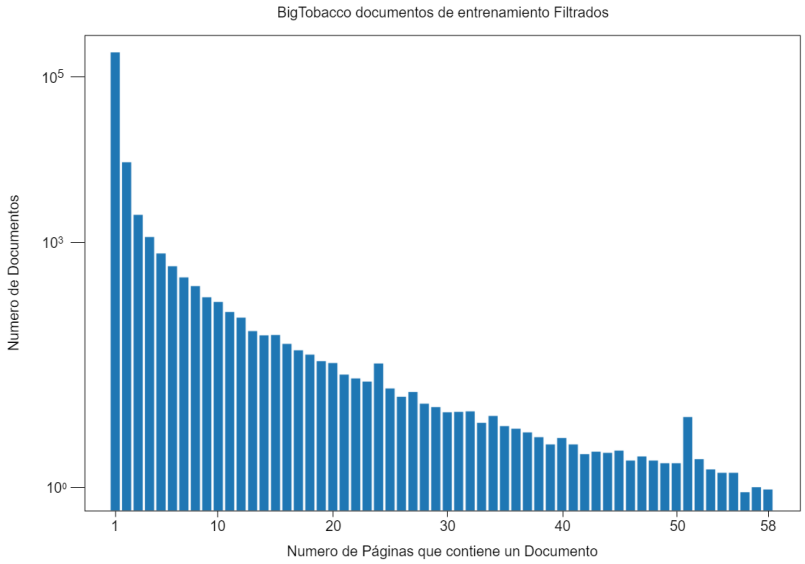


Figure 6: Filtrado de un máximo de 58 páginas y reducción progresiva de documentos.

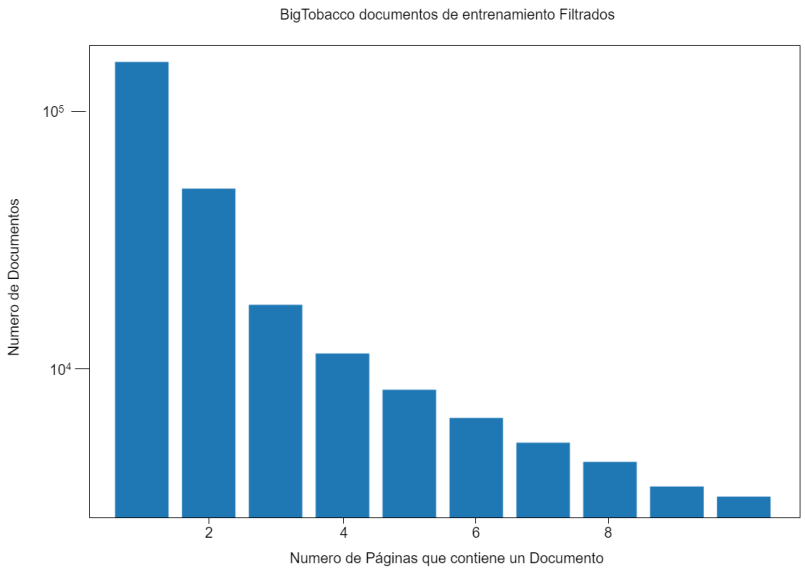


Figure 7: Descartados todos los documentos de más de 10 páginas

destruir todos los documentos de más de 10 páginas, conservaba el 92% de los documentos respecto el total, mientras que el primer método apenas conservaba el 70%, debido al notable desbalance en las ocurrencias de documentos con 1 única página. No obstante aun en caso de perder solo un 8% de los documentos respecto el total, la perdida en cantidad de páginas resulta del 59%.

4 Modelos

4.1 Decisión de Modelos para texto e imagen

Una vez ya tenemos todos los datos preparados, es momento de seleccionar la arquitectura de los modelos, para esto por nuestra parte optamos como decisiones base que el modelo que analizara el OCR fuera Distil BERT [20], ya que Distil BERT es un modelo que pertenece a la familia de los Transformers, que últimamente están demostrando un gran rendimiento tanto en imagen como texto [11][12] por lo que decidimos poner a prueba cuanto rendimiento podríamos obtener de ellos y que la parte que analizara la imagen fuera EfficientNet[21], ya que no solo ha mostrado un gran rendimiento en tareas relacionadas con imágenes, sino que es una arquitectura muy eficiente haciendo uso de la mínima cantidad necesaria de parámetros.

4.1.1 Modelo para imagen

EfficientNet es una familia de redes convolucionales la cual se basa en la máxima eficiencia reduciendo al máximo el tamaño del modelo, optimizando la velocidad de entrenamiento. Existen diferentes tipos de EfficientNet comenzando por la B0, que es la más pequeña y la menos potente, pero más veloz, hasta la B6 que es la más grande de todas y por ende la más potente, aunque la más lenta de entrenar. En todos los modelos que mencionemos que hagan empleo de EfficientNet, se hará empleo de la EfficientNetB0 en caso de que no se diga lo contrario.

Como se puede observar en la figura 8 la familia de redes EfficientNet toma por entrada una imagen de entrada, y solo una, la cual en nuestro caso es RGB, y la procesa por las diferentes capas convolucionales que conforman a la red. Finalmente la última capa convolucional es procesada por una capa Densa de 256 neuronas, habiendo sido este número decidido previo a los experimentos. Aunque al tratarse de una red optimizada en todo momento para conservar el *accuracy* y tener un tiempo necesario para entrenar mínimo, la red está optimizada para tratar con imágenes de 224x224 pixeles, por lo que en todo momento las imágenes son redimensionadas a esta medida. Por lo que finalmente obtenemos un tensor de 256 neuronas que contienen la información visual de la imagen.

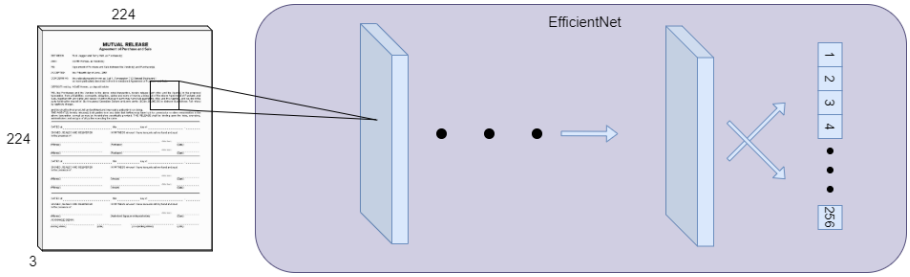


Figure 8: Esquema visual de EfficientNet

4.1.2 Modelo para texto

Distil BERT es una versión destilada de BERT[22]. Un modelo destilado se obtiene con el proceso de transferir conocimiento, también conocido como *distilling*[23], de un modelo, como es en nuestro caso BERT, a un modelo con una arquitectura más reducida. El modelo original, BERT, tendrá más capacidad de almacenar conocimiento, pero modelos como BERT que son muy extensos y que han sido pre-entrenados con diversas tareas, es posible que no se utilice todo su potencial dependiendo de la tarea, por lo que una versión reducida ya puede ser válida. Por lo que el proceso de destilación, permite transferir el conocimiento evitando, en algunos casos, que disminuya el *accuracy*. Además, al ser más pequeños estos se pueden implementar en hardware menos potente o bien combinar con otros modelos sin ocupar tanto espacio.

Por lo que usaremos Distil Bert para procesar las relaciones entre palabras de la página o páginas introducidas y nos dará información sobre el texto del documento.

Para lograr esta tarea, en primer lugar haremos uso de un Tokenizador. Un Tokenizador es una función, que dado una entrada en forma de texto, mapea a un identificador numérico. Este mapeado puede ser simple, como coger todas las palabras del diccionario, asignarles un identificador y así poder mapear cada palabra a un número o bien puede ser más complejo, donde lo que se mapean no son directamente palabras sino secciones de palabras. Por ejemplo la palabra, amigo y amable comparten la combinación "am" al principio, por lo que se podría hacer un mapeo de esta combinación a un identificador numérico, lo cual permite que no se deba de hacer un mapeo por cada palabra existente.

El Tokenizador seleccionado ha sido utilizado en el pre-entrenamiento de distil BERT, por lo que nos podemos aprovechar de esta herramienta para traducir las palabras de entrada que nosotros entendemos, a identificadores que nuestro modelo es capaz de entender.

En segundo lugar, debemos verificar la longitud de estas secuencias de números generados, ya que distil BERT fue originalmente entrenada para ser capaz de procesar frases de una longitud máxima de 512, siendo siempre la primera posición un token especial llamado [CLS]. Este token no proviene del texto y tiene como objetivo asimilar el contexto general de todo nuestro texto des de un marco global, además este token fue empleado en el pre-entrenamiento de la red, por lo que nos podemos beneficiar del hecho que la red ya tiene el conocimiento de como extraer el contexto general de la página.

En caso de exceder este límite, nuestro modelo podría seguir procesando la información, pero o bien no sería capaz de asimilarla o bien no estaríamos aprovechando al máximo la capacidad de nuestra red, ya que al cambiar la longitud presentada en su pre-entrenamiento estaríamos obligando al modelo a re-aprender la relación máxima entre las palabras, por lo que en caso de exceder este número nos quedaremos con las primeras 511 posiciones del texto y le añadiremos el token especial [CLS] tal y como se puede ver en la figura 11.

Distil BERT también es capaz de procesar 2 o más páginas de forma simultánea, pero en caso de introducir 2 páginas o más un token especial llamado [SEP] deberá ser introducido entre los tokens de cada página para indicar al modelo la existencia de uno o más textos independientes, así mismo, en caso de que se introduzcan 2 o más páginas, se limitará el número máximo de tokens

por página dependiendo del número de páginas existente, por lo que si tenemos dos páginas de entrada y tenemos un máximo de 512 tokens, restando el token [CLS] y [SEP] nos quedan 510 posiciones, por lo que se limitará a un máximo de 255 tokens por documento, y en caso de que queden posiciones vacías, ya bien porque se introduzca un documento de únicamente 4 palabras o bien porque al introducir 2 páginas una contenga 255 tokens y la siguiente 200 tokens, el resto de posiciones irán cubiertas por el último tipo de token especial llamado [PAD] el cual le indica a la red que en esa posición no hay información. Un ejemplo de esto se puede observar en la figura 10.

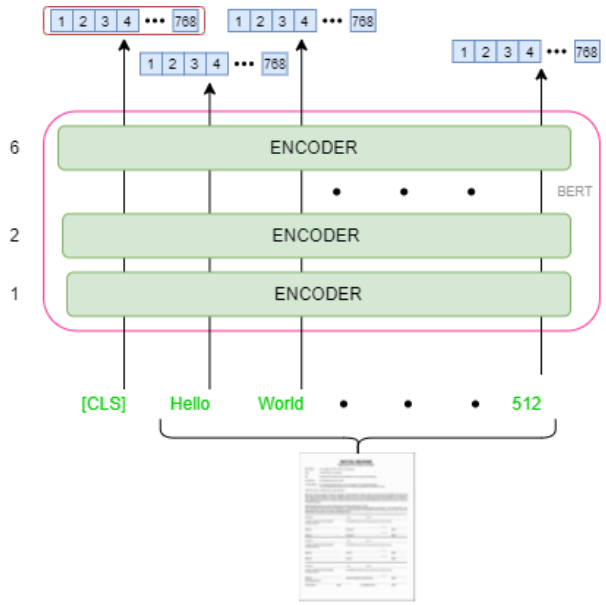


Figure 9: Modulo de texto haciendo uso de Distil BERT con 1 documento

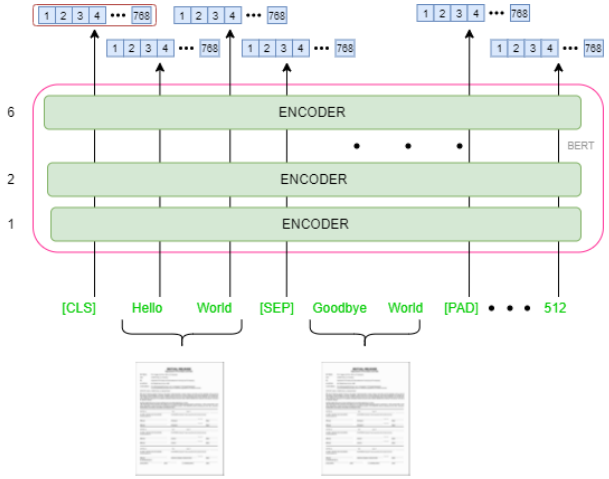


Figure 10: Modulo de texto haciendo uso de Distil BERT con 2 documentos

Una vez las páginas sean introducidas, ya sea en el caso de una sola página o bien de múltiples, se seleccionará la salida obtenida de la entrada [CLS] que como se ha mencionado contiene la información global del documento, lo que se podría interpretar como el contexto. Por lo que finalmente obtenemos un tensor de 768 posiciones que contiene el contexto general del texto.

4.2 Unión de modelos de texto e imagen

Una vez definidos los modelos solo nos queda hacer la unión de estos, no obstante debido a que estos pueden ser tratados como módulos independientes, siendo EfficientNet capaz de procesar una única página, y distil BERT capaz de procesar de 1 a N páginas, se toman tres decisiones:

1. Se plantearán modelos que hagan uso de dos páginas y tres páginas, ya que como se ha podido ver en el estudio del estado del arte para esta tarea, el uso de 3 páginas nos ofrece una ventaja adicional, y a causa de que para nosotros la diferencia entre 2 y 3 páginas es el número de llamadas a EfficientNet y distil BERT, la estructura del modelo no cambia.
2. Se probarán diferentes formas de utilizar distil BERT, puesto que en el caso de 2 páginas bien podemos o llamar a distil BERT una vez y concatenar los tokens a la entrada, o bien podemos pasar las páginas de forma independiente, obtener 2 tensores de 786 y concatenarlos con las llamadas de EfficientNet. Por lo que en el caso de 2 páginas se harán experimentos llamando a distil BERT una o dos veces, y en el caso de 3 páginas se llamará 1 vez, haciendo la unión de los 3 documentos, 2 veces, haciendo la unión de la manera (K_{n-1}, K_n) y (K_n, K_{n+1}) siendo K el documento y n la posición o bien haciendo 3 llamadas independientes.
3. En vista de los resultados de VGG16, haremos dos pruebas con una única llamada a distil BERT, es decir concatenando todos los tokens en la entrada, y haciendo uso de EfficientNetB2. El objetivo de estas pruebas es observar el impacto que tiene la red convolucional en el rendimiento global del modelo.

Por lo que ya con los modelos y el cómo hacer empleo de ellos solo resta hacer la unión de estos, para ello se concatenan los tensores obtenidos de todas las llamadas en un único tensor resultante, el cual es procesado por una última capa Densa formada por una neurona con una activación sigmoide. La activación sigmoide nos servirá para ser capaces de obtener como resultado en la salida un valor numérico que tendrá un rango des del 0 hasta el 1, siendo este interpretado como una probabilidad de que las dos páginas en cuestión formen parte de dos documentos diferentes.

Los modelos propuestos en esta sección, pueden ser visualizados en la tabla 1:

Experimentos		
Modelo Imagen	Páginas	Llamadas Distil BERT
EfficientNetB0	2	1
EfficientNetB0	2	2
EfficientNetB0	3	1
EfficientNetB0	3	2
EfficientNetB0	3	3
EfficientNetB2	2	1
EfficientNetB2	3	1

Table 1: Modelos propuestos en este proyecto.

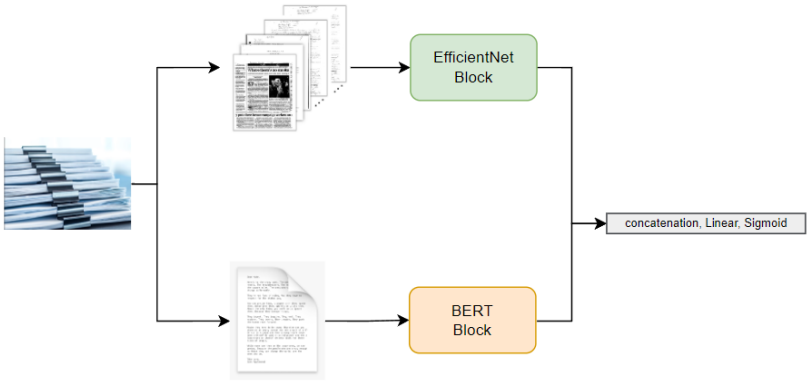


Figure 11: Modulo de texto e imagen unidos

4.3 Entrenamiento

Debido a la naturaleza de las redes neuronales, existe la posibilidad que dos entrenamientos diferentes nos proporcionen dos modelos con un rendimiento diferente, siendo estos entrenados con los mismos datos, por lo que se han realizado como mínimo 3 pruebas de cada experimento para poder obtener información sobre la variabilidad de nuestros resultados.

4.3.1 Hardware

Para el entrenamiento de los modelos el hardware utilizado ha sido el siguiente:

- 4xGPUs Tesla-V100
- 2 x IBM Power9 8335-GTH @ 2.4GHz
- 512GB de memoria RAM distribuida en 16 dimms x 32GB @ 2666MHz

- 2 x SSD 1.9TB as local storage

4.3.2 Hiperparámetros

Para entrenar a nuestro modelo se ha hecho uso de la función de *loss* llamada *Binary Cross Entropy* compara cada una de las probabilidades pronosticadas con el resultado real de la clase, que puede ser 0 o 1. Luego calcula la puntuación que penaliza las probabilidades en función de la distancia desde el valor esperado.

El optimizador usado ha sido *Adam*[24]. Al igual que el *learning rate* se puede entender como el tamaño del paso, el optimizador se puede entender como el cómo damos este paso. Adam es un algoritmo de optimización de reemplazo para el descenso de gradiente estocástico[25], que combina propiedades de otros dos algoritmos llamados AdaGrad[26] y RMSProp[27].

El número de *Epochs* se ha establecido en 3, el batch size utilizado ha sido de 9 y el número de workers para cargar nuestro dataset han sido 16, en cuanto al *Learning Rate* aplicado al modelo convolucional ha sido de $2 * 10^{-2}$ y a distil BERT se han aplicado dos técnicas llamadas *Learning Rate Adaptative y Slanded triangular Learning Rate*.

Las técnicas *learning rate adaptative y slanded triangular learning rate* se observaron cuando se hizo un estudio de distil BERT y de como mejorar su rendimiento, ya que se había observado que en Transformers es posible que un entrenamiento resulte en un modelo que no ha logrado aprender los conceptos deseados, pero que tampoco sea capaz de realizar tareas que ya había aprendido con anterioridad, este fenómeno es denominado *Catastrophic Forgetting* [28]. Por lo que para enfrentarnos a este problema aplicamos *Learning Rate Adaptative* [29], el cual consiste en que a cada capa se le establece un *Learning Rate* diferente, siendo menor las capas más profundas y mayor las más externas, para que el modelo sea capaz de aprender pero no olvidar. Por lo observado en nuestra investigación este error solo afecta a los Transformers, por lo que esta estrategia de *Learning Rate Adaptative* solo ha sido aplicada al módulo de texto.

Con *Learning Rate Adaptative* se combinó una técnica llamada WarmUp la cual consistía en incrementar el LR en las primeras Epochs, y después lentamente reducirlo, por lo que en las primeras fases el *Learning Rate* es muy bajo, por lo que el modelo aprende poco a poco, y a medida que las Epochs avanzan, este va aumentando para que el modelo empieza a aprender, seguida de una fase de disminución del Learning Rate en la que el modelo aprende los últimos detalles de los datos. Esta técnica es conocida con el nombre de Warmup[22] y aplicando *Slanded triangular Learning Rate* a nuestro LR, podemos lograr el mismo efecto.

Se realizó una prueba para visualizar la mejora causada por la estrategia del *Learning Rate* y *Slanded triangular Learning Rate* en distil BERT, la cual se puede visualizar en la tabla 2.

LLamadas a Distil BERT	Accuracy	
	Con ADLR y SLTR	Sin ADLR y SLTR
1	0.85	0.3759
2	0.77	0.54

Table 2: Experimentos con los primeros modelos propuestos aplicando y sin aplicar las técnicas mencionadas

4.4 Implementacion MultiGPU

4.4.1 Escalabilidad

En primer lugar se han hecho diversas pruebas de entrenamiento con una única GPU y progresivamente se han ido añadiendo GPUs, para ver como escalaban los modelos a medida que se añadían GPUs para determinar si merecía o no la pena aplicar este enfoque. Los resultados obtenidos son fruto de procesar el 10% de los datos de entrenamiento y haciendo uso de una única llamada a distil BERT con dos páginas. Los datos se pueden observar en la tabla 3.

GPUs	Tiempo cuando el número de llamadas a Distil BERT es Una
1	7h 2min
2	3h 8min
4	2h

Table 3: Tiempo de ejecución para el entrenamiento de un modelo

Por lo que los experimentos se han tenido que realizar todos haciendo empleo de 4 GPUs, debido a que el entorno en el que nos encontramos no es posible ejecutar una tarea por más de 48h seguidas. A causa de esta limitacion, se ha implementado el código necesario para un entrenamiento de una única GPU, pero no se ha podido utilizar.

4.4.2 Estrategias

Existen diferentes estrategias y enfoques a la hora de entrenar de forma paralela un modelo de inteligencia artificial, cada una con sus ventajas y desventajas. A continuación se presentan dos tecnicas muy conocidas, así como sus puntos fuertes y débiles:

1. *Data parallelism*: este método consiste en copiar el modelo en cada GPU, hacer el entrenamiento independiente en cada GPU, con datos independientes en cada GPU (ya que queremos que cada modelo de cada GPU vea datos diferentes), y al final de cada batch, después de calcular el error y, por lo tanto, actualizar los pesos, hacer una media de estos. Por lo que al final aumentamos la velocidad de entrenamiento, pero debido a la sincronización necesaria, al

hecho de que se hace una media de los pesos, y así como el factor de tener que copiar tantas veces el modelo como gráficas, puede provocar que sea más lento a la hora de entrenar de lo que consideraríamos ideal, por lo que el speed-up obtenido no siempre será el ideal.

2. *Model parallelism*: consiste en dividir el modelo entre diferentes GPUs, por ejemplo en un escenario donde solo tenemos 2 GPUs, la estrategia sería enviar el módulo de imagen a la GPU_1 y el modelo de texto a la GPU_2. El efecto de esto es que podemos aumentar substancialmente el batch, aunque el efecto más notable es que podemos crear modelos mucho más complejos y potentes que antes, ya que al dividir el modelo por diferentes GPUs, este puede crecer en tamaño.

En el caso actual utilizaremos la técnica de *data parallelism*, puesto que en nuestro caso nos interesa aumentar la velocidad de entrenamiento debido a las 48 horas límite que disponemos de entrenamiento.

4.4.3 Implementación

En cuanto a la implementación de este bucle de entrenamiento paralelo que hace uso de multiples GPUs, se han usado diversas librerías para facilitar la implementación del mismo. En primer lugar, se ha usado la librería de multiprocesado de Pytorch [30] la cual permite llamar a una función de forma paralela en diferentes hilos de ejecución independientes, y puesto que vamos a hacer uso de 4 GPUs se ha hecho uso de esta llamada indicando que genere 4 hilos independientes.

Seguidamente, se ha hecho empleo de la librería distributed dentro de Pytorch [31], la cual nos permite establecer el entorno, es decir un medio por el cual los diferentes hilos se comunicaran en todo momento para informar entre ellos del estado de entrenamiento y, por lo tanto, de llevar a cabo las acciones necesarias, así como el backend en el que se entrenaran las GPUs, lo que nos permite acceso a una serie de llamadas asíncronas las cuales permiten a los hilos independientes generar puntos de sincronización o de parada según el entrenamiento lo requiera. Junto a esta implementación se hace uso de una librería llamada DistributedDataParallel [32] la cual se usará para copiar el modelo a las distintas gráficas para que estas puedan entrenarlo de forma independiente.

Finalmente, para decidir con que datos se entrenarán los diferentes modelos en las diferentes gráficas lo que hemos hecho ha sido dividir los 470 ficheros con estructura h5df generados previamente en 4 segmentos independientes, por lo que los 4 modelos no entrenan en ningun momento con los mismos datos.

4.5 Recreación del modelo estado del arte

Para ser capaces de validar nuestros resultados debemos comparar estos con el estado actual del arte, siendo este VGG16 [15].

El único problema que encontramos es que por su parte ellos entrenan, validan y hacen un test en dos datasets diferentes, uno de ellos lo nombran AI.Lab.Splitter, al cual no tenemos acceso

porque no es público. Y el segundo es tobacco800, el problema de este es que tal y como ellos mismos demuestran el cómo se haga la división de train, validación y test tiene un gran impacto en los resultados obtenidos, y además ellos por su parte mencionan el porcentaje de validación respecto al de entrenamiento, pero no mencionan el porcentaje de test, por lo que no tenemos forma de imitar sus experimentos, por lo que decidimos implementar el mismo modelo que ellos proponen, con el mismo procedimiento sobre los datos para así por nuestra parte primero comprobar que los datos son de confianza y segundo poder hacer nuestras propias pruebas sobre BigTobacco para así tener un mejor contraste sobre nuestra mejora sobre VGG16.

Una vez creado el modelo de VGG16 lanzamos una pequeña prueba con VGG16 sobre Tobacco800 para primero confirmar el funcionamiento de la red, y segundo verificar resultados similares a los presentados en su paper. Los resultados son visibles en la tabla 4.

	10% train, 20%val, 70% test	30% train, 20%val, 50% test
VGG16	Acc 0.913, F1 0.944, Kappa 0.755	Acc 0.92, F1 0.87, Kappa 0.90

Table 4: Experimentos con dos separaciones diferentes haciendo uso de VGG16

Por lo que procedemos a implementar el pipeline completo para poder implementarlo con BigTobacco, aunque nos encontramos con dos factores destacables:

1. El modelo que ellos usan consta de una VGG16, que al igual que nuestros modelos extrae un tensor de dimensiones $(batch, n\_features)$ el cual conecta a una capa Densa. Pero ellos han congelado los pesos del modelo VGG16.
2. Ellos implementan un proceso a las imágenes llamado Otsu [33] el cual tiene como objetivo mejorar el rendimiento de la red aplicando un *threshold* a las imágenes para dividir el contenido y el fondo de las páginas, haciendo que todos los pixeles superiores al threshold opten por un valor de 255 y todos los inferiores por un valor de 0.

En vista de estos dos puntos por nuestra parte tomamos dos decisiones.

La primera siendo revertir el efecto de los pesos congelados en VGG16, ya que al entrenar con BigTobacco si VGG16 está con los pesos congelados no será capaz de aprender toda la información de un dataset tan grande.

La segunda es aplicar las técnicas usadas por ellos a nuestro pipeline para que el proceso sea lo más fiel posible, ya que nuestra intención es poder comparar lo mejor posible nuestros resultados.

Experimentos	
Modelo Imagen	Páginas
VGG16	2
VGG16	3

Table 5: Modelos que representan el estado del arte actual.

5 Resultados

En las tablas 6 y 7 se pueden visualizar los resultados obtenidos tanto en BigTobacco como en Tobacco800. Los resultados han sido obtenidos de seleccionar la Epoch con la mejor *accuracy* en la sección de validación, y seleccionar la *accuracy*, F1 Score y Kappa del resultado de test para esa misma Epoch. Además, debido a que estamos ejecutando los entrenamientos con 4 GPUs, y como ya se ha comentado por la naturaleza del mismo existe una posible inestabilidad, hemos ejecutado cada entrenamiento 3 veces y después hemos hecho la media de los resultados, para obtener un resultado representativo, además para representar la estabilidad de los resultados se ha calculado la desviación estándar de la media calculada.

			BigTobacco Dataset					
			Entrenamiento sin filtrar datos			Entrenamiento con datos filtrados		
Modelo Imagen	Paginas	Llamadas Distil BERT	Accuracy	F1 Score	Kappa	Accuracy	F1 Score	Kappa
EfficientNetB0	2	1	0.966 ± 0.027	0.924 ± 0.046	0.886 ± 0.068	0.956 ± 0.044	0.903 ± 0.108	0.881 ± 0.126
EfficientNetB0	2	2	0.944 ± 0.040	0.895 ± 0.076	0.846 ± 0.112	0.951 ± 0.047	0.889 ± 0.119	0.829 ± 0.190
EfficientNetB0	3	1	0.961 ± 0.021	$\mathbf{0.965 \pm 0.010}$	$\mathbf{0.943 \pm 0.016}$	0.940 ± 0.056	0.915 ± 0.056	0.874 ± 0.096
EfficientNetB0	3	2	$\mathbf{0.977 \pm 0.006}$	0.956 ± 0.006	0.929 ± 0.009	0.938 ± 0.056	0.902 ± 0.072	0.845 ± 0.111
EfficientNetB0	3	3	0.957 ± 0.013	0.927 ± 0.021	0.882 ± 0.032	0.913 ± 0.088	0.866 ± 0.109	0.786 ± 0.170
EfficientNetB2	2	1	0.972 ± 0.001	0.955 ± 0.001	0.904 ± 0.035	$\mathbf{0.987 \pm 0.001}$	$\mathbf{0.981 \pm 0.002}$	0.956 ± 0.003
EfficientNetB2	3	1	0.958 ± 0.003	0.929 ± 0.007	0.888 ± 0.005	0.974 ± 0.004	0.959 ± 0.014	0.920 ± 0.021
VGG16	2	0	0.940 ± 0.001	0.927 ± 0.025	0.883 ± 0.003	0.968 ± 0.006	0.938 ± 0.016	0.932 ± 0.015
VGG16	3	0	0.935 ± 0.002	0.934 ± 0.009	0.893 ± 0.014	0.982 ± 0.003	0.980 ± 0.002	$\mathbf{0.974 \pm 0.003}$

Table 6: Resultados experimentos de test en BigTobacco

			BigTobacco Dataset					
			Entrenamiento sin filtrar datos			Entrenamiento con datos filtrados		
Modelo Imagen	Paginas	Llamadas Distil BERT	Accuracy	F1 Score	Kappa	Accuracy	F1 Score	Kappa
EfficientNetB0	2	1	0.918 ± 0.005	0.948 ± 0.003	$\mathbf{0.757 \pm 0.011}$	0.909 ± 0.007	0.942 ± 0.005	$\mathbf{0.730 \pm 0.012}$
EfficientNetB0	2	2	0.880 ± 0.001	0.924 ± 0.002	0.644 ± 0.013	0.869 ± 0.002	0.916 ± 0.002	0.624 ± 0.003
EfficientNetB0	3	1	0.921 ± 0.007	0.953 ± 0.004	0.694 ± 0.023	0.909 ± 0.014	0.946 ± 0.008	0.641 ± 0.053
EfficientNetB0	3	2	$\mathbf{0.921 \pm 0.008}$	$\mathbf{0.953 \pm 0.005}$	0.696 ± 0.034	0.870 ± 0.032	0.921 ± 0.020	0.535 ± 0.092
EfficientNetB0	3	3	0.887 ± 0.007	0.935 ± 0.004	0.502 ± 0.035	0.862 ± 0.023	0.918 ± 0.013	0.462 ± 0.093
EfficientNetB2	2	1	0.908 ± 0.006	0.942 ± 0.004	0.717 ± 0.014	0.878 ± 0.014	0.922 ± 0.012	0.643 ± 0.011
EfficientNetB2	3	1	0.893 ± 0.030	0.935 ± 0.021	0.630 ± 0.052	$\mathbf{0.911 \pm 0.001}$	$\mathbf{0.948 \pm 0.001}$	0.655 ± 0.010
VGG16	2	0	0.910 ± 0.002	0.941 ± 0.001	0.744 ± 0.001	0.890 ± 0.015	0.927 ± 0.012	0.700 ± 0.020
VGG16	3	0	0.891 ± 0.004	0.935 ± 0.003	0.581 ± 0.010	0.892 ± 0.007	0.936 ± 0.004	0.580 ± 0.020

Table 7: Resultados experimentos de test en Tobacco800

5.1 Análisis de Resultados BigTobacco

Como se puede observar en la tabla 6 en todos los parámetros estudiados los valores máximos siempre los alcanzan diferentes modelos propuestos por nosotros, superando así el estado del arte que ha existido ahora siendo este VGG16. Además, nuestro modelo propuesto contiene por parte de distil BERT 66 millones de parámetros y por parte de EfficientB2 9.2 millones de parámetros, que comparándolo con VGG16, la cual tiene 130 millones de parámetros, podemos ver como hemos reducido en 54 millones el número de parámetros del modelo, reduciendo el espacio que ocupa este y teniendo margen para aumentar la potencia de EfficientNet.

Si observamos las tendencias de estos podremos visualizar como en nuestros modelos propuestos cuantas más llamadas a distil BERT, menor el *F1 Score* logrado, dejando claro que distil BERT se beneficia de una única llamada donde pueda tener acceso a toda la información textual, aunque este efecto es visible no perjudica en gran medida el rendimiento general del modelo. Lo observado también se aplica al *accuracy* y el *kappa* que como se puede observar este es él más variabilidad tiene de todos los parámetros.

Cuando analizamos los resultados de VGG16, podemos ver que esta obtiene unos resultados de *accuracy* muy elevados, pero unas puntuaciones en *F1 Score* más reducidas, excepto en el uso de 3 páginas con datos filtrados como se puede apreciar, por lo que es posible que esté sucediendo lo mismo que se planteaba en el ejemplo cuando se explicaba el concepto de *accuracy* y su problema principal para esta tarea, aunque se puede observar como *accuracy* y *F1 Score* siguen la misma tendencia en todos los modelos mostrados.

Por otro lado, comparamos los resultados con el dataset filtrado respecto el que no se ha filtrado se puede observar como tanto el *accuracy*, *F1 score* y *kappa* se ven reducidos en la sección donde el dataset ha sido filtrado, mostrando como los modelos estaban sacando ventaja del desbalance de clases mostrado en secciones anteriores y así obteniendo unos resultados mejores que los reales, aunque como se puede ver en la tabla hay casos donde no ha sido como se menciona y aun existiendo este factor la diferencia de resultados en algunas pruebas es mínima.

Por otra parte, si comparamos los modelos que hacen empleo de EfficientNetB2 con EfficientNetB0 para imagen podemos ver una clara subida de la precisión obtenida, esta se puede observar en el modelo entrenado con 3 páginas y 1 llamada a distil BERT cuando el dataset ha sido balanceado, lo que nos indica claramente que el papel de la convolucional sigue teniendo un rol fundamental aun cuando esta se combina con una red especializada en el análisis textual.

En cuanto a la estabilidad de los resultados, se puede observar como los modelos entrenados en datos filtrados, son en lo general más estables que los modelos entrenados en datos que no se han filtrado, aunque teniendo en cuenta la distribución de los datos cuando no se han filtrado, es bastante probable que el modelo se esté beneficiando de este debalance no solo en resultados, sino en estabilidad también. Aunque al igual que sucedía con los resultados obtenidos, la diferencia de estabilidad en los resultados no es muy pronunciada, aunque si visible.

Los mejores modelos obtenidos serían o bien EfficientNetB0 con 3 páginas y 1 llamada a distil BERT para los modelos entrenados con datos sin filtrar o EfficientNetB2 con 2 páginas y 1 llamada a distil BERT siendo este el mejor de los modelos obtenidos que se han entrenado con los datos

filtrados.

5.2 Análisis de Resultados Tobacco800

En la tabla 7 podemos observar como los resultados son bastante más incoherentes, esto se puede ver claramente cuando comparamos los modelos que hacen empleo de EfficientNetB0 y EfficientNetB2, 1 sola llamada a distil BERT y 2 páginas, obteniendo peores resultados con un modelo claramente más potente. Esto no es de extrañar debido a que como hemos mencionado previamente a pesar de utilizar todo al dataset como un test, y, por lo tanto, evitar el problema de que dependiendo de como dividiéramos el dataset los resultados se verían gravemente afectados, como ya se presenta en trabajos previos, sino que además tiene un problema de posible DataLeak, donde hay páginas que son tan similares que es posible que el modelo se beneficie o perjudique de esta similitud. Por lo que las tendencias mostradas en la tabla de resultados anterior, aquí se siguen viendo pero menos marcadas.

Aunque si analizamos nuestros resultados con la sección de datos filtrados y sin filtrar, en este caso podemos observar, nuevamente con excepciones, como los modelos entrenados con los datos sin filtrar son mejores que los datos filtrados, aunque viendo como el modelo VGG16 con datos filtrados obtiene resultados notablemente inferiores a los otros modelos en esta misma categoría, nos hace pensar que los resultados obtenidos por parte del proyecto de investigación que actualmente se encuentra en el estado del arte, se ha visto influenciado por características como *DataLeak* y el efecto de particionar el dataset, que tiene Tobacco800.

Por otra parte, al igual que en la tabla anterior, el parámetro Kappa es el que presenta una mayor variabilidad, obteniendo resultados que van des de un 50% a resultados que alcanzan el 75%, aunque estos son notablemente menores que los observados en BigTobacco.

Si nos centramos en la estabilidad de los resultados, es aquí donde vemos que todos los resultados, independientemente de si se han entrenado o no con datos filtrados, muestran una desviación muy leve, lo que no nos permite poder comparar los resultados de forma más directa, aunque teniendo en cuenta los problemas que este dataset presenta.

Si quisiéramos seleccionar los mejores modelos obtenidos estos serían EfficientNetB0, con 3 páginas y 2 llamadas para los modelos entrenados con datos sin filtrar y EfficientNetB0, con 3 páginas y 1 llamada para los modelos entrenados con los datos filtrados, ya que si bien hay modelos que alcanzan un *F1 Score* más elevado como ya hemos explicado nos basamos en F1 Score como métrica más importante.

6 Gestión del proyecto

6.1 Obstáculos y riesgos previstos

Por supuesto al igual que en todo proyecto existen riesgos y posibles obstáculos que uno se puede encontrar por el camino, y en este caso no es diferente, por el hecho de que al haber muchos factores en juego no es muy difícil que haya algún contratiempo en algún momento del proyecto, a continuación menciono los principales obstáculos y riesgos que más potencial tienen de causar algún problema.

1. Teniendo en cuenta que ya ha habido otros equipos que han intentado una aproximación a este problema, probablemente la obtención de los datos no sea un problema, no obstante eso no niega el hecho de que de forma muy segura los datos no estén preparados para ser introducidos directamente a la red neuronal, por lo que seguramente sea necesario hacer un pre procesado intensivo a los datos.

2. Una vez tengamos todos los datos y el modelo, estos deben de ser introducidos en la memoria de la GPU que usemos, la cual es limitada, por lo que en caso de que los modelos sean demasiado complejos, se tendrán que buscar alternativas más simples, aunque esta simplificación si no es llevada a cabo correctamente, podría dejarnos en una situación donde el modelo no es capaz de aprender la tarea que tiene que llevar a cabo.

3. Aunque el modelo y los datos sean escogidos correctamente, es probable que otros factores como el propio proceso de entrenamiento del modelo acabe en fracaso, no por la implementación por mi parte, sino por la propia naturaleza del ámbito en el que nos encontramos.

6.2 Seguimiento realizado del proyecto

Para la realización de este proyecto, debido a que las pruebas pueden llevar diversas horas o incluso días, se tendrá que optar por un método ágil y flexible, donde pueda añadir cambios, comprobarlos y decidir si utilizo o no estos cambios.

Por lo que la intención es concretar una reunión semanal con el director del proyecto de forma semanal para poder tomar decisiones rápidamente en cualquier situación con el mínimo tiempo de latencia posible.

6.3 Herramientas de control empleadas

Para facilitar el seguimiento de las tareas se hará uso de Slack [34]. Se trata de una herramienta online, donde se pueden enviar mensajes de manera instantánea para mantener el contacto en todo momento, así como para hacer un control de las tareas pendientes y los resultados obtenidos.

En cuanto al control de versiones se hará empleo de la plataforma Github [35], ya que esta permitirá llevar un control del código y tener una copia de seguridad en todo momento para evitar la perdida de información.

Y finalmente para planificar el tiempo necesario a emplear en cada tarea haré uso de Gantter [36], porque gracias al uso de un Gantt se podrá tener una primera idea de la panificación temporal del proyecto.

6.4 Planificación temporal

A continuación se mostrará la planificación temporal elaborada teniendo en cuenta el tiempo disponible y las tareas a realizar.

El proyecto empieza el 13 de septiembre de 2021 y la finalización está prevista para el 27 de enero de 2022. Por lo que el estimado de días del proyecto es de 133 días con un total de 6h diarias dedicadas tendríamos 798 horas de dedicación totales.

6.4.1 Descripción de tareas.

A continuación se describen las tareas a realizar y en que consiste cada una de ellas. Aunque para un mejor entendimiento y organización del documento las tareas han sido distribuidas en grupos para así poder organizarlas fácilmente.

(TP) Trabajo Previo

En esta sección se encuentran las tareas que he ejecutado previas al inicio de GEP y del TFG como tal, las cuales, pese a ser superficiales, son una primera aproximación al problema.

1. TP.0 Investigación estado del arte: En este apartado he explorado las últimas aplicaciones hasta la fecha con el objetivo de tener un primer conocimiento de por donde empezar a elaborar mi plan de ruta y también para saber si la tarea es en primer lugar posible.
2. TP.1 Preparación entorno de trabajo: En este campo concretamente donde cada día hay novedades las librerías se ven constantemente actualizadas día a día, y, por lo tanto, cada día se generan nuevas incompatibilidades lo cual puede ser bastante problemático.

(GP) Gestión del proyecto

Esta sección está dedicada a tareas más relacionadas con la gestión del proyecto, ya que esta gestión es necesaria si quiero llevar el proyecto al día y actualizado. Por lo que como se puede intuir, gran parte de GEP está contenido dentro de esta sección.

1. GP.0 Reuniones: Como es habitual en estos proyectos para el correcto seguimiento de la evolución del trabajo, se concertarán reuniones semanales con el director del TFG para que este tenga siempre la información disponible en todo momento y también para corregir el rumbo en caso de que fuese necesario.
2. GP.1 Documentación: Al igual que las reuniones se irán produciendo hasta la finalización de este proyecto lo mismo ocurre con la documentación la cual se irá actualizando semanalmente para así tener los documentos actualizados en todo momento.
3. GP.2 Alcance: Se estudia el alcance que tendrá el proyecto, es decir que objetivos tendrá, que se pretende hacer y los medios necesarios. Por supuesto para esto se tiene que haber investigado sobre el estado del arte para tener un punto de partida.
4. GP.3 Planificación: Como en todo proyecto la planificación es esencial para que este sea organizado y estructurado, por lo que este tramo será invertido en la estructuración del mismo.
5. GP.4 Presupuesto: Quizás uno de los aspectos más importantes en el día a día de todo proyecto, en este intervalo de tiempo se estudiará el presupuesto necesario para llevar a cabo este proyecto.
6. GP.5 Sostenibilidad: Quizás uno de los apartados más ignorados, pero más relevantes de cara a un futuro cada vez más inmediato aquí se estudiará el impacto medioambiental que este proyecto puede tener.
7. GP.6 Presentación: En estos días se preparará una presentación oral para que un jurado gestionado por la FIB pueda evaluar el trabajo hecho por el alumno.

(PT) Preparación del dataset

Esta sección se centra en la preparación del dataset o conjunto de datos, ya que si bien la construcción del modelo es una etapa crucial, puesto que es el momento en el que estamos construyendo el núcleo del proyecto, la construcción del dataset es bien igual o más importante.

1. PT.0 Analisis de la información: Aquí se estudiarán los datos de los datasets disponibles de forma pública para, basándonos en la información que cada dataset nos provea, decidir cuál parece ser más prometedor y, por lo tanto, cuál deberíamos usar.
2. PT.1 Prueba Mejora calidad de las imágenes: En esta sección se estudiará si es posible hacer un pre procesado a las imágenes con el objetivo de que estas se vean con mejor calidad y mayor detalle.
3. PT.2 Creación OCR: Se crea el OCR (o texto) para cada imagen en caso de que no se encuentre disponible.
4. PT.3 Prueba ficheros H5DF en rendimiento y memoria: Pruebas sobre la mejor manera de estructurar los datos dentro de estos ficheros H5DF.

5. PT.4 Creación ficheros H5DF: Generamos los ficheros H5DF, los cuales los emplearemos para cargar los datos durante el entrenamiento (se usa este tipo de formato, ya que son altamente eficientes).

(CR) Creación del modelo

En esta sección se analiza los pasos en la creación del modelo que entrenaremos.

1. CR.0 Elección de modelo para imagen: Análisis y búsqueda de las arquitecturas más utilizadas para este tipo de problemas en relación con la imagen.
2. CR.1 Elección de modelo para texto: Análisis y búsqueda de las arquitecturas más empleadas para este tipo de problemas en relación con el texto.
3. CR.2 Estudio arquitectura del modelo: Se estudia como unir los modelos en un único modelo optimizando lo máximo posible el tamaño de la arquitectura y la *accuracy*.
4. CR.3 Unión de modelos y primeros entrenamientos de prueba: Una vez unidos los modelos se entrena el conjunto para ver los resultados.

(MGPU) Implementación Multi GPU

En esta sección se implementará una versión paralelizada del modelo.

1. MGPU.0 Adaptación del bucle de entrenamiento: Adaptar el bucle de entrenamiento para que se pueda hacer de forma paralela
2. MGPU.1 Entrenamiento: Entrenamiento del modelo y visualización de resultados.

(IMA) Implementacion pipeline y modelo Estado del Arte

En esta sección se implementará una versión paralelizada del modelo que representa el estado del arte actual para poder comparar los resultados obtenidos de este modelo con nuestras propuestas.

1. AD.0 Adaptación del problema: Aplicar los cambios necesarios para aplicar el modelo y el pipeline.
2. AD.1 Modificación del modelo: Crear el modelo propuesto en el estado del arte.
3. AD.2 Entrenamiento: Entrenar el modelo para ver resultados.

6.4.2 Recursos

6.4.3 Recursos Humanos

En este proyecto se encuentran se pueden encontrar 3 roles diferentes, el de jefe de proyecto, investigador y programador, las tareas de cada uno de estos es la siguiente:

1. Jefe de proyecto: Encargado de dirigir el rumbo del proyecto en todo momento y de controlar la calidad de este, así como las fechas establecidas (Jf).
2. Investigador: Encargado de obtener información sobre el estado del arte y todo lo relevante a las últimas técnicas usadas en los últimos años para poder aplicarlas en este proyecto (In).
3. Programador: Encargado de programar toda la parte de código con tal de que todo funcione correctamente (Pr).

6.4.4 Recursos materiales

A continuación se listan los recursos que serán necesarios en cada etapa, siendo los recursos disponibles y necesarios los siguientes:

- Ordenador de altas prestaciones con las siguientes prestaciones:
  - 4xGPUs Tesla-V100
  - 2 x IBM Power9 8335-GTH @ 2.4GHz
  - 512GB de memoria RAM distribuida en 16 dimms x 32GB @ 2666MHz
  - 2 x SSD 1.9TB as local storage

6.4.5 Gestión de riesgos

Al tratarse de una implementación que hasta día de hoy no se ha hecho, es posible que surjan diversos errores durante el desarrollo del proyecto, a continuación se describen los posibles obstáculos que podría ir enfrentando a lo largo del desarrollo y sus soluciones.

1. Incompatibilidad de librerías: Si bien es cierto que parte del trabajo previo ha sido la configuración del entorno, es posible que en algún momento del desarrollo necesite de alguna otra herramienta, la cual sea incompatible. En caso de que esto ocurra se deberán no solo actualizar las librerías, sino entrenar los modelos nuevamente, ya que es posible que debido al cambio de versión haya cambios sutiles que no podamos visualizar sin una búsqueda exhaustiva.
2. Entrenamientos fallidos: Es posible que debido al ámbito en el que nos estamos moviendo algunos entrenamientos resulten exitosos, mientras que otros fracasen, por lo que de ocurrir este error se debería de volver a entrenar a la red para confirmar si esta es nuestra situación.

3. Arquitectura o hyperparametros erróneos: Es perfectamente posible que bien debido a la arquitectura o a los hyperparametros el entrenamiento fracase, este hecho no se podrá confirmar hasta el momento en el que construyamos la arquitectura, por lo que en caso de que ocurra se deberán buscar rápidamente una solución.

4. Dataset con errores: Si el dataset escogido contiene errores no detectados inicialmente, como un mal balanceo de ejemplos o cualquier otro detalle que pase desapercibido se deberá o bien corregir estos errores o bien cambiar de dataset.

ID	Tarea	Tiempo (horas)	Dependencia	Recursos	Roles
TP	**Trabajo Previo**	**177**			
TP.0	Investigación estado del arte	33		Portátil	In
TP.1	Preparación entorno del trabajo	141		Portátil	Pr
GP	**Gestión de Proyectos**	**798**			
GP.0	Reuniones	775		Portátil	Jf,In,Pr
GP.1	Documentación	775		Portátil	In
GP.2	Alcance	37	TP.0	Portátil	In
GP.3	Planificación	40	GP.2	Portátil, Grannter [36]	Jf
GP.4	Presupuesto	19	GP.3	Portátil	Jf
GP.5	Sostenibilidad	15	GP.4	Portátil	Jf
GP.6	Presentación	21	GP.1	Portátil	Jf
PT	**Preparación dataset**	**228**			
PT.0	Analisis de la información	30	TP.1	Portátil	In
PT.1	Pruebas Mejora Calidad de Imágenes	30	PT.0	1xGPU Tesla-V100 + 1 x IBM Power9 8335-GTH @ 2.4GHz	Pr
PT.2	Creación OCR	60	PT.1	1 x IBM Power9 8335-GTH @ 2.4GHz	Pr
PT.3	Prueba ficheros H5DF en rendimiento y memoria	48	PT.2	1 x IBM Power9 8335-GTH @ 2.4GHz	Pr
PT.4	Creación ficheros H5DF	60	PT.3	1 x IBM Power9 8335-GTH @ 2.4GHz	Pr
CR	**Creación del modelo**	**132**			
CR.0	Elección de modelo para imagen	30	PT.4	Portátil	In
CR.1	Elección de modelo para texto	30	CR.0	Portátil	In
CR.2	Estudio arquitectura del modelo	36	CR.1	Portátil	In
CR.3	Unión de modelos y primeros entrenamientos de prueba	36	CR.2	1xGPU Tesla-V100 + 1 x IBM Power9 8335-GTH @ 2.4GHz	Pr
MGPU	**Implementación Multi GPU**	**72**			
MGPU.0	Adaptación del bucle de entrenamiento	36	CR.3	Portátil	In
MGPU.1	Entrenamiento	36	MGPU.0	4xGPU Tesla-V100 + 2 x IBM Power9 8335-GTH @ 2.4GHz	Pr
IMA	**Implementacion pipeline y modelo Estado del Arte**	**138**			
IMA.0	Adaptación del problema	72	MGPU.1	Portátil	In
IMA.1	Modificación del modelo	30	IMA.0	Portátil	In
IMA.2	Entrenamiento	36	IMA.1	1xGPU Tesla-V100 + 1 x IBM Power9 8335-GTH @ 2.4GHz	Pr
Total		**798**			

Table 8: Tareas con sus tiempos y cargos encargados de cada tarea.

6.5 Gestión económica

6.5.1 Presupuesto

Coste de Personal

Basándonos en la planificación se definen los costes del personal, teniendo en cuenta los roles designados en anteriormente en el apartado de recursos humanos, siendo estos: jefe de proyecto, investigador y programador. En la tabla 9 se pueden observar los costes por hora según las observaciones hechas en la empresa de reclutamiento *Hays*[37].

Rol	Coste por hora
Jefe de equipo	30 €
Investigador	20 €
Programador	16 €

Table 9: Costes de personal dependiendo de su rol.

En la siguiente sección se detallan los costes de cada etapa según los roles implicados en cada etapa, así como el coste total del proyecto el cual se estima multiplicando el coste por 1,3. En total el coste del proyecto es de 33.872 €.

ID	Tarea	Tiempo (horas)	Coste	Coste + SS	Roles
TP	**Trabajo Previo**	**177**	**2.916€**	**3.791€**	
TP.0	Investigación estado del arte	33	660€	858€	In
TP.1	Preparación entorno del trabajo	141	2.256€	2.933€	Pr
GP	**Gestión de Proyectos**	**798**	**6.090€**	**7.917€**	
GP.0	Reuniones	30	660€	858€	Jf,In,Pr
GP.1	Documentación	70	1470€	1911€	Jf, In
GP.2	Alcance	37	1110€	1443€	Jf
GP.3	Planificación	40	1200€	1560€	Jf
GP.4	Presupuesto	19	570€	741€	Jf
GP.5	Sostenibilidad	15	450€	585€	Jf
GP.6	Presentación	21	630€	819€	Jf
PT	**Preparación dataset**	**228**	**3.768€**	**4.898€**	
PT.0	Analisis de la información	30	600€	780€	In
PT.1	Pruebas Mejora Calidad de Imágenes	30	480€	624€	Pr
PT.2	Creación OCR	60	960€	1.248€	Pr
PT.3	Prueba ficheros H5DF en rendimiento y memoria	48	768€	998€	Pr
PT.4	Creación ficheros H5DF	60	960€	1.248€	Pr
CR	**Creación del modelo**	**132**	**2.496€**	**3.245€**	
CR.0	Elección de modelo para imagen	30	600€	780€	In
CR.1	Elección de modelo para texto	30	600€	780€	In
CR.2	Estudio arquitectura del modelo	36	720€	936€	In
CR.3	Unión de modelos y primeros entrenamientos de prueba	36	576€	749€	Pr
MGPU	**Implementación Multi GPU**	**72**	**1.296€**	**1.685€**	
MGPU.0	Adaptación del bucle de entrenamiento	36	720€	936€	In
MGPU.1	Entrenamiento	36	576€	749€	Pr
IMA	**Implementacion pipeline y modelo Estado del Arte**	**138**	**2.616€**	**3.401€**	
IMA.0	Adaptación del problema	72	1.440€	1.872€	In
IMA.1	Modificación del modelo	30	600€	780€	In
IMA.2	Entrenamiento	36	576€	749€	Pr
Total			**19.182€**	**24.937€**	

Table 10: Coste asociado a cada etapa del proyecto según el personal encargado

Costes genéricos

Debido a la situación actual con Covid, actualmente el trabajo se está ejerciendo de forma telemática, con posibilidad de ir presencialmente de lunes a viernes a las oficinas. Como ambos espacios se encuentran en Barcelona se ha hecho una estimación del coste del espacio de *coworking* siendo este un total 220€, el cual incluye electricidad, agua, una mesa individual y acceso todos los días de la semana. Por lo que a los 5 meses de finalizar el proyecto serían 1100€.

El software utilizado no tiene ningún coste aplicado al ser este público, así como el dataset el cual también es de dominio público y sin coste.

En la tabla 10 se presentan los costes en cada trámite según el personal encargado de cada tarea, de esta manera como su coste aplicando la seguridad social.

Además del coste de los trabajadores por cada tramo en el proyecto también se debe tener en cuenta el coste del hardware usado, ya que este será necesario para llevar a cabo la tarea. Así mismo se ha calculado la amortización de cada pieza de hardware haciendo un estimado de la esperanza de vida de cada uno, teniendo en cuenta que al año hay 220 días hábiles de 8 horas diarias de trabajo, siendo $Amortización = \frac{Horas\_de\_uso * Coste\_dispositivo}{Vida\_útil * días\_habiles * horas\_laborales}$ la fórmula utilizada para saber la amortización. En la tabla 11 se pueden observar el precio de los materiales necesarios, las unidades, una estimación de la vida útil de cada uno y las horas de uso que se indican en la planificación laboral.

Hardware	Precio	Unidades	Vida útil	Horas	Amortización
Portatil	700 €	1	4 años	530h	52€
Ordenador de altas prestaciones	29.889 € [1]	1	4 años	504h	2139€

Table 11: Presupuesto del hardware utilizado.

Contingencia

Como en todo proyecto, siempre puede haber imprevistos u obstáculos por el camino que dificulten la completación de una tarea y, por lo tanto, un gasto inesperado, por lo que es aconsejable hacer un plan de contingencia dejando un margen por si surgiera cualquier imprevisto.

Como la probabilidad de encontrar algún problema es alta, estableceré un 20% de margen por si surgiera algún imprevisto. En la tabla 12 se puede observar el total de contingencia para cada tipo de gasto.

[1] Para obtener el coste del Ordenador en altas prestaciones me he basado en el servicio de AWS[38] donde he escogido un equipo con las mismas prestaciones necesarias que el usado para hacer este proyecto.

Tipo	Coste	Contingencia
Espacio	1.100€	220€
Hardware	30.589€	6.117€
Personal	24.937€	4.987€
Total	56.626€	11.324€

Table 12: Contingencia del 20% por cada tipo de gasto.

Imprevistos

A continuación se mencionan los imprevistos que podría haber durante el desarrollo del proyecto, los cuales ya se comentaron en la planificación temporal. En esta sección se estimará su coste en términos de dinero y tiempo para cada uno de ellos. En la tabla 13 se puede observar con detalle las probabilidades y costos asociados a cada riesgo.

1. Incompatibilidad de librerías: En caso de encontrarnos en esta situación, se deberán volver a ejecutar, solo en los casos más extremos, pero no imposibles, todos los experimentos para así confirmar que el cambio de versiones de librería no afecta al rendimiento de los modelos, por lo que se deberán añadir 72 horas de entrenamiento, los costes de hardware serían 172€ y los costes del programador serian 384€. El riesgo de este caso es del 40%, ya que pese a que hay mucha gente que hace uso de estas librerías al ser actualizadas debido a los avances diarios siempre pueden surgir imprevistos
2. Entrenamientos fallidos: En este caso se deberán volver a hacer como mínimo un entrenamiento más, para confirmar si se trata de la arquitectura del modelo o de un fallo en el proceso de entrenar el propio modelo. Los costes en horas serian de 48h, los costes de hardware serian 115 € y los costes en personal serían 256 €. El riesgo de este caso es del 10% porque si bien puede ocurrir no suele ser el caso y se puede detectar rápidamente.
3. Arquitectura o hyperparametros erróneos: Este caso sería el mismo que el anterior, pero esta vez sería a causa del propio programador, los costes en tiempo y dinero serian los mismos. El riesgo en este caso es del 30%, puesto que como se está aproximando el problema de una forma donde se combinan redes que nunca antes se ha probado es posible que surjan complicaciones en el proceso.
4. Dataset con errores: En este caso se debería de contribuir 32h de revisión del dataset y si este no fuera viable debido a su gran cantidad de errores se deberían invertir 100h en volver a buscar y tratar un dataset lo más rápidamente posible. Por lo que teniendo en cuenta que estas tareas son llevadas a cabo por un programador implicaría un coste de 512 € en el primer caso donde el dataset fuera utilizable y un coste de 1.600€ en caso de que se tuviera que volver a hacer todo el procedimiento. El riesgo de este caso es del 5% porque la mayoría de datasets públicos son conocidos y ya han sido verificados por la comunidad.
5. Fallo en el ordenador de altas prestaciones: En este equipo se dispone de una copia de seguridad por lo que el fallo de este dispositivo supondría 5h de tiempo perdido para restaurar el

entorno y un coste de personal de 80€. El riesgo de este caso es del 2% porque al tratarse de un equipo de altas prestaciones este ha pasado unas pruebas muy rigurosas de calidad.

6. Fallo del portátil: En el caso del portátil no se dispone de datos relevantes, pero el reemplazo de este supondría una perdida de tiempo de aproximadamente 48h y un coste de 500€. El riesgo de este caso es del 5%.

Imprevisto	Coste	Riesgo	Coste total
Incompatibilidad de librerías	556€	40%	222.4€
Entrenamientos fallidos	371€	10%	37.1€
Arquitectura o hyperparametros erróneos	371€	10%	37.1€
Dataset con errores	512€	5%	25.6€
Fallo en el ordenador de altas prestaciones	80€	2%	1.6€
Fallo del portátil	500€	5%	25€
Total	2.390€		348,8€

Table 13: Riesgos para los posibles imprevistos y costes asociados

Coste Total

Con todo lo que hemos visto el coste total del proyecto ascendería a 94.336€, en la tabla 14 se pueden observar los detalles de cada coste.

Tipo	Coste
Personal	24.937€
Espacio	1.100€
Hardware	56.626€
Contingencia	11.324€
Imprevistos	349€
Total	94.336€

Table 14: Presupuesto final del proyecto

6.5.2 Control de gestión

Una vez definido el presupuesto se definen mecanismos de control para controlar que no haya perdidas en tiempos o dinero perdido al final del proyecto. De esta forma se podrá encontrar rápidamente cualquier error.

1. Desviación coste personal por tarea: (coste estimado coste real) horas reales
2. Desviación realización tareas: (horas estimadas horas reales) coste real
3. Desviación total en la realización de tareas: coste estimado total coste real total
4. Desviación total de recursos (software, hardware, espacio o personal): coste estimado total coste real total
5. Desviación total coste de imprevistos: coste estimado imprevistos coste real imprevistos
6. Desviación total de horas: horas estimadas horas reales

6.5.3 Imprevistos encontrados

Los imprevistos que nos hemos visto obligados a resolver y que han afectado a nuestra planificación inicial ha sido la necesidad de volver a recrear el dataset debido a los fallos encontrados en este y el re-entrenamiento de los modelos en este dataset, aunque esta situación ya se ha tenido en cuenta en el apartado de imprevistos y ya se le ha asignado un presupuesto, por lo que la planificación general no se ha visto afectada.

6.6 Sostenibilidad

A continuación se hace un estudio sobre la sostenibilidad en tres diferentes apartados, siendo estos: económico, ambiental y social. Aunque previamente se hará una autoevaluación sobre los conocimientos del autor de este trabajo sobre esta área.

6.6.1 Autoevaluación

A lo largo del Grado en ingeniería informática muchas veces nos han no solo recordado y recalcado, sino también enseñado como de importante es la sostenibilidad de un proyecto y como de ineficientes somos hoy en día en cuanto a gestión de recursos, ya que debido a esta cultura que tenemos de "tirar y usar" generamos una gran cantidad de residuos de los cuales la mayoría ni somos conscientes, pese a que somos nosotros quienes generamos gran parte de estos.

Por mi parte, des de un punto de vista personal siempre he pensado que la sostenibilidad de un proyecto es relevante, quiero decir si se va a hacer algo mejor hacerlo bien des del principio no? Aunque por supuesto es más fácil decirlo que hacerlo, ya que si bien pueden parecer palabras muy fáciles de utilizar nunca he creído seriamente en estos términos y lo que implican.

Por lo que mientras hacia la encuesta proporcionada por los profesores me he dado cuenta de la poca importancia, si no inexistente, que le daba a la sostenibilidad social y económica, y no solo eso, sino también el poco conocimiento que tenía sobre estas áreas, considerando siempre en la ambiental, la cual suele ser la más conocida de todas, pese a que el apartado económico no se puede obviar, ya que no importa lo noble que sea un proyecto si este no se puede mantener, entonces no tiene sentido iniciarlo, al igual que si este no beneficia a la sociedad, no tiene sentido que sea llevado a cabo.

Por lo que en este proyecto, se ha reflexionado y tenido en cuenta los ámbitos más conocidos de la sostenibilidad como lo puede ser la dimensión ambiental, así como otras áreas que no siempre se les da la importancia que merecen pese a que son igual de importantes, como lo son la económica y la social.

6.6.2 Dimensión Económica

En cuanto a la visión económica, estamos ante un caso en el que al tratarse de las últimas técnicas en *Deep Learning*, los requisitos de los equipos necesarios para entrenar estas redes no es pequeño.

No obstante debido a la misma naturaleza de esta área, una vez entrenado nuestro modelo este ya no tendría la necesidad de volver a ser entrenado nuevamente, por lo que el coste de inferencia de un modelo es despreciable, ya que estamos hablando de que un móvil ya tiene la potencia necesaria para hacer esta tarea.

Por lo que si bien la entrada tiene un alto coste, el posterior mantenimiento económico y sostenibilidad económica se reducen prácticamente a 0, mientras que se hace uso del modelo, y,

por lo tanto, se obtienen beneficios. Aunque no solo eso, una vez entrenado, se puede utilizar posteriormente como base para pre entrenar para otras tareas reduciendo significativamente el tiempo necesario para hacerlo y así reduciendo el coste económico de futuros proyectos.

6.6.3 Dimensión Ambiental

Desgraciadamente el impacto que puede llegar a tener este proyecto en el medioambiente puede llegar a ser notable, no porque el modelo sea directamente perjudicial o vaya en contra del mismo, sino porque para entrenar cualquier modelo, debido a las prestaciones mínimas del equipo, este consumirá bastante energía, lo cual se traduce en contaminación casi directa.

Por supuesto se intenta mitigar esto mirando de que las fuentes de donde se consume esta energía sean fuentes de energía verde respetuosamente con el medio ambiente, también se han tomado medida para evitar en todo momento repetir experimentos innecesarios, ya que si bien puede haber necesidad de repetir experimentos cuanto menos ocurra mejor.

Aun de esta forma, debido a que si queremos obtener, o como mínimo intentar obtener, resultados cercanos o superiores al estado del arte no tenemos otra opción que recurrir a este tipo de recursos.

6.6.4 Dimensión Social

Los beneficios sociales, como ya he comentado previamente, son inmensos. Ya que para comenzar atacamos un problema real y existente no solo en empresas sino en organismos públicos.

Aparte de eso como se ha comentado repetidas veces una vez entrenado el modelo, este puede ser usado por cualquier persona en cualquier lugar sin ningún tipo de problema, aportando un beneficio directo para esta persona. O bien se puede utilizar para el entrenamiento en otra área, que haciendo empleo del *transfer learning* podrían obtener mejores resultados y verse beneficiados.

Por lo que en este proyecto el beneficio social es mucho mayor que en otros, ya que es un beneficio directo para un público muy amplio y donde no se necesitan más recursos que un ordenador doméstico para poder ponerlo a prueba.

7 Conclusión

7.1 Conclusiones personales

En este proyecto se han planteado diversos modelos de deep learning que han hecho uso de convolucionales y Transformers, algo nunca antes hecho, para la tarea de segmentación de documentos logrando resultados que superan al estado del arte actual.

Además, se ha hecho un estudio exhaustivo sobre los datasets seleccionados, gracias al cual se ha detectado un problema, por lo que se ha ideado una solución para solucionarlo permitiendo así entrenar nuestros modelos con y sin esta corrección en el dataset para visualizar el impacto en el rendimiento.

El código utilizado se puede encontrar en el siguiente enlace: `https://github.com/Asocsar/Segmentacion-de-documentos-digitales`.

7.2 Objetivos cumplidos

Todos los objetivos propuestos en un inicio han sido alcanzados con éxito, ya que los modelos construidos han sido capaces de sobrepasar no solo el 70% de *accuracy* que inicialmente se planteó como un mínimo a lograr, sino que hemos sido capaces de obtener en diversas propuestas de modelo un rendimiento superior al obtenido en trabajos de investigación previos que hasta el día de hoy se consideran el estado del arte en esta tarea de segmentación.

Y finalmente, hemos sido capaces de aplicar un método de entrenamiento en paralelo que nos ha permitido acortar los tiempos lo suficiente como para poder ser capaces de entrenar nuestros modelos en menos de 48h.

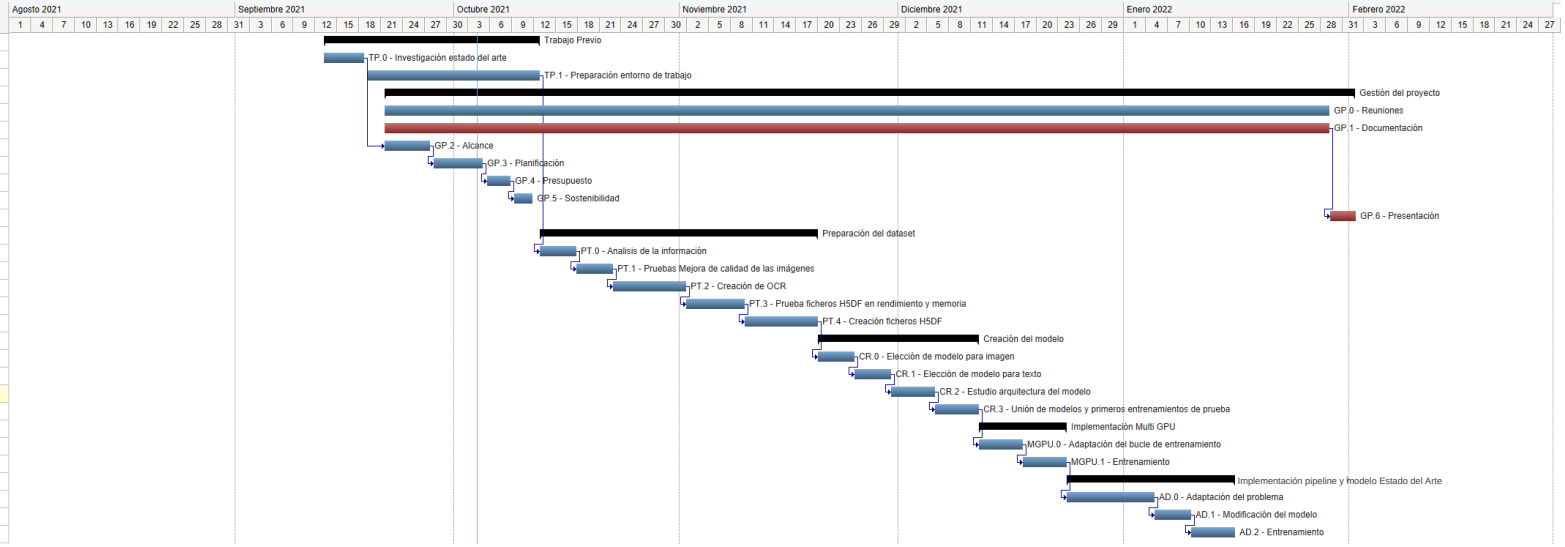
7.3 Trabajo futuro

En estos 4 meses hemos invertido cientos de horas en desarrollar un modelo que sobrepase o iguale al estado del arte actual, no solo para obtener resultados, sino también para hacer una exploración sobre el rendimiento de los Transformers en este tipo de tareas, demostrando resultados impresionantes como ya hemos visto, por lo que en vista de esto nuestro trabajo está lejos de terminar, en vista del rendimiento de Transformers con convolucionales nos queda investigar el rendimiento de hacer uso únicamente de Transformers. Después de un análisis más reciente se ha determinado que el mejor candidato es LayoutLM_V2, un modelo creado por Microsoft, por lo que en un futuro llevaremos a cabo las pruebas con este modelo para ver si es posible no tratar con redes convolucionales y obtener un rendimiento equivalente en el problema de segmentación de documentos.

gantter™

Project Name **GEP**

	Nombre	Inicio	Fin	Predecesoras
1	**Trabajo Previo**	**13/09/2021**	**12/10/2021**	
2	TP.0 - Investigación estado del arte	13/09/2021	18/09/2021	
3	TP.1 - Preparación entorno de trabajo	19/09/2021	12/10/2021	
4	**Gestión del proyecto**	**21/09/2021**	**01/02/2022**	
5	GP.0 - Reuniones	21/09/2021	28/01/2022	
6	GP.1 - Documentación	21/09/2021	28/01/2022	
7	GP.2 - Alcance	21/09/2021	27/09/2021	2
8	GP.3 - Planificación	28/09/2021	04/10/2021	7
9	GP.4 - Presupuesto	05/10/2021	08/10/2021	8
10	GP.5 - Sostenibilidad	09/10/2021	11/10/2021	9
11	GP.6 - Presentación	28/01/2022	01/02/2022	6
12	**Preparación del dataset**	**12/10/2021**	**19/11/2021**	
13	PT.0 - Analisis de la información	12/10/2021	17/10/2021	3
14	PT.1 - Pruebas Mejora de calidad de las imágenes	17/10/2021	22/10/2021	13
15	PT.2 - Creación de OCR	22/10/2021	01/11/2021	14
16	PT.3 - Prueba ficheros H5DF en rendimiento y memoria	01/11/2021	09/11/2021	15
17	PT.4 - Creación ficheros H5DF	09/11/2021	19/11/2021	16
18	**Creación del modelo**	**19/11/2021**	**11/12/2021**	
19	CR.0 - Elección de modelo para imagen	19/11/2021	24/11/2021	17
20	CR.1 - Elección de modelo para texto	24/11/2021	29/11/2021	19
21	CR.2 - Estudio arquitectura del modelo	29/11/2021	05/12/2021	20
22	CR.3 - Unión de modelos y primeros entrenamientos de prueba	05/12/2021	11/12/2021	21
23	**Implementación Multi GPU**	**11/12/2021**	**23/12/2021**	
24	MGPU.0 - Adaptación del bucle de entrenamiento	11/12/2021	17/12/2021	22
25	MGPU.1 - Entrenamiento	17/12/2021	23/12/2021	24
26	**Implementación pipeline y modelo Estado del Arte**	**23/12/2021**	**15/01/2022**	
27	AD.0 - Adaptación del problema	23/12/2021	04/01/2022	25
28	AD.1 - Modificación del modelo	04/01/2022	09/01/2022	27
29	AD.2 - Entrenamiento	09/01/2022	15/01/2022	28



PDF Generated On: 10/04/2021, 16:44:16

Page 1 / 1

Figure 12: Gráfica de la plánificación temporal del proyecto.

References

[1] "Barcelona supercomputing center." https://www.bsc.es/, 2021. Visitado (27-09-2021).

[2] I. P. Z. S.-t. X. Zhong-Qiu Zhao, Member and F. Xindong Wu, "Object detection with deep learning: A review." https://arxiv.org/pdf/1807.05511.pdf&usg=ALkJrhhpApwNJOmg83O8p2Ua76PNh6tR8A, 2019. Visitado (21-09-2021).

[3] Y. L. Arjun Jain, Jonathan Tompson and C. Bregler, "Modeep: A deep learning framework using motion features for human pose estimation." https://arxiv.org/pdf/1409.7963.pdf, 2019. Visitado (22-09-2021).

[4] R. T. Siddhant Srivastava, Anupam Shukla, "Machine translation : From statistical to modern deep-learning practices." https://arxiv.org/pdf/1812.04238.pdf, 2018. Visitado (22-09-2021).

[5] T. K. W. Z.-C. Y. Chuanqi Tan, Fuchun Sun and C. Liu, "A survey on deep transfer learning." https://arxiv.org/pdf/1808.01974.pdf, 2018. Visitado (24-09-2021).

[6] "Capa densa." https://towardsdatascience.com/applied-deep-learning-part-1-artificial-neural-netwo

[7] "Convolutional neural networks, explained." https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939#:~:text=A%20Convolutional%20Neural%20Network%2C%20also,topology%2C%20such%20as%20an%20image.&text=Each%20neuron%20works%20in%20its,cover%20the%20entire%20visual%20field.

[8] S. K. Mark HUGHE, Irene LI and T. SUZUMURA, "Medical text classification using convolutional neural networks." https://arxiv.org/pdf/1704.06841.pdf, 2017. Visitado (20-09-2021).

[9] A. Sherstinsky, "Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network." https://arxiv.org/pdf/1808.03314.pdf, 2021. Visitado (21-09-2021).

[10] N. P. J. U.-L. J. A. N. G. K. Ashish Vaswani, Noam Shazeer, "Attention is all you need." https://arxiv.org/pdf/1706.03762.pdf, 2017. Visitado (25-10-2021).

[11] A. K. D. W.-X. Z. T. U. M. D. M. M. G. H. S. G. J. U. N. H. Alexey Dosovitskiy, Lucas Beyer, "An image is worth 16x16 words: Transformers for image recognition at scale." https://arxiv.org/pdf/2010.11929.pdf, 2021. Visitado (26-10-2021).

[12] N. R. M. S. J. K. P. D. A. N. P. S. G. S. A. A. S. A. A. H.-V. G. K. T. H. R. C. A. R. D. M. Z. J. W. C. W. C. H. M. C. E. S. M. L. S. G. B. C. J. C. C. B. S. M. A. R. I. S. D. A. Tom B. Brown, Benjamin Mann, "Language models are few-shot learners." https://arxiv.org/pdf/2005.14165v4.pdf, 2020. Visitado (26-10-2021).

[13] "Understand the impact of learning rate on neural network performance." https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/.

[14] J. A. S. L. Fabricio Ataides Braz, Nilton Correia da Silva, "Leveraging effectiveness and efficiency in page stream deep segmentation." https://www.sciencedirect.com/science/article/pii/S0952197621002426, 2020. Visitado (24-09-2021).

[15] A. Z. Karen Simonyan, "Very deep convolutional networks for large-scale image recognition." https://arxiv.org/pdf/1409.1556.pdf, 2015. Visitado (24-09-2021).

[16] G. H. Gregor Wiedemann, "Page stream segmentation with convolutional neural nets combining textual and visual features." https://arxiv.org/pdf/1710.03006.pdf, 2019. Visitado (23-09-2021).

[17] A. B. Chems Neche, Yolande Belaïd, "Use of language models for document stream segmentation." https://hal.inria.fr/hal-02975046/document, 2020. Visitado (20-10-2021).

[18] A. U. Adam W. Harley and K. G. Derpanis, "The rvl-cdip dataset." https://www.cs.cmu.edu/~aharley/rvl-cdip/, 2017. Visitado (21-09-2021).

[19] S. Hoffstaetter, "Python tesseract." https://github.com/madmaze/pytesseract, 2019. Visitado (23-09-2021).

[20] J. C. T. W. Victor SANH, Lysandre DEBUT, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter." https://arxiv.org/pdf/1910.01108.pdf, 2020. Visitado (25-10-2021).

[21] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks." https://arxiv.org/pdf/1905.11946v5.pdf, 2020. Visitado (23-10-2021).

[22] J. D. M.-W. C. K. L. K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding." https://arxiv.org/pdf/1810.04805.pdf, 2019. Visitado (24-10-2021).

[23] J. D. Geoffrey Hinton, Oriol Vinyals, "Distilling the knowledge in a neural network." https://arxiv.org/pdf/1503.02531.pdf, 2015. Visitado (25-10-2021).

[24] J. L. B. Diederik P. Kingma, "Adam: A method for stochastic optimization." https://arxiv.org/pdf/1412.6980.pdf, 2017. Visitado (04-10-2021).

[25] S. M. Herbert Robins, "A stochastic approzimation method." https://projecteuclid.org/journals/annals-of-mathematical-statistics/volume-22/issue-3/A-Stochastic-Approximation-Method/10.1214/aoms/1177729586.full, 1951. Visitado (04-10-2021).

[26] Y. S. John Duchi, Elad Hazan, "Adaptive subgradient methods for online learning and stochastic optimization." https://www.jmlr.org/papers/volume12/duchi11a/duchi11a.pdf, 2011. Visitado (04-10-2021).

[27] A. Graves, "Generating sequences with recurrent neural networks." https://arxiv.org/pdf/1308.0850.pdf, 2014. Visitado (04-10-2021).

[28] D. X. A. C. Y. B. Ian J. Goodfellow, Mehdi Mirza, "An empirical investigation of catastrophic forgetting in gradient-based neural networks." https://arxiv.org/pdf/1312.6211.pdf, 2015. Visitado (26-10-2021).

[29] B. G. Sharath Sreenivas, Swetha Mandava and C. Forster, "Pretraining bert with layer-wise adaptive learning rates." https://developer.nvidia.com/blog/pretraining-bert-with-layer-wise-adaptive-learning-rates/, 2019. Visitado (21-10-2021).

[30] "Multiprocessing package - torch.multiprocessing." https://pytorch.org/docs/stable/multiprocessing.html.

[31] "Pytorch distributed." https://pytorch.org/tutorials/beginner/dist_overview.html.

[32] "Getting started with distributed data parallel." https://pytorch.org/tutorials/intermediate/ddp_tutorial.html.

[33] L. J. E. S. Xiangyang Xu, Shengzhou Xu, "Characteristic analysis of otsu threshold and its applications." https://www.sciencedirect.com/science/article/abs/pii/S0167865511000365, 2011. Visitado (24-09-2021).

[34] "Slack." https://slack.com/. Visitado (24-09-2021).

[35] "Github." https://github.com/, 2021. Visitado (27-09-2021).

[36] "Gantter." https://www.gantter.com/, 2021. Visitado (27-09-2021).

[37] "Hays." https://www.hays.es/, 2021. Visitado (04-10-2021).

[38] "Aws." https://calculator.aws/#/, 2021. Visitado (06-10-2021).