

Clasificación, detección y análisis de letras escritas a mano y por ordenador.

Jordi Morales (1564921), Mireia Fernández (1562636),
Marina Vázquez (1563735)

Abstract— En este artículo exponemos el trabajo hecho para el proyecto final de la asignatura “Visión por Computador” de la Universidad Autònoma de Barcelona.

Consiste en la construcción de un programa que nos permitirá clasificar textos según su origen, segmentarlos para encontrar palabras y, finalmente, implementar un reconocimiento óptico de caracteres.

Veremos que el producto final difiere según el origen de la muestra: para las imágenes digitales se llegará a poder leer los resultados en los textos de salida; en cambio, para los documentos manuscritos, encontraremos problemas para reconocer las palabras encontradas.

Keywords— Image classification, Character classification, Machine Learning, Image Segmentation, Convolutional Neural Networks, Deep Learning, Optical Character Recognition, Computer Vision.

1. INTRODUCCIÓN

La Visión por Computador es una disciplina que nos permite entender cómo los ordenadores procesan y analizan la vida real a través de imágenes. Por lo tanto, como proyecto final de la asignatura, este trabajo pretende profundizar en los conceptos que se han presentado a lo largo del semestre, a la vez que incorporamos técnicas de aprendizaje computacional para clasificar textos.

Hemos escogido este tema en concreto porque el análisis de texto a partir de imágenes es un campo que aún no hemos tenido oportunidad de trabajar en profundidad. Queremos construir un proyecto que nos permita explorar, dentro de lo posible, las posibilidades que nos ofrece la visión artificial.

Por ende, nos hemos marcado tres hitos a cumplir que nos ayudaran a sacar el máximo potencial de esta experiencia.

En primer lugar, programaremos un clasificador de textos según su origen: querremos saber si se trata de textos manuscritos o generados digitalmente. Para ello, buscaremos construir un clasificador que nos permita diferenciar la escritura a mano de la no manuscrita usando modelos convolucionales sencillos.

A continuación, como segundo objetivo, buscaremos usar técnicas de segmentación para la detección de textos en una imagen. Esto nos permitirá trabajar con varias herramientas cuya utilidad se especializa en el tratado de imágenes.

Por último, buscaremos construir un programa que, aprendiendo a clasificar letras, nos permita hacer una conversión de imágenes a texto. Así, después de procesar una imagen dada usando las técnicas desarrolladas en los objetivos anteriores, querremos obtener un archivo de texto que contenga el mensaje del input original escrito en él.

2. ESTADO DEL ARTE

La necesidad de extraer información de documentos, tanto escritos como impresos, de forma automatizada ha existido desde antes que los propios ordenadores. Como ejemplo tenemos el *Optophone*, un dispositivo orientado para gente ciega, que permitía escanear documentos para obtener sonidos que identificaban letras.

Alrededor de los años 70 y 80 aparecieron las primeras tecnologías de OCR (del inglés Optical Character Recognition), que rápidamente se popularizaron entre empresas de todos los sectores. El principal exponente de esta nueva moda es *Omni-font*, el cual era capaz de reconocer y convertir texto a un formato digital, sin importar la fuente o tamaño de la letra.

El rendimiento de estos sistemas, especialmente en cuanto a reconocimiento y clasificación de caracteres, fue mejorando de manera estable a medida que los acercamientos clásicos se sofisticaban y el poder de cómputo aumentaba. Al entrar el nuevo siglo, se empezaron a popularizar diferentes servicios que ofrecían acceso a programas de extracción de texto, tanto gratuitos como de pago (WebOCR, Tesseract, Adobe Capture, etc.).

El siguiente gran paso llegó en 2013 cuando se empleó por primera vez una red neuronal, entrenada sobre MNIST [5]. Se obtuvieron tasas de error significativamente más bajas que lo visto hasta el momento, lo que inició una revolución que convertiría a las técnicas de aprendizaje profundo en el estándar para OCR, acercando la precisión de estos sistemas a niveles comparables con los del ser humano. Como ejemplo tenemos [6], donde se presenta un modelo que usa una red neuronal convolucional para extraer las características de alto nivel de las imágenes de entrada, las cuales son secuenciadas e introducidas a una LSTM, una red neuronal recurrente de memoria corta a largo término, para generar el texto final.

3. METODOLOGIA

En esta sección entraremos en detalle sobre nuestras propuestas para lograr los objetivos mencionados previamente. Asimismo, hablaremos también de los artículos que nos servirán como referencia en cada apartado y los datasets de imágenes que usaremos correspondientemente.

3.1. Clasificación de texto por origen

Nuestra primera meta consiste en programar un clasificador con un objetivo sencillo: diferenciar entre texto manuscrito y digital. Para ello, utilizaremos dos bancos de imágenes distintos: la recopilación de manuscritos GoodNotes[13] y una recopilación de documentos digitales, DocVQA [3].

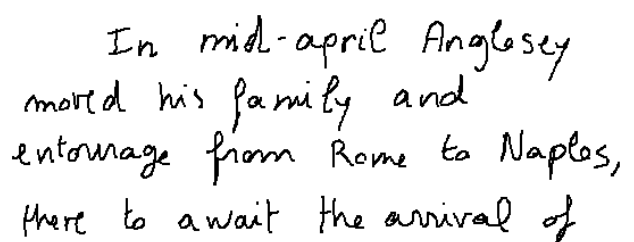


Fig. 1: Ejemplo de imagen de un manuscrito de [2].

Nos gustaría documentar también la importancia del preprocesado de dichas imágenes, por lo que utilizaremos [1] como principal guía para extraer únicamente el texto de nuestros datos de entrada. Para el modelo en si, utilizaremos un bosque aleatorio de árboles de clasificación.

Además, para verificar los resultados, tomaremos unas pocas imágenes a color de escritos en bolígrafo azul, por ejemplo, para comparar la precisión cuando las imágenes originales no están en escala de grises.

3.2. Segmentación de texto en imágenes

La detección de los textos en imágenes es nuestro segundo objetivo y lo resolveremos a partir de la segmentación. Para ello, compararemos diferentes técnicas estudiadas en la asignatura. Algunos ejemplos de métodos para obtener imágenes segmentadas son: la binarización, regiones, clústering, *watershed*, *snakes*, entre otros. A partir de los resultados obtenidos, optimizaremos nuestro algoritmo para que solo haga uso de la más adecuada.

Seguidamente, buscaremos separar dicho texto en subgrupos: palabras enteras o letras. Las contaremos para estudiar el resultado de la segmentación realizada. En esta última tarea explicada, tenemos previsto usar *DBSCAN* [7], un algoritmo que, a partir de la computación de distancias, nos permite agrupar espacialmente según las densidades que presentan las líneas de texto.

3.3. Conversión de imágenes a texto

Obtener un documento de texto donde se recopile la información escrita en una imagen es nuestra tercera gran meta. Supondrá combinar las ideas trabajadas en los puntos anteriores, esta vez con el objetivo de substraer y procesar letra a letra.

Siguiendo las bases establecidas por las infraestructuras diseñadas en [4]. Buscaremos construir una adaptación de dichos diseños que nos permita obtener la salida deseada a partir de las imágenes, usando los textos encontrados por segmentación como etiquetas semánticas.

4. EXPERIMENTOS, RESULTADOS Y ANÁLISIS

En esta sección se expnderán los resultados que se han obtenido, asimismo también se discutirá sobre los fallos o cambios realizados durante el transcurso del estudio.

4.1. Clasificación del origen de las imágenes

El primer objetivo de nuestro trabajo consiste en poder automatizar la clasificación del origen de un documento. Esto nos será útil en el paso 4.3.2 explicado a continuación. Para ello, hemos separado los datos con una partición 80/20 que nos permitirá entrenar y validar los modelos. Hemos decidido combinar dos algoritmos: un método de reducción de dimensionalidad para reducir el espacio de atributos y un método de aprendizaje computacional para la clasificación final.

Dado que para utilizar esos métodos necesitamos un número fijo de variables, antes de nada se ha escalado el conjunto de datos a imágenes de 647x670 píxeles -medidas que corresponden con las mínimas dimensiones de las imágenes de entrenamiento-. Entonces, éstas se han usado para entrenar un modelo de análisis de componentes principales [14]. Como se puede observar en los gráficos de la figura 2, hemos escogidos las dimensiones necesarias para explicar el 90 % de la varianza explicada.

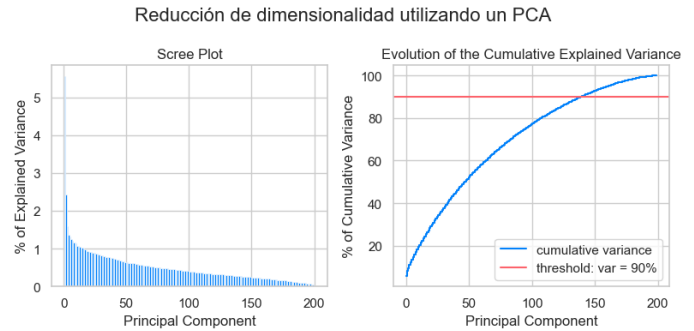


Fig. 2: Gráficos informativos sobre los componentes del análisis de reducción de dimensiones

El conjunto modificado de datos resultantes consiste, por lo tanto, en sólo 140 valores que describen nuestras imágenes. Éstos, son los datos que hemos usado para entrenar nuestro modelo *RandomForest* [8] con 150 árboles de decisión.

En la figura 3 podemos observar la matriz de confusión obtenida con los datos de validación, donde la precisión obtenida es del 90 %. Observamos que el error se encuentra en los documentos que se clasifican como manuales siendo digitales. En concreto, hemos visto que nuestras imágenes digitales contienen dibujos o gráficos que podrían causar confusión.

True label	Digital	40	10
	Manual	0	50
		Digital	Manual
		Predicted label	

Fig. 3: Matriz de confusión de la clasificación del origen de las imágenes utilizando árboles de decisión

4.2. Segmentación de palabras

La segmentación es un procedimiento muy importante para poder partir de una buena base en el siguiente apartado, la conversión de imágenes a texto. La finalidad de la segmentación es detectar las palabras que contienen las imágenes de nuestro conjunto de datos creado.

Para conseguir nuestro objetivo, antes de nada hemos preprocesado las imágenes de la siguiente manera: primero, hemos hecho un reescalado dinámico para mantener las proporciones de los documentos; después, un *Black Hat* que consiste en resaltar la parte negra de las imágenes, en nuestro caso el texto; en tercer lugar, hemos hecho una binarización; finalmente, hemos aplicado una dilatación para agrupar letras en palabras.

Una vez tenemos todo esto, trabajamos en segmentación utilizando la librería *cv2*. Encontramos los contornos y obtenemos las cajas con el texto detectado. Es necesario realizar un filtrado

de las cajas encontradas con el objetivo de eliminar falsas detecciones que carecen de sentido, quitando así las detecciones de diagramas o gráficas, por ejemplo.

Como se observa en la sección 4.4.1, los resultados obtenidos son variados. Encontramos que hay imágenes en las que la segmentación funciona pero también encontramos que hay otras donde no detecta apenas nada. Explicamos la agrupación de más de una palabra en una única caja como resultado de la dilatación para segmentar, puesto que estamos reajustando el espacio entre ellas y nuestro programa podría confundirse.

4.3. Reconocimiento óptico de caracteres

Como último experimento buscamos combinar todo lo presentado en los apartados anteriores, junto con algunos elementos que introduciremos a continuación, para crear un sistema OCR capaz de recibir una imagen y devolver palabras segmentadas y con su texto correspondiente extraído.

4.3.1. Clasificador de caracteres

Para dar el paso de una imagen a un caracter necesitaremos un clasificador. Optamos por utilizar un modelo ResNet-18 [12], preentrenado con ImageNet [10], el cual ajustaremos haciendo fine-tuning sobre Chars74k [9], un conjunto de datos con alrededor de 66,000 imágenes divididas en 62 clases (números del 0 al 9 y todas las letras del alfabeto en mayúsculas y minúsculas) con muestras tanto escritas a mano como provenientes de diversas fuentes digitales.

Hacemos una partición 80/20 de los datos para crear los conjuntos de entrenamiento y validación. Entrenamos durante 10 épocas, con un *learning rate* de 0.001 y utilizando un optimizador Adam [11]. La mejor iteración obtiene una precisión de 84.56 % durante la validación.

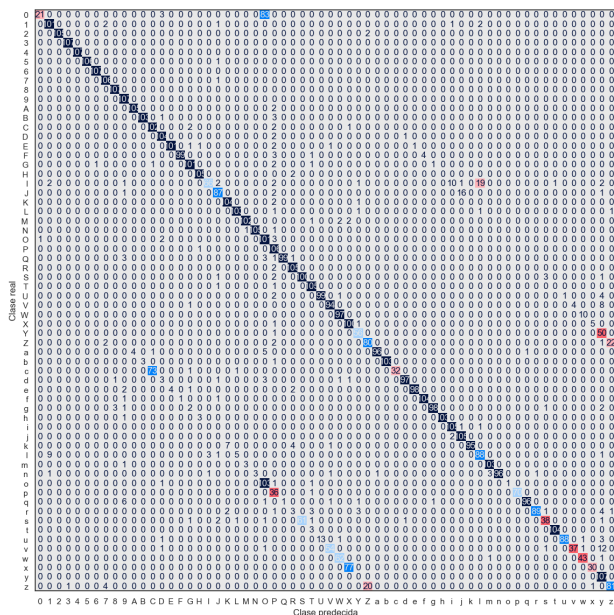


Fig. 4: Matriz de confusión de la clasificación de caracteres

En la Figura 4 podemos observar que, exceptuando el caso entre la letra 'O' y el número '0', una gran parte de los fallos se concentran a lo largo de dos líneas diagonales a ambos lados de la diagonal principal. Esto sugiere que, pese a que la precisión del

modelo parece no ser excelente, la mayoría de errores están relacionados con confundir una letra minúscula con su contraparte mayúscula, por lo que el resultado final seguirá siendo entendible.

4.3.2. Algoritmo OCR

Lo primero que se hace al recibir una imagen de entrada es utilizar el clasificador descrito en el Apartado 4.1 para detectar su origen. En función del tipo de documento que se haya determinado, preprocesamos la imagen acorde con lo visto en apartados anteriores y segmentamos las palabras del texto como en el Apartado 4.2. En función del origen del documento emplearemos un método de extracción diferente:

- **Documentos digitales:** Con documentos impresos o digitales, el tipo de escritura y su presentación es generalmente muy regular. Esto facilita la tarea de segmentación de caracteres, ya que se puede asumir que en una mayoría de casos existirá una separación entre letra y letra. Esta idea puede verse afectada negativamente dependiendo de la calidad del proceso de digitalización de los documentos, o por el preprocesado al que sometemos a las imágenes, pero el producto final normalmente continúa siendo reconocible. Para cada imagen de una palabra segmentada buscamos las separaciones entre caracteres, es decir, aquellas columnas sin píxeles negros. Utilizamos estos espacios para dividir la palabra inicial en subimágenes de caracteres, las cuales son descartadas si no observamos píxeles negros en las filas centrales, ya que asumimos que representan signos de puntuación o alguna irregularidad de la imagen original. Los caracteres restantes se preprocesan añadiendo *padding* hasta obtener las dimensiones adecuadas y se pasan por el clasificador de caracteres. Las predicciones del modelo para cada letra conformará la cadena de caracteres que describe una palabra.
- **Documentos manuscritos:** A diferencia que con el caso anterior, no podemos esperar regularidad en la escritura manual. El texto de este tipo de documentos tiende a no dejar separaciones verticales entre letras, si es que se deja algún tipo de separación. Decidimos optar por un acercamiento más superficial utilizando una ventana deslizante. Para cada palabra, empleamos una ventana con la misma altura que la imagen de la palabra segmentada y un ancho de del 80 % de la altura. Desplazaremos esta ventana de izquierda a derecha con saltos de 5 píxeles. Igual que para documentos digitales, filtramos, preprocesamos y utilizamos el clasificador de caracteres para determinar que se está viendo por la ventana en cada iteración. Al final del proceso obtenemos una cadena de caracteres, la cual se compara contra un diccionario con las 1000 palabras más comunes de la lengua inglesa, y la más parecida será el resultado final.

Para ambos casos al final obtenemos una lista, con formato estilo *json*, con la *bounding box* de cada palabra y el texto extraído asociado.

4.4. Resultados

Por último, mostraremos los resultados obtenidos para explicar las fortalezas y debilidades de nuestro trabajo.

4.4.1. Documentos digitales

Para poder medir el rendimiento de nuestro programa sobre los documentos digitales hemos escogido las siguientes métricas.

- **IoU (Intersection over Union):** Puntuación que cuantifica el grado de solapamiento entre dos cajas, se calcula $\frac{Intersección(A,B)}{Unión(A,B)}$ donde A y B son las dos cajas a comparar. Lo usaremos para identificar cómo de bien segmentamos las palabras.

Puntuación IoU	
Sólo Cajas Relevantes	Todas las Cajas
28.21 %	13.59 %

Taula 1: Media de puntuación IoU de todos los documentos según el tipo de las cajas tenidas en cuenta

Como podemos ver en la tabla 1, si tenemos en cuenta sólo aquellas cajas que realmente se encuentran solapadas con la verdad básica establecida en nuestra base de datos, obtenemos alrededor de un 30 % de solapamiento. En cambio, si tenemos en cuenta todas las cajas que encuentra nuestro algoritmo, el porcentaje disminuye a la mitad.

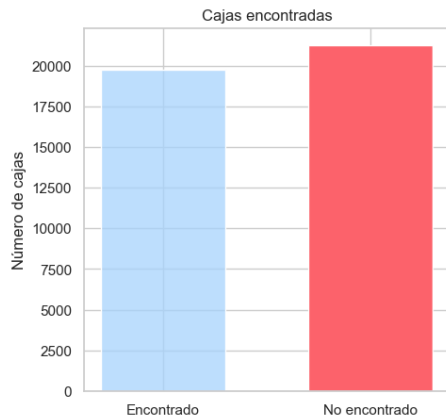


Fig. 5: Histograma comparativo del número de cajas relevantes encontradas contra las cajas irrelevantes.

En la figura 5 podemos ver fácilmente que nuestro programa no ha detectado, o no ha encontrado con suficiente precisión, un poco más de la mitad de las palabras de todos los documentos. Este hecho ayuda a entender por qué el índice de solapamiento disminuye tanto cuando tenemos en cuenta todas las muestras de nuestro conjunto de datos.

Por último, la figura 6 nos ayuda a entender más detalladamente la distribución de puntuaciones de cada caja. En el gráfico de la izquierda vemos el histograma que recoge la información de todas las cajas individualmente. En el gráfico de la derecha vemos las puntuaciones finales para cada documento único; en este segundo caso, vemos que la cola de la derecha se hace mucho menos visible, pero nos informa que existen documentos cuya puntuación global está por encima del 50 % de solapamiento.

- **Distancia de Levenshtein:** Distancia entre dos palabras que devuelve el número mínimo de cambios requeridos para transformar la primera cadena de caracteres en la segunda.

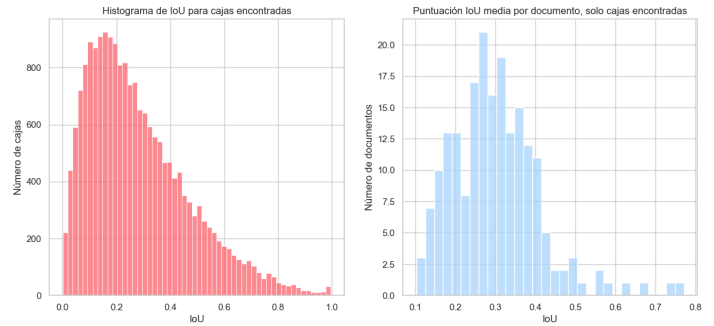


Fig. 6: Histograma del número de cajas encontradas según su puntuación IoU tanto por cajas como por documento

Para esta métrica, recogida solo de aquellas cajas que suponemos relevantes, hemos obtenido una puntuación global de 6. Esto significa que necesitaríamos hacer seis cambios -reemplazar un carácter, añadirlo, quitarlo, etc.- de media para llegar de una predicción a la cadena real.

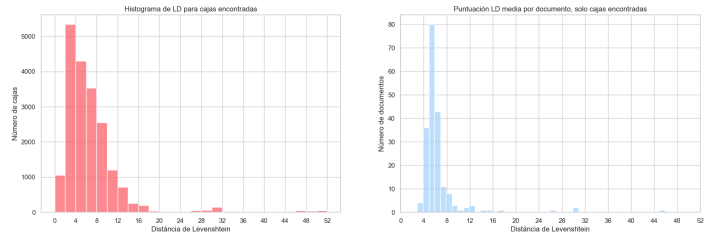


Fig. 7: Histograma del número de cajas encontradas según su distancia de Levenshtein tanto por cajas como por documento

En la figura 7 observamos que más del 90 % de los documentos presentan una distancia entre 3 y 12. Aún así, vemos que aparecen cajas relevantes cuya distancia entre la palabra real y aquella detectada es 51. Observando al detalle, descubrimos que estos números aparecen porque agrupamos varias palabras juntas y se procesan como una única cadena.

4.4.2. Documentos manuscritos

Al procesar los documentos manuscritos, nos hemos dado cuenta que no hemos obtenido los resultados esperados. En los ejemplos que hay a continuación se puede ver que aunque la segmentación funciona como es debido, mayoritariamente, la transcripción de palabras no podrías estar más lejos de la realidad.

- **Casos de segmentación fallida:** No se ha dado muchas veces, pero cuando en la imagen de muestra hay ruido en el fondo, la segmentación no nos funciona y no se detecta ninguna letra. Este hecho se puede ver perfectamente en 8.
- **Casos de interpretación errónea de los caracteres:** Como se ha explicado al principio, nuestra mayor inconveniencia han sido los resultados de la interpretación de las letras para escanear las palabras encontradas en las cajas. En el siguiente ejemplo, figura 9, observamos que la segmentación ha funcionado: a pesar de juntar un par de palabras en la misma caja, o separar una sola en varios trozos, se puede ver que muchas están segmentadas correctamente. Dado que los ejemplos manuales suelen estar torcidos y es más complicado segmentar letra a letra, nuestro código procesa la palabra entera para sacar un resultado con la cadena que cree encontrar en dicha caja.

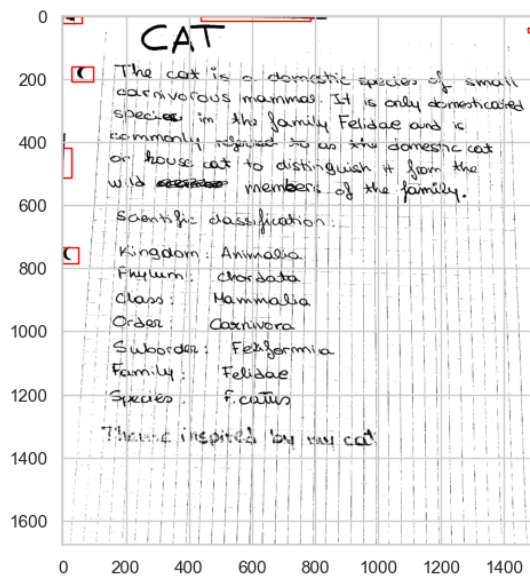


Fig. 8: Ejemplo de prueba escrito por nosotros donde no se ha segmentado ninguna palabra.

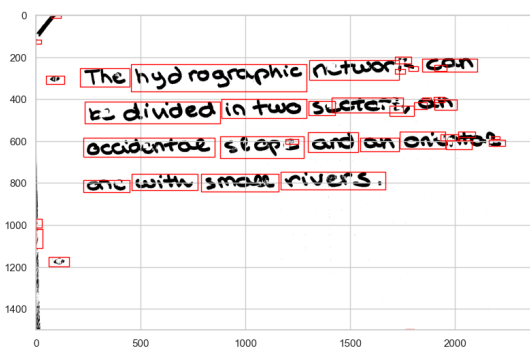


Fig. 9: Ejemplo de segmentación escrito por nosotros

En las figuras 10 y 11 podemos observar este suceso. En el primer caso, la palabra está bien segmentada pero cuando se pide que enseñe qué palabra encuentra en ese trozo de la imagen la respuesta es “man”. En la misma línea, en la segunda imagen vemos que se ha devuelto una única caja para dos palabras; aquí, esperaríamos una salida que fuera una combinación de ambas, pero para nuestra sorpresa la palabra que el código dice encontrar ahí es “wood”.



Fig. 10: Ejemplo de segmentación de palabras: “river”.

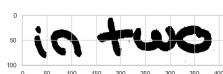


Fig. 11: Ejemplo de segmentación de palabras: “in two”.

5. CONCLUSIONES

En este artículo hemos presentado tres tareas clásicas en la extracción automática de información de documentos en imágenes:

Clasificación de documentos según el tipo de escritura, segmentación de palabras y Reconocimiento óptico de caracteres. Además, hemos creado un conjunto de datos nuevo y hemos balanceado con 500 imágenes de documentos digitales y escritos a mano (Con una partición de 400 para entrenamiento y 100 para validación), extraídas de los datasets DocVQA [3] y GoodNotes [13].

En la tarea de clasificación, hemos explorado modelos adecuados para la separación de documentos dependiendo de su origen. Hemos visto que nos ha sido posible clasificarlos utilizando un modelo de aprendizaje computacional clásico con una precisión excelente.

Para la segmentación de palabras, hemos observado los distintos resultados, con algunas imágenes hemos conseguido cumplir nuestro objetivo pero para otras, no lo hemos conseguido. Hemos visto que dependía bastante del tipo de hoja usada y del espacio entre las palabras. No obstante, en general hemos superado el objetivo con éxito.

Como último objetivo, hemos propuesto dos soluciones para la segmentación de caracteres y posterior extracción de texto, adaptadas a las particularidades de cada tipo de documento. En documentos digitales hemos aprovechado la regularidad en la separación entre caracteres para hacer la segmentación. De esta manera hemos obtenido resultados razonablemente satisfactorios en la calidad de la extracción. Para documentos manuscritos, hemos optado por utilizar una ventana deslizante e ir clasificando lo que se ve a cada iteración, para al final devolver la palabra más parecido de un diccionario con las 1000 palabras más comunes de la lengua inglesa. Nos hemos decidido por este método para poder ignorar la diversidad de presentaciones de esta clase de documentos, pero no hemos observado que nuestro algoritmo consiga acercarse a la realidad.

Por último, creemos que el siguiente paso para conseguir mejores resultados sería aplicar pequeños cambios en los tres pasos por separado. Para la clasificación por origen, estaría bien observar el rendimiento de un método al que le pasáramos atributos de las imágenes en vez de las imágenes en si. Para la sección de segmentación, creemos conveniente reforzar la extracción del fondo para que muestras como la figura 8 den resultados positivos. Finalmente, creemos que sería necesario un método capaz de determinar de forma más efectiva cuál es la región que ocupa cada carácter para el apartado final.

REFERENCIAS

- [1] P. Barlas, S. Adam, C. Chatelain and T. Paquet, “A Typed and Handwritten Text Block Segmentation System for Heterogeneous and Complex Documents”, 2014 11th IAPR International Workshop on Document Analysis Systems, Tours, France, 2014, pp. 46-50.
- [2] Liwicki, M. and Bunke, H., “IAM-OnDB - an On-Line English Sentence Database Acquired from Handwritten Text on a Whiteboard. 8th Intl. Conf. on Document Analysis and Recognition”, 2005, Volume 2, pp. 956 - 961.
- [3] M. Minesh, D. Karatzas, and C.V. Jawahar. “Docvqa: A dataset for vqa on document images.” Proceedings of the IEEE/CVF winter conference on applications of computer vision. 2021.
- [4] X. He and L. Deng, “Deep Learning for Image-to-Text Generation: A Technical Overview”, in IEEE Signal Processing Magazine, vol. 34, no. 6, pp. 109-116, Nov. 2017.

- [5] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, “Gradient-based learning applied to document recognition”. Proceedings of the IEEE. 1998.
- [6] Namysl, Marcin, and Konya, Iuliu. “Efficient, Lexicon-Free OCR using Deep Learning”. Proceedings of the ICDAR. 2019.
- [7] Sklearn.Cluster.DBSCAN. (s/f). Scikit-Learn. Recuperado el 25 de abril de 2023, de <https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html>
- [8] Liaw, A., Wiener, M. (2002). Classification and regression by randomForest. R news, 2(3), 18-22.
- [9] de Campos, Teofilo and Babu, Bodla and Varma, Manik. “Character Recognition in Natural Images.” Proceedings of the 4th International Conference on Computer Vision Theory and Applications. VISAPP 2009.
- [10] Deng, Jia and Dong, Wei and Socher, Richard and Li, Li-Jia and Li, Kai and Fei-Fei, Li. “Imagenet: A large-scale hierarchical image database.” IEEE conference on computer vision and pattern recognition. 2009.
- [11] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization” 2014.
- [12] Kaiming He and Xiangyu Zhang and Shaoqing Ren and Jian Sun. “Deep Residual Learning for Image Recognition” 2015.
- [13] Lee, Alex W. C. and Chung, Jonathan and Lee, Marco. “GNHK: A Dataset for English Handwriting in the Wild.” ICDAR International Conference of Document Analysis and Recognition. 2021.
- [14] Ringnér, M. “What is principal component analysis?”. Nat Biotechnol 26, 303–304 (2008). <https://doi.org/10.1038/nbt0308-303>