Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Automatización de la Verificación de Recogida de Firmas para Proposiciones de Ley mediante Inteligencia Artificial

Automation of the Verification of Signature Collection for Law Proposals through Artificial Intelligence

Ismael Martín Herera

La Laguna, 17 de junio de 2024

Dña. **Elena Sánchez Nielsen**, profesora Titular de Universidad adscrita al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutora

C E R T I F I C A (N)

Que la presente memoria titulada:

”*Automatización de la Verificación de Recogida de Firmas para Proposiciones de Ley mediante Inteligencia Artificial*”

ha sido realizada bajo su dirección por D. **Ismael Martín Herrera**.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 17 de junio de 2024

Agradecimientos

Quiero agradecer a mi tutora, Elena Sánchez Nielsen, por su compromiso y apoyo en la realización del trabajo. Y también a mi familia y amigos que me han apoyado durante toda la carrera haciendo más fácil el camino.

Licencia

Resumen

Las iniciativas legislativas populares se consideran un mecanismo de democracia directa, a través del cual se confiere a los ciudadanos el derecho de realizar propuestas de ley al poder legislativo. Este Trabajo de Fin de Grado tiene como objetivo el desarrollo de una aplicación para sustituir al sistema tradicional de revisión y verificación de los datos manuscritos en las recogidas de firmas para proposiciones de ley en el contexto del Parlamento de Canarias. En la actualidad existen algunas soluciones para realizar el reconocimiento óptico de caracteres (OCR) manuscritos, algunas de ellas son de software libre y otras conllevan un coste. Sin embargo, en este trabajo se ha realizado una implementación en Python adaptada al caso de uso y con tres técnicas de inteligencia artificial (IA) para la fase de OCR, los transformers, las máquinas de soporte vectorial (SVM) y redes neuronales convolucionales (CNN). Se han evaluado los resultados obtenidos con cada una de las técnicas de forma individual en la aplicación desarrollada. La evaluación concluyó que los transformers son la técnica de IA que proporciona los mejores resultados en el reconocimiento óptico de caracteres para este caso específico. Por lo que esto sugiere que los transformers son efectivos para abordar los desafíos asociados con la verificación de datos manuscritos, en este caso asociados a las recogidas de firmas.

Palabras clave: Iniciativas legislativas populares, reconocimiento óptico de caracteres, inteligencia artificial, *transformers*, redes neuronales convolucionales, máquinas de soporte vectorial

Abstract

Popular legislative initiatives are considered a mechanism of direct democracy, through which citizens are granted the right to propose laws to the legislative power. This Bachelor's Thesis aims to develop an application to replace the traditional system of reviewing and verifying handwritten data in signature collections for legislative proposals in the context of the Parliament of the Canary Islands. Currently, there are some solutions for performing handwritten optical character recognition (OCR), some of which are open-source while others incur a cost. However, this work has implemented a Python-based solution tailored to the use case and utilizing three artificial intelligence (AI) techniques for the OCR phase: transformers, support vector machines (SVM), and convolutional neural networks (CNN). The results obtained with each technique were evaluated individually in the developed application. The evaluation concluded that transformers are the AI technique that provides the best results in optical character recognition for this specific case. This suggests that transformers are effective in addressing the challenges associated with verifying handwritten data, in this case related to signature collections.

Keywords: Popular legislative initiatives, optical character recognition, artifitial intelligence, transformers, convolutional neural networks, support vector machines

Índice general

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

Las proposiciones de ley son iniciativas legislativas que se presentan en los órganos legislativos competentes (Congreso, Senado, Parlamentos Autonómicos). Dichas proposiciones de ley pueden ser propuestas por los grupos parlamentarios, así como los ciudadanos a través de una iniciativa legislativa popular (ILP).

Las iniciativas legislativas populares se consideran un mecanismo de democracia directa, a través del cual se confiere a los ciudadanos el derecho de realizar propuestas de ley al poder legislativo. A nivel nacional, la ley orgánica 3/1984, de 26 de marzo, regula las formas de ejercicio y requisitos de la iniciativa popular para la presentación de proposiciones de ley, requiriendo un mínimo de 500.000 firmas acreditadas.

En el caso concreto del Parlamento de Canarias, las proposiciones de ley presentadas como iniciativas populares deben ser suscritas por las firmas de al menos 15.000 personas, o del 50% del electorado de una circunscripción insular en aquellas iniciativas cuyo contenido afecte en exclusiva a una isla.

1.1. Antecedentes y estado actual

En el Parlamento de Canarias, las proposiciones de ley de iniciativa popular deben ser examinadas por la Mesa del Parlamento con la finalidad de que ésta determine el cumplimiento de los requisitos legalmente establecidos. Su procedimiento se ajustará a lo establecido por el Reglamento del Parlamento de Canarias [1].

Uno de los requisitos necesarios para la admisibilidad de una iniciativa popular como proposición de ley es la verificación y certificación de la condición de elector firmante. En el caso de recogida de firmas en papel, se deberá realizar la comprobación de los datos obligatorios.

Actualmente, en el Parlamento de Canarias, el procedimiento para la verificación y certificación de electores firmantes se realiza de forma manual, conllevando un trabajo laborioso.

1.2. Objetivos

El objetivo del presente Trabajo Fin de Grado se centra en el análisis y desarrollo de un aplicativo para reemplazar el método manual y tradicional de la verificación y

certificación de los datos manuscritos de los electores firmantes en la recogida de firmas en papel.

La aplicación basa su funcionamiento en el uso de técnicas de inteligencia artificial para la realización del reconocimiento óptico de los caracteres del número del documento nacional de identidad de los firmantes, así como la detección de la presencia de la firma correspondiente.

Como caso de uso, se planteará y evaluará dicha propuesta dentro del contexto del Parlamento de Canarias, analizando la aplicación en sus tres versiones, cada una de ellas con una técnica diferente en la etapa de OCR.

1.3. Estructura del trabajo

El siguiente Trabajo Fin de Grado se ha estructurado comenzando por un análisis y descripción de los aspectos más relevantes sobre el contexto actual de las herramientas OCR existentes. Y presentando algunas de las más relevantes entre las que se enmarcan las técnicas de inteligencia artificial empleadas en la aplicación. A continuación, se describe el desarrollo de la aplicación y las características de la misma. Finalmente, se lleva a cabo una evaluación de las tasas de éxito para concluir con la propuesta que se considera más adecuada, en base a los resultados.

Capítulo 2

Estado del arte

2.1. Descripción de OCR, ejemplos y problemas

El reconocimiento óptico de caracteres (OCR) es el proceso mediante el que se convierte el texto de una imagen, ya sea escrito a mano o impreso, en una cadena de caracteres ASCII legible por una máquina. Para ello, existen múltiples técnicas y algoritmos para implementarlo.

El proceso se puede dividir, generalmente, en las siguientes cinco etapas [11] [20] :

1. **Preprocesado**

   En esta fase, se pretende preparar la imagen capturada, para eliminar cualquier tipo de ruido o imperfección. Además, en función de la técnica a utilizar, se podría requerir que la imagen esté en escala de grises, por ejemplo, o que los caracteres tengan un tamaño y número de píxeles determinado.

2. **Segmentación**

   Esta fase es una de las más complejas, y permite la descomposición de un texto en diferentes entidades lógicas. En esta etapa se localizan los caracteres, los espacios en blanco y los finales de línea. La misma se divide en dos pasos, en primer lugar, se detectan los renglones que conforman el texto original. Y en segundo lugar, se aíslan los caracteres de los renglones, cada uno en un rectángulo, esto incluye también los espacios en blanco.

3. **Extracción de características**

   En esta fase se extrae la información más relevante del texto y sus características para, a continuación, poder reconocer todos los caracteres. La elección de las características se realiza teniendo en cuenta que las mismas cumplan con lo siguiente: son diferentes unas de otras, son independientes, y que no sean demasiadas. Con lo que se debería obtener un conjunto estable y representativo de características.

4. **Reconocimiento**

   Teniendo en cuenta las características obtenidas en la fase anterior, en esta etapa se utilizarán una serie de técnicas para reconocer los distintos caracteres.

5. **Post-procesado**

En esta última fase se trata de mejorar el reconocimiento, realizando un análisis del texto reconocido, y se corrigen posibles errores ortográficos, gramaticales o de validación de formatos.

El uso del reconocimiento óptico de caracteres se está viendo fuertemente impulsado, debido a la necesidad de digitalización rápida de documentos en muchos ámbitos, por lo que destacan algunas de las posibles aplicaciones:

- **Procesamiento de recibos**

  La tecnología OCR puede facilitar mucho el procesamiento de recibos, puesto que la mayoría son impresos o escritos a mano. Por lo que existe la necesidad de digitalizarlos. Por ejemplo, podemos pensar el caso de que tengamos una serie de facturas en papel de distintas fuentes, y queremos incluirlo en nuestra contabilidad. En este caso, las distintas fuentes siempre nos van a proporcionar esa factura en ese formato, por lo que no podemos cambiarlo, sino adaptarnos a él.

- **Sector bancario**

  El uso de OCR en la banca surge por la necesidad de que los cheques, que generalmente son escritos a mano, puedan ser procesados sin la intervención humana.

- **Sector sanitario**

  En el ámbito de la sanidad, muchas veces los distintos profesionales del área tratan con documentos en papel sobre los pacientes, por lo que resultaría interesante poder extraer toda esa información y adjuntarla a su historia clínica, por ejemplo.

Tal y como se ha comentado previamente, el OCR se puede realizar sobre texto impreso o escrito a mano. Sin embargo, la mayoría de textos se encuentran manuscritos, como es el caso que se plantea en este trabajo. Por lo que, cabe distinguir que existen dos métodos diferenciados [14] para el reconocimiento de escritura a mano. Por un lado, tenemos los métodos en línea, que conllevan el uso de un lápiz digital. En este caso, el reconocimiento se va realizando en tiempo real, mientras el usuario escribe. Con lo que la complejidad es menor y permite tener mayor información, como puede ser la velocidad de escritura del usuario, número de veces que el lápiz tiene contacto con la pantalla o la dirección del mismo. Y por otro lado, tenemos los métodos sin conexión, es decir, reconocer el texto una vez se haya escrito, escaneando el mismo o mediante un foto.

La utilización del OCR ofrece los siguientes beneficios: la **reducción de costes**, **acelerar los flujos de trabajo** o **automatizar el procesamiento de los documentos**, entre otros. Sin embargo, también puede presentar los siguientes problemas:

- Al utilizar *software* para realizar OCR puede ocurrir que **no siempre sea del todo preciso**.
- Pueden ocurrir también **problemas de compatibilidad** con distintos idiomas y fuentes.
- La tecnología OCR puede tener **dificultades para mantener el formato** de un texto, es decir, los saltos de líneas, estilos, tablas, sangrías o gráficos.

- Las herramientas de OCR **depende de la calidad de la imagen**, por lo que en caso de imágenes de baja calidad o en malas condiciones de iluminación se puede dificultar el reconocimiento.

En los siguientes apartados se expondrán las diferentes técnicas con las que contamos, clasificándolas en:

- Técnicas mediante aprendizaje automático.
- Técnicas mediante aprendizaje profundo.

Asimismo, se realizará una comparativa de ventajas y desventajas entre diferentes aproximaciones basadas en *software* libre y comercial.

2.2. Técnicas mediante aprendizaje automático

El aprendizaje automático "es la disciplina dentro de la ciencia de datos que permite que las máquinas aprendan sin ser programadas con reglas específicas. Aplica la estadística para inferir propiedades y otros métodos matemáticos para detectar patrones en los datos y, a partir de ahí, hacer predicciones e incluso tomar decisiones."[17]

Se considera un subconjunto de la inteligencia artificial (IA). El aprendizaje automático usa algoritmos para identificar patrones en los datos, y esos patrones luego se usan para crear un modelo de datos que puede hacer predicciones. Con más experiencia y datos, los resultados del aprendizaje automático son más precisos, de forma muy similar a cómo los humanos mejoran con más práctica. En el área del OCR existen algunas técnicas de clasificación mediante algoritmos de aprendizaje automático, las cuales se presentan a continuación.

2.2.1. Algoritmo *RandomForest*

RandomForest es un algoritmo de aprendizaje automático que se usa generalmente para manejar problemas de clasificación y regresión, por lo que se puede utilizar para tareas de OCR. Es el algoritmo más popular de combinación de modelos y está basado en árboles de decisión. Estos árboles de decisión a diferencia de los tradicionales no disponen de todas las características para realizar cada corte del árbol, sino que se elige un conjunto aleatorio de características. Los cortes en los árboles de decisión consisten en seguir por la rama izquierda o derecha en cada nodo del árbol, en función de si una característica específica está por debajo o por encima de un determinado umbral de confianza. [17]

A continuación, se enumeran una serie de ventajas y desventajas del uso de *RandomForest*, como algoritmo de clasificación aplicado en OCR.

Ventajas:

- Es robusto y capaz de manejar grandes conjuntos de datos.
- Tiene la capacidad de manejar diferentes tipos de características de las imágenes.

- Reduce los sobreajustes que podrían ocurrir si se usara un sólo árbol.
- Es capaz de generalizar a datos no vistos, por lo que es interesante por ejemplo para casos de diferentes estilos de escritura.

Desventajas:

- Complejidad a la hora de interpretar los resultados, en comparación con modelos más simples.
- En conjuntos de datos grandes, el tiempo de entrenamiento puede ser bastante significativo.
- Requiere espacio de almacenamiento, en los casos de necesitar guardar múltiples árboles.

2.2.2. Algoritmo de los k vecinos más próximos

El algoritmo KNN o de los k vecinos más próximo es un método clasificador de aprendizaje supervisado no paramétrico, cuyo criterio de predicción es la proximidad. Aunque también puede ser utilizado para tareas de regresión. El algoritmo comienza seleccionando un número de vecinos (k), que serán los que se tendrán en cuenta para obtener la etiqueta de un punto. Y a continuación, se miden las distancias con los vecinos y se identifican los mismos. Finalmente, al nuevo punto se le asigna como etiqueta el valor que tengan la mayoría de sus vecinos. [9]

Ventajas:

- Es fácil de entender e implementar.
- El algoritmo se adapta a la distribución de los datos.
- Este método no necesita un entrenamiento costoso, por lo que se puede adaptar fácilmente a nuevos datos.

Desventajas:

- kNN puede ser sensible ante conjuntos de datos muy grandes.
- Puede resultar costoso computacionalmente el cálculo de la distancia entre puntos.
- El algoritmo es sensible a datos atípicos del conjunto.

2.2.3. Máquinas de soporte vectorial

Las máquinas de soporte vectorial [12] son un importante conjunto de algoritmos de aprendizaje supervisado que se vienen utilizando para solucionar problemas de regresión y clasificación. La clasificación es precisamente el uso que nos interesa para el caso del reconocimiento de caracteres escritos a mano.

En el caso de este trabajo, una de las herramientas a utilizar, el paquete de *Python Scikit-learn* [4], utiliza precisamente las máquinas de soporte vectorial para realizar la clasificación de los caracteres a reconocer.

Tal y como se ha comentado las máquinas de soporte vectorial, son un método de clasificación y regresión. Este se basa en el uso de hiperplanos de margen máximo, mediante los que se pretende separar clases en un espacio de características. A continuación, se definen dos términos importantes en el ámbito de este método.

- **Margen:** es la distancia perpendicular entre el hiperplano y los puntos más cercanos de cada clase.
- **Vectores de soporte:** son los puntos más cercanos al hiperplano, aquellos que definen el margen. Y por tanto, son los puntos críticos para la determinación del hiperplano de margen máximo.

La tarea de SVM es precisamente formular y resolver un problema de optimización para encontrar el hiperplano que maximiza el margen, pero asegurando que este separa correctamente las clases. Por lo que cada punto de entrenamiento queda clasificado correctamente. Asimismo, cabe destacar que existen dos tipos de clases, las **linealmente separables** y las **linealmente no separables**.

En el caso de las **linealmente separables**, es posible trazar una línea o hiperplano de tal forma que todos los puntos de una clase queden a un lado y los de la otra clase al otro lado, como se muestra en la figura 2.1.

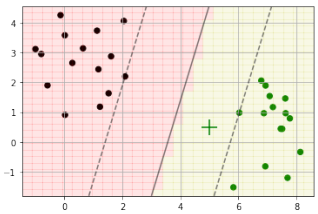


Figura 2.1: Hiperplano de margen máximo separando dos clases. Extraída de [16].

Y por otro lado, respecto a las **linealmente no separables**, al contrario de las anteriores, son aquellas en las que no se puede trazar una línea que separe perfectamente las dos clases. Puesto que, puede ocurrir que las dos clases estén superpuestas o distribuidas de manera que no se puede encontrar dicho plano, tal y como se muestra en la figura 2.2.

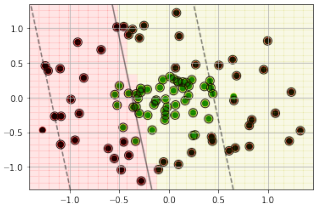


Figura 2.2: Dos clases linealmente no separables. Extraída de [16].

En este caso, lo que se hace es aumentar la dimensionalidad del espacio de características, pasando por ejemplo de un espacio bidimensional a tridimensional, para lo que se puede utilizar la función kernel radial. Tal y como se observa en la figura 2.3.

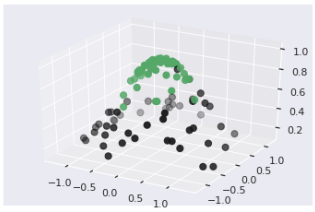


Figura 2.3: Aumento en la dimensionalidad del espacio. Extraída de [16].

Este conjunto de algoritmos presenta las siguiente ventajas y desventajas.

Ventajas:

- Son efectivos en espacios de alta dimensionalidad, aun cuando el número de dimensiones supera el número de muestras.
- Hacen un uso eficiente de la memoria, puesto que utilizan un subconjunto de puntos en la función de decisión.

Desventajas:

- Su eficacia depende del kernel que se utilice.
- Reduce su eficiencia con conjuntos de datos grandes.
- Se debe evitar el sobreentrenamiento, en caso de que el número de características es mayor que el de muestras.
- La frontera de decisión depende directamente de los valores más próximos, aunque sean erróneos.
- Dependen de la escala de datos.

2.3. Técnicas mediante aprendizaje profundo

El aprendizaje profundo o *Deep Learning* [6] en inglés es una rama del aprendizaje automático o *Machine Learning* que se basa en la manera de pensar del ser humano. Se implementa mediante redes neuronales, y su nombre viene del hecho de que estas redes neuronales tienen varias capas de profundidad. Una red neuronal es un modelo computacional compuesto por neuronas artificiales o nodos organizadas en capas interconectadas, como se muestra en la figura 2.4. En este sentido, las redes cuentan, generalmente, con tres tipos de capas.

1. **Capa de entrada**

   Recibe los datos de entrada.

2. **Capas ocultas**

   Son las encargadas de procesar la información recibida de la capa de entrada, y aprender patrones y características relevantes de los datos de entrada.

3. **Capa de salida**

   Produce la salida final de la red tras procesar la información.

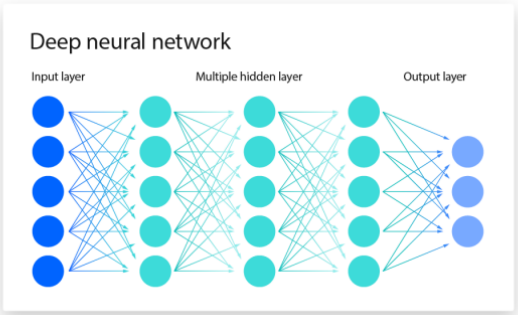


Figura 2.4: Estructura de una red neuronal. Extraída de [13].

En los siguientes apartados se exponen algunas de las redes neuronales más habituales: las redes neuronales **convolucionales** (CNNs), las redes neuronales **recurrentes** (RNNs) y los ***transformers***.

2.3.1. Redes neuronales convolucionales

Las redes neuronales convolucionales [22] [19] [7] o CNN por sus siglas en inglés, son redes neuronales artificiales cuyo diseño se inspira en el funcionamiento del sistema visual del ser humano. Estas están diseñadas para procesar y analizar datos en forma de cuadrícula como las imágenes.

Las CNNs surgen para resolver un problema que existe a la hora de intentar clasificar imágenes mediante redes neuronales tradicionales. Una imagen se suele representar como un vector, y en cada posición del vector se tiene el valor de un píxel que se corresponde con una neurona de la capa de entrada. Además, generalmente, las imágenes se introducen a color, por lo que para cada píxel tenemos tres componentes RGB (Red-Green-Blue). Por tanto, si tenemos un imagen de 64x64 píxeles, por ejemplo, y 3 componentes para cada uno de ellos, tendríamos $64 * 64 * 3 = 12288$ posiciones en el vector, por lo que el coste computacional podría llegar a ser bastante elevado.

Las CNNs introducen una serie de capas diferenciadas. En primer lugar, tenemos las capas convolutivas que aplican como su nombre indica la convolución. Dicha convolución es una operación matemática de productos y sumas que se aplica entre la imagen de entrada y un filtro o kernel. El filtro es una matriz de tamaño más pequeño que la propia imagen, y que se va convolucionando con las distintas regiones de la imagen. Cabe destacar, que el filtro es el mismo en todo el proceso. Asimismo, cabe añadir, que cada neurona oculta sólo está conectada con un pequeño subconjunto de elementos de la imagen de entrada. En segundo lugar, tenemos la capa de *pooling*, que sirve para reducir la dimensión espacial de las representaciones de las capas convolutivas, pero manteniendo las características más importantes. Esta capa lo que hace es agrupar los valores adyacentes y buscar el máximo o la media, con lo que por ejemplo si tenemos cuatro valores adyacentes, ahora pasaríamos a tener uno, por lo que se reduce el tamaño. Y finalmente, tenemos las capas totalmente conectadas, a las cuales se les pasa la imagen

cuya dimensión está reducida tras el *pooling* y que viene a ser como una red neuronal tradicional.

Las ventajas y desventajas de las CNNs son las siguientes [7].

Ventajas:

- Son capaces de aprender automáticamente características en diferentes niveles de abstracción, tales como bordes, texturas y patrones complejos.
- Robustas frente a variaciones en la forma y estilo de los caracteres.
- Las CNNs utilizan parámetros compartidos, por lo que se reduce la cantidad de pesos que deben aprenderse, mejorando en eficiencia computacional.
- La transferencia de aprendizaje permite utilizar redes CNN pre-entrenadas en conjuntos de datos grandes.
- Se pueden adaptar fácilmente a diferentes tareas de visión por computador, como por ejemplo el caso de este trabajo, el OCR.

Desventajas:

- Necesitan gran conjunto de datos etiquetados.
- La profundidad de las redes CNN influye en la interpretabilidad de los resultados, puede resultar complejo cuanto más profundas es.
- El entrenamiento y la inferencias de las redes pueden requerir bastantes recursos computacionales.

2.3.2. Redes neuronales recurrentes y *Long Short-Term Memory*

Las redes neuronales recurrentes [19] están diseñadas para el procesamiento de datos secuenciales y que por tanto tienen una relación de orden entre los mismos. Es por ello que uno de los usos más extendidos podría ser el procesamiento del lenguaje natural. En adición a ello, las RNN tratan de solucionar dos problemas de las redes neuronales tradicionales. Por un lado, tenemos el inconveniente de que en el caso del reconocimiento de una oración, por ejemplo, deberíamos tener tantas entradas como palabras tiene la oración. Sin embargo, no todas las oraciones tienen el mismo número de palabras. Y por otro lado, en el caso de las redes neuronales tradicionales no se comparten parámetros, lo cuál es sumamente importante en algunos casos que se requiere del contexto previo. En este sentido, las RNN incorporan un "estado oculto"que tiene información sobre lo que se ha procesado previamente. Además, con el fin de mejorar la robustez de las mismas y debido a la memoria de corto plazo que tienen, se incorporan también celdas LSTM (Long Short-Term Memory). Este nuevo tipo de celda tiene en primer lugar, una llamada compuerta del olvido, que en el caso del reconocimiento de texto, evita palabras como pueden ser ”de” o ”que” que no aportan gran información. En segundo lugar, incorporan también una compuerta de entrada que indica que parámetros deben actualizarse en el estado de la celda. Y finalmente, tenemos la compuerta de salida que mediante el estado de la celda permite indicar cuál será el nuevo estado oculto de la red.

La redes neuronales recurrentes tienen una serie de ventajas y desventajas, relacionadas a continuación.

Ventajas:

- Son efectivas en el manejo de dependencia temporal entre los datos.
- Son flexibles respecto a longitud de las entradas.

Desventajas:

- Las RNN tradicionales sin LSTM pueden tener problemas de desvanecimiento.
- Puede ocurrir que la inferencia y el entrenamiento sean lentos por su naturaleza secuencial.
- A pesar del uso de LSTM o GRU, aún sigue existiendo un problema con las secuencias muy largas.
- Dificultad para manejar datos con patrones irregulares.

2.3.3. *Transformers*

Los *transformers* [21] son una arquitectura de red neuronal que se introdujo principalmente para el procesamiento de lenguaje natural, y para solucionar los problemas de falta de memoria de las redes neuronales recurrentes. En este sentido, en los *transformers* se utilizan mecanismos de atención que indican a la red neuronal en qué parte de la entrada resulta más interesante fijarse.

La arquitectura de esta red para el procesamiento de lenguaje natural se basa en un bloque codificador y un bloque decodificador, tal y como se muestra en la figura 2.5. La entrada al *transformer* es una secuencia de representaciones vectoriales de palabras o *embeddings*. Esta entrada pasa a un mecanismo de atención en el que se calculan una serie de puntuajes, para valorar la relevancia de cada posición respecto a las demás. A continuación, se pasa a realizar una operación de *feedforward* que realiza transformaciones matemáticas que introducen comportamiento no lineal al modelo, que le permiten aprender y representar patrones más complejos. Además de lo anterior, a la entrada se añade también un bloque de representación posicional, para las posiciones de las palabras. Posteriormente, una vez realizadas las citadas operaciones en el codificador, la salida del mismo pasa al decodificador, para generar la salida correspondiente. Cabe destacar que a diferencia de la redes recurrentes en los *transformers* cada posición de la frecuencia se procesa de manera independiente, lo cuál junto al mecanismo de atención permite evitar las pérdidas de memoria a largo plazo.

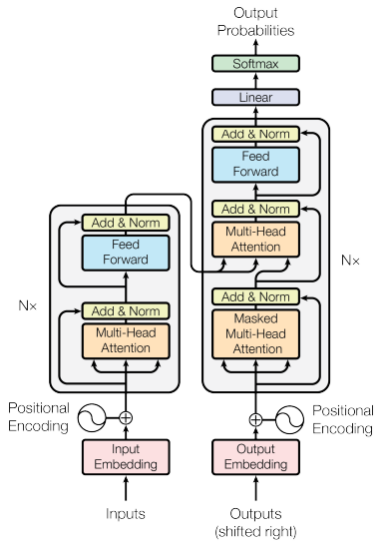


Figura 2.5: Arquitectura de los *Transformers* propuesta por Google. Extraída de [21].

Sin embargo, en el caso de este trabajo queremos aplicar los *transformers* para una tarea de visión por computador, el OCR. Por lo que es importante remarcar que, a pesar de que esta arquitectura fue principalmente desarrollada para el ámbito del NPL *(Natural Language Processing)* se ha visto que también tiene utilidad para tareas de visión por computador. En ese caso lo que se hace es dividir la imagen en distintas regiones, con lo que la entrada pasaría a ser una secuencia con las distintas regiones de la imagen. Y ahora el posicionamiento habitual que se utiliza en los *transformers* se refiere a la posición espacial de la imagen, lo cuál es crucial para la VC.

Ventajas:

- Captura relaciones a largo plazo.
- Procesa la información de manera paralela, agilizando el entrenamiento y la inferencia.
- Son escalables, pudiendo manejar secuencias de diferentes longitudes sin tener que realizar grandes cambios.
- Facilidad para aprender patrones complejos y adaptación a diferentes estilos de texto, crucial en el ámbito del OCR.
- Los bloque de atención y multi-atención permiten que el modelo se centre en diferentes partes de la imagen.

Desventajas:

- Requiere grandes conjuntos de datos y recursos.
- La implementación puede resultar compleja y requerir de amplios conocimientos en el ámbito del aprendizaje profundo.
- Complejidad en la interpretación de los resultados por la complejidad del propio modelo.

2.4. Aproximaciones basadas en software libre y comercial

El reconocimiento de caracteres escritos a mano se puede realizar además con una serie de herramientas tanto **gratuitas** como **comerciales**. En primer lugar, respecto a las de uso libre cabe nombrar las siguientes:

- OpenCV
- Tesseract
- EasyOCR
- Python

Cada una de estas herramientas presenta una serie de ventajas y de desventajas que se pueden observar, a continuación, en las tablas 2.1, 2.2, 2.3, 2.4, 2.5 y 2.6.

OpenCV	
Ventajas	**Desventajas**
Gratuita	Funcionalidades para análisis de imagen y vídeos más generales no específicos para reconocimiento óptico de caracteres
Admite múltiples lenguajes (C++, Python, Java y MatLab)	Curva de aprendizaje pronunciada
Alta precisión y rendimiento en tareas de visión por computadora	
Comunidad grande y activa	

Tabla 2.1: Ventajas y desventajas de la herramienta OpenCV

Tesseract	
Ventajas	**Desventajas**
Gratuita	Necesita bastantes recursos y tiempo
Motor específico de reconocimiento OCR	Su rendimiento en reconocimiento de texto escrito a mano es variable
Mayor facilidad de aprendizaje	Funciona mejor con documentos con formato sencillo
Alta precisión en el reconocimiento de texto	
Admite múltiples idiomas	

Tabla 2.2: Ventajas y desventajas de la herramienta Tesseract

EasyOCR	
Ventajas	**Desventajas**
Soporta más de 80 idiomas	No soporta detección de escritura a mano por ahora
Modelo ligero con buen rendimiento	Proporciona resultados más precisos con textos organizados
Facilidad de uso	

Tabla 2.3: Ventajas y desventajas de la herramienta EasyOCR

Python	
Ventajas	**Desventajas**
Amplia variedad de bibliotecas y motores incluidos como Tesseract o EasyOCR	Requiere conocimientos previos sobre el lenguaje
Comunidad amplia y activa	Requiere conocimientos sobre los módulos a utilizar
	Dependencia de bibliotecas externas
	Limitaciones en documentos complejos

Tabla 2.4: Ventajas y desventajas de la herramienta Python

En segundo lugar, respecto a las herramientas comerciales destacar las siguientes:

- Adobe Acrobat
- Kofax OmniPage Ultimate

Adobe Acrobat	
Ventajas	**Desventajas**
Interfaz intuitiva	Requiere una suscripción
Integración con otros productos de Adobe	Precisión variable en documentos complejos
Amplia compatibilidad en cuanto a tipos de documentos	Puede ser bastante costosa la licencia

Tabla 2.5: Ventajas y desventajas de la herramienta Adobe Acrobat

Kofax OmniPage Ultimate	
Ventajas	**Desventajas**
Alta precisión	Puede ser costosa
Amplia compatibilidad	Curva de aprendizaje pronunciada para usuarios no familiarizados con software OCR

Tabla 2.6: Ventajas y desventajas de la herramienta Kofax OmniPage Ultimate

Capítulo 3

Desarrollo

3.1. Requisitos de la aplicación

La aplicación se ha desarrollado en el lenguaje de programación Python. Desde el punto de vista del *software* requiere tener al menos la versión 3.12 de Python, y la instalación de los *drivers* de NVIDIA, CUDA 11.8 y cuDNN, en caso de disponer de tarjeta gráfica dedicada y compatible. Así como la instalación de un conjunto de dependencias que varían en función de la técnica de inteligencia artificial que se desee utilizar en la fase de reconocimiento óptico de caracteres (OCR). Por otro lado, desde el punto de vista del *hardware* es recomendable disponer, como se ha comentado previamente, de una tarjeta gráfica NVIDIA, puesto que en su defecto se usará la CPU lo que conlleva a una posible demora en la ejecución de la aplicación.

3.2. Descripción de la aplicación

La aplicación desarrollada tiene tres versiones, en función de la técnica de inteligencia artificial que se desea utilizar en la fase de OCR, es decir, extraer de cada imagen la cadena de caracteres correspondiente que en el caso de uso de este trabajo sería un número de DNI seguido de su correspondiente letra de control. Sin embargo, las tres versiones tienen en común toda la arquitectura de la aplicación que se observa en la figura 3.1.

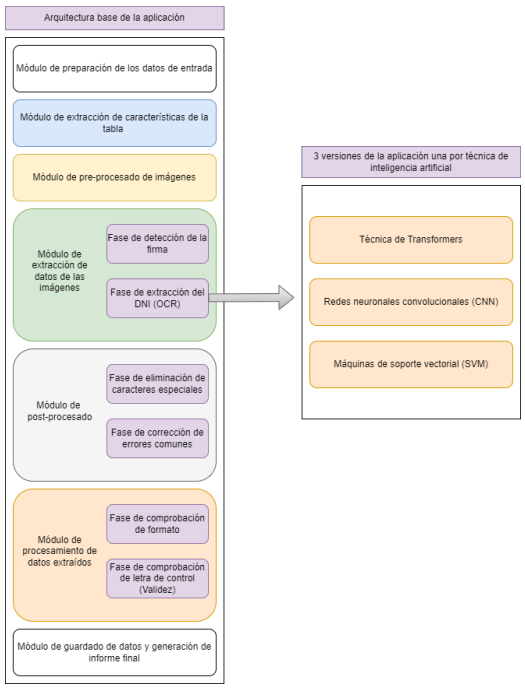


Figura 3.1: Diagrama de la arquitectura de la aplicación

En los siguientes apartados se describen los módulos comunes en la arquitectura de la aplicación.

3.2.1. Módulo de preparación de datos de entrada

Los datos entran en la aplicación en un documento PDF, y en cada una de las páginas de dicho documento hay una tabla siguiendo la plantilla que se describirá en el apartado de datos de entrada. Sin embargo, la aplicación trabaja con imágenes entonces este primer módulo o función es el encargado de convertir en imágenes cada una de las páginas del documento PDF, para luego pasarlas una a una al resto de módulos de la aplicación.

3.2.2. Módulo de extracción de características de la tabla

El módulo de extracción de características de la tabla recibe como entrada la imagen completa de una página del documento PDF. Y es el encargado de detectar en dicha imagen las filas y columnas de la tabla, y devolver las cuatro coordenadas de cada fila y columna. Con estas coordenadas se calculan las posiciones de cada una de las celdas de las dos últimas columnas de la tabla, que al ser siempre la misma plantilla, van a ser [DNI, Firma], en este mismo orden. A continuación, se va iterando fila por fila y dadas las coordenadas de las dos celdas para la fila concreta se obtienen dos imágenes como recorte de la imagen original de la página, correspondientes al DNI y la firma. Estas dos imágenes se pasan al siguiente módulo para continuar el análisis.

La extracción de las filas y columnas de la tabla se realiza mediante la técnica de inteligencia artificial *Transformers*, y se hace igual en las tres versiones de la aplicación. Para ello se emplea un modelo específico para esta tarea [5]. El modelo devuelve las etiquetas de cada uno de los elementos detectados, las coordenadas de las cajas delimitadoras que los rodean y el índice de confianza asociado a cada uno de los objetos detectados, en este

caso, filas y columnas, y que se puede observar en la imagen 3.2. También devuelve dos clases, por un lado ’1’ se corresponde con las filas y ’2’ con las columnas.

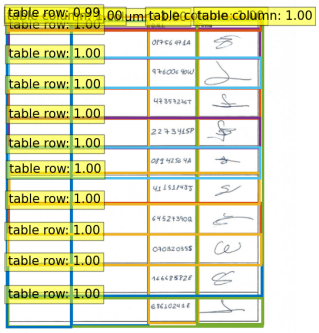


Figura 3.2: Ejemplo de detección de filas y columnas en una tabla por el modelo de *transformers*.

3.2.3. Módulo de pre-procesado de imágenes

La entrada a este módulo serían dos imágenes, una correspondiente a la celda en la que debería haber un DNI y otra de la correspondiente a la firma. Con el fin de mejorar la etapa de OCR para extraer los caracteres del DNI se realizan una serie de operaciones sobre la imagen. Cabe resaltar que el pre-procesado no es exactamente igual en las tres versiones.

En primer lugar, en el caso de la versión con la **técnica de *transformers*** para el OCR a la imagen no se le aplica ningún pre-procesado sólo se recorta un poco los bordes y se le pasa al modelo. Aunque, cabe destacar que el propio modelo tiene incorporado el pre-procesado de la imagen por lo que no es necesario realizar ninguna operación más.

En segundo lugar, en la versión con **máquinas de soporte vectorial (SVM)** la imagen pasa por una fase de detección de los caracteres que hay en la misma, para luego recortarlos y así obtener imágenes individuales de cada carácter. Esto se justifica puesto que en la fase de OCR que se describirá más adelante las máquinas de soporte vectorial trabajan con caracteres individuales, clasificando según la clase correspondiente. Además, a estas imágenes de caracteres separados, también se les aplica una operación de redimensionamiento para que todas tengan 8x8 píxeles.

Y por último, para el caso de la variante con **redes neuronales convolucionales**, también se realiza el recorte en imágenes individuales de caracteres por el mismo motivo. Y además, a cada una de ellas, se le aplican las siguiente operaciones: redimensionado a 128x128 píxeles, binarización, operación *not* sobre cada píxel, filtro Gausiano, redimensionado a 28x28 píxeles con interpolación bicúbica y la conversión a un tensor. Todos estos pasos se han llevado a cabo con el objetivo de que las imágenes que se pasan a la red queden similares a la descritas en este artículo sobre el conjunto de datos EMNIST [8], puesto que el entrenamiento se realizó precisamente con las imágenes de dicho *dataset*.

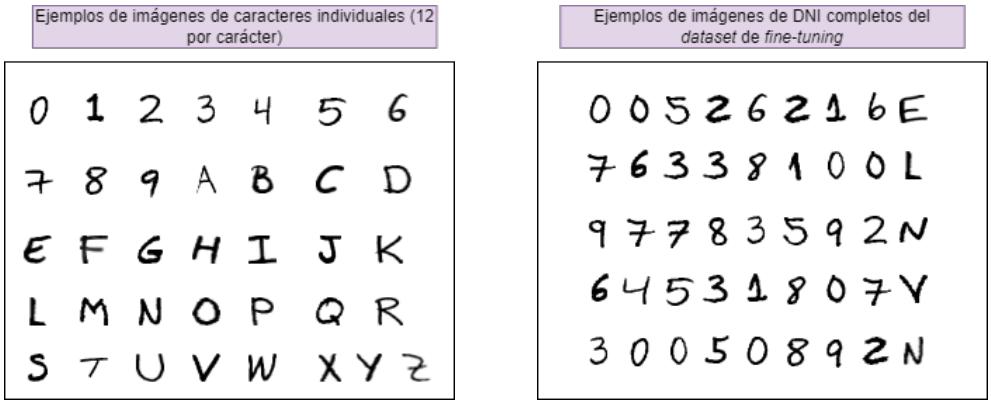
3.2.4. Módulo de extracción de datos de las imágenes

Esta parte de la aplicación es en la que se encuentra la principal diferencia entre las tres versiones de la misma. Puesto que para realizar el reconocimiento óptico de caracteres se han empleado tres técnicas diferentes de inteligencia artificial.

Reconocimiento óptico de caracteres con *transformers*

El reconocimiento óptico de caracteres mediante *transformers* se realiza empleando el modelo pre-entrenado [15]. A este modelo se le proporcionan como entrada los valores de los píxeles de la imagen devueltos por el procesador de imágenes correspondiente. Y, a continuación, obtenemos el texto extraído para el correspondiente DNI.

El modelo pre-entrenado no se ajusta exactamente al tipo de datos con los que trabaja la aplicación, por lo que para mejorar su desempeño se ha llevado a cabo un ajuste del mismo (*fine-tunning*). Este ajuste ha consistido en entrenar el modelo con un *dataset* de imágenes, de DNI completos, generadas sintéticamente de manera similar a como se describe en el apartado de generación de datos de entrada de esta memoria. Las imágenes de los DNI completos, similares a los que se muestran en la imagen 3.3b, se generan a partir de un pequeño *dataset* de imágenes de caracteres individuales en formato PNG, con 12 imágenes por carácter que es una selección del *dataset* EMNIST, que se muestra en la imagen 3.3a.



(a) Ejemplos de imágenes de caracteres individuales.

(b) Ejemplos de imágenes de DNI completos para *fine-tuning*.

Figura 3.3: Imágenes de ejemplo de los *datasets* para el *fine-tuning*

Reconocimiento óptico de caracteres con máquinas de soporte vectorial

El reconocimiento óptico de caracteres con máquinas de soporte vectorial (SVM) requiere que las imágenes de los caracteres sean procesadas de una en una. Por lo que a partir de la imagen original del DNI pasa por una fase de detección de los caracteres mediante el uso de OpenCV y un posterior recorte de la misma en tantos caracteres como se hayan detectado. Estas imágenes de caracteres individuales son cargadas al modelo correspondiente en función de si la posición de la misma correspondería a un número o a la letra. Para ello se comprueba cuántos caracteres se han detectado y se van pasando hasta la posición ocho, si los hubiera, al modelo de números, y la imagen en la posición novena al modelo de letras, puesto que se entiende que es el orden lógico de los DNI, obviando todas las imágenes adicionales a partir de la posición novena.

El uso de las máquinas de soporte vectorial para el OCR requiere el entrenamiento de las mismas por ello para este trabajo se han entrenado dos modelos, uno para letras y otro para números, por separado, para evitar confusiones entre letras y números en la fase de reconocimiento. El entrenamiento se ha realizado con las imágenes del *dataset* de EMNIST completo [8] redimensionadas a 8x8 píxeles. Para el caso de las letras haciendo

uso del *split* `'letters'` y para los números el *split* `'mnist'`.

Reconocimiento óptico de caracteres con redes neuronales convolucionales

El reconocimiento óptico de caracteres con redes neuronales convolucionales, al igual que las SVM, también requieren que se pasen las imágenes de los caracteres de una en una. En este sentido, como en el caso anterior también se realiza una fase de detección de caracteres mediante OpenCV y el posterior recorte en varias imágenes. Asimismo, y de manera similar a las máquinas de soporte vectorial se sigue la misma lógica, pasando las ocho primeras imágenes, que se entienden de un carácter cada una, a la red neuronal CNN entrenada con dígitos del 0 al 9. Y la imagen correspondiente al carácter en la novena posición se le pasa la red neuronal CNN entrenada con letras.

El entrenamiento de ambas redes neuronales convolucionales lo he realizado también con los *split* `'letters'` y `'mnist'` del conjunto de datos de EMNIST. La arquitectura definida para ambas redes es simple y está compuesta en este orden por dos capas convolucionales, encargadas de extraer las características de la imagen. Y seguidas por capas *fully connected* para la clasificación. Asimismo, se emplean las técnicas de *max pooling* y *dropout* que ayudan a reducir la dimensionalidad y prevenir el sobreajuste, respectivamente. Finalmente, la salida de la red se obtiene mediante una función *softmax* que proporciona probabilidades de pertenencia a cada clase para la imagen.

Detección de la firma

La detección de la firma en la imagen de la celda, se realiza mediante un modelo pre-entrenado de *Transformers* de tipo *zero-shot* para la clasificación de imágenes [23], en este caso para determinar si hay o no firma en la correspondiente celda, estableciendo las clases *signature* y *empty*.

3.2.5. Módulo de post-procesado de datos

El módulo de post-procesado de los datos, recibe como entrada los DNI extraídos, y se justifica porque que en ciertas ocasiones los caracteres extraídos por el modelo no son los correctos o introduce algunos adicionales.

En primer lugar, tenemos la fase encargada de eliminar algunos caracteres especiales (puntos, guiones, comas y espacios en blanco) y que, generalmente, no están en la imagen del DNI, por lo que son introducidos por el propio modelo, concretamente el modelo original de *transformers* sin ajuste fino (*fine-tuning*). Asimismo, también se encarga de pasar a mayúsculas las letras de los distintos DNI extraídos.

En segundo lugar, se aplica una fase de corrección de errores comunes en la extracción de los dígitos y la letra de los DNI. En esta etapa se analizan los caracteres de los DNI extraídos y se comprueba según la posición el tipo de carácter que corresponde (número o letra). Para ello, se comprueba primero si el tamaño del DNI predicho es el correcto. Y, a continuación, se van comprobando las distintas posiciones, si debería estar un número y hay una letra se comprueba si el error está en el diccionario predefinido de errores comunes. Y para el caso de la posición en la que debería haber una letra se comprueba también en otro diccionario si se podría corregir el error. Esta fase se justifica para tratar de mejorar los resultados extraídos, puesto que al haber realizado las pruebas con datos

que son correctos y válidos se ha percibido que el modelo confunde algunas letras con números y viceversa y se ha tratado de corregir, sin embargo son muy pocos los casos contemplados para corregir.

3.2.6. Módulo de procesamiento de datos extraídos

El módulo para el procesamiento de datos engloba la lógica una vez que ya se tiene el DNI extraído y post-procesado, y que la firma también se ha analizado. En este módulo lo que se hace, en primer lugar, es comprobar si el formato del DNI extraído es el correcto y cumple con la siguiente expresión regular `^\d{8}[A-z]$` (ocho números seguidos de una letra). En segundo lugar, una vez el formato se comprueba, si es correcto, se pasa a validar el DNI, que consiste en comprobar si la letra de control es la que corresponde a los ocho dígitos del DNI. Para lo que se sigue el algoritmo de DNI español establecido por el Gobierno de España [10].

3.2.7. Módulo de guardado de datos y generación de informes

El módulo de guardado de datos y generación de informes está presente durante toda la ejecución de la aplicación, puesto que va recolectando toda la información que se va generando durante el análisis y además apoya a todo el proceso. Al guardar toda la información permite también determinar si un DNI está repetido o no durante el análisis evitando continuar si el DNI extraído ya se hubiera analizado previamente.

Esta parte de la aplicación se sirve principalmente de dos clases que están relacionadas, para guardar toda la información tanto del análisis a nivel global como de cada firmante analizado. En la figura 3.4 se puede observar un diagrama de las citadas clases.

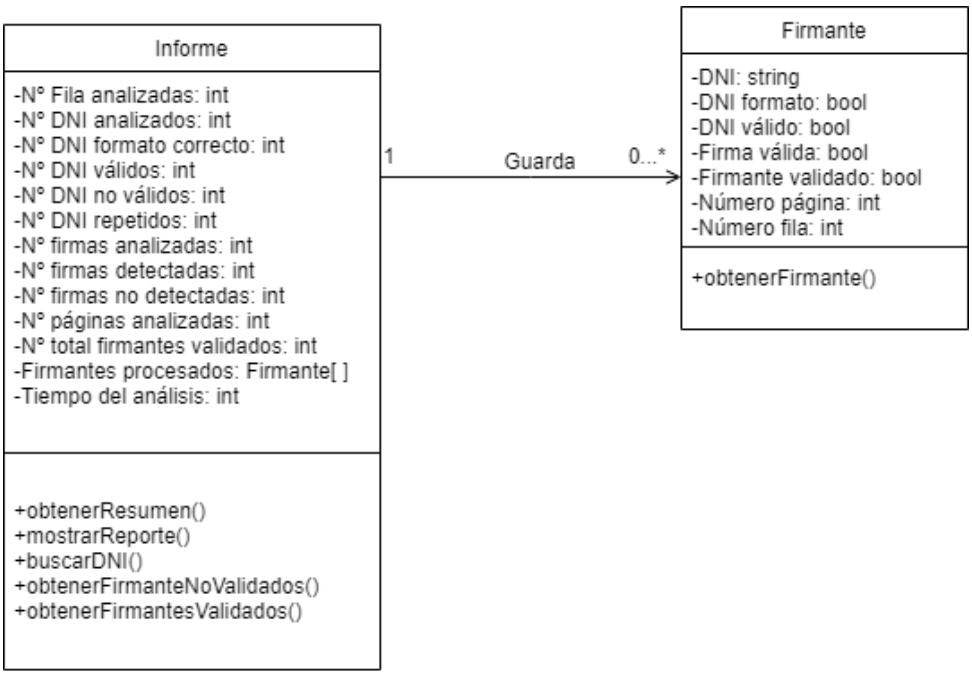
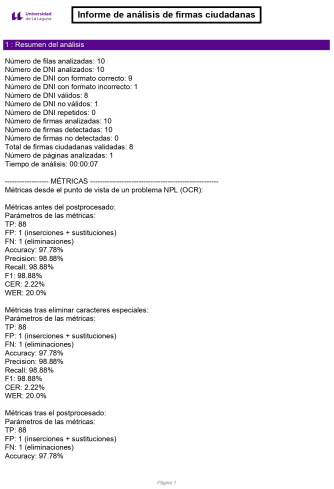


Figura 3.4: Estructura de clases para el guardado de la información del análisis

Se crea una instancia de la clase informe encargada de guardar toda la información global del análisis. En este sentido, de entre los diversos datos que guarda también alma-

cena un *array* de objetos de la clase firmante. Estos objetos guardan toda la información de cada una de las firmas analizadas, por lo que cada vez que se examinan los datos de un firmante se crea una nueva instancia de esta clase y se almacena en la clase informe.

Esta estructura de clases está apoyada además por una tercera clase independiente que es la encargada de generar el documento PDF con el informe, como el que se muestra en las figuras 3.5a, 3.5b, 3.5c. Y que centraliza el proceso apoyándose de la dependencia correspondiente. Sin embargo, esta clase se sirve completamente de toda la información guardada en la instancia de informe y todas las instancias de firmante que correspondan.


Informe de análisis de firmas ciudadanas

1 - Resumen del análisis

Número de filas analizadas: 10
Número de DNI analizados: 10
Número de DNI con formato correcto: 9
Número de DNI con formato incorrecto: 1
Número de DNI válidos: 8
Número de DNI no válidos: 1
Número de DNI repetidos: 0
Número de firmas analizadas: 10
Número de firmas detectadas: 10
Número de firmas no detectadas: 0
Total de firmas ciudadanas validadas: 8
Número de páginas analizadas: 1
Tiempo de análisis: 00:00:07

---------- MÉTRICAS ----------
Métricas desde el punto de vista de un problema NPL (OCR):

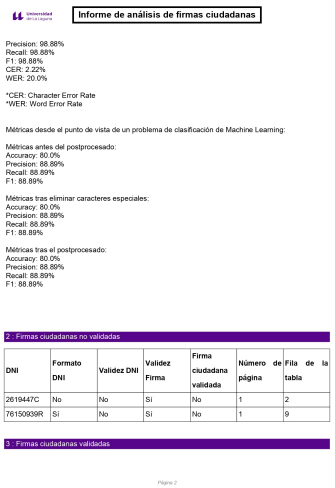
Métricas antes del postprocesado:
Parámetros de las métricas:
TP: 88
FP: 1 (inserciones + sustituciones)
FN: 1 (eliminaciones)
Accuracy: 97.78%
Precision: 98.88%
Recall: 98.88%
F1: 98.88%
CER: 2.22%
WER: 20.0%

Métricas tras eliminar caracteres especiales:
Parámetros de las métricas:
TP: 88
FP: 1 (inserciones + sustituciones)
FN: 1 (eliminaciones)
Accuracy: 97.78%
Precision: 98.88%
Recall: 98.88%
F1: 98.88%
CER: 2.22%
WER: 20.0%

Métricas tras el postprocesado:
Parámetros de las métricas:
TP: 88
FP: 1 (inserciones + sustituciones)
FN: 1 (eliminaciones)
Accuracy: 97.78%

Página 1

(a) Página 1, ejemplo de informe


Informe de análisis de firmas ciudadanas

Precision: 98.88%
Recall: 98.88%
F1: 98.88%
CER: 2.22%
WER: 20.0%

*CER: Character Error Rate
*WER: Word Error Rate

Métricas desde el punto de vista de un problema de clasificación de Machine Learning:

Métricas antes del postprocesado:
Accuracy: 80.0%
Precision: 88.89%
Recall: 88.89%
F1: 88.89%

Métricas tras eliminar caracteres especiales:
Accuracy: 80.0%
Precision: 88.89%
Recall: 88.89%
F1: 88.89%

Métricas tras el postprocesado:
Accuracy: 80.0%
Precision: 88.89%
Recall: 88.89%
F1: 88.89%

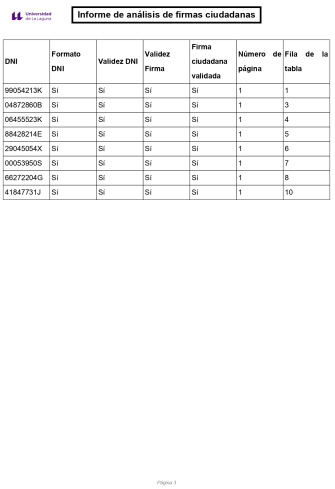
2 - Firmas ciudadanas no validadas

DNI	Formato DNI	Validez DNI	Validez Firma	Firma ciudadana validada	Número de página	Fila de la tabla
2619447C	No	No	Si	No	1	2
76150939R	Si	No	Si	No	1	9

3 - Firmas ciudadanas validadas

Página 2

(b) Página 2, ejemplo de informe


Informe de análisis de firmas ciudadanas

DNI	Formato DNI	Validez DNI	Validez Firma	Firma ciudadana validada	Número de página	Fila de la tabla
99054213K	Si	Si	Si	Si	1	1
04872860B	Si	Si	Si	Si	1	3
06455523K	Si	Si	Si	Si	1	4
88428214E	Si	Si	Si	Si	1	5
29045054X	Si	Si	Si	Si	1	6
00053950S	Si	Si	Si	Si	1	7
66272204G	Si	Si	Si	Si	1	8
41847731J	Si	Si	Si	Si	1	10

Página 3

(c) Página 3, ejemplo de informe

Figura 3.5: Páginas de ejemplo de un informe generado

3.3. Datos de entrada

La aplicación recibe como entrada de datos un documento PDF, siguiendo la plantilla proporcionada por el Parlamento de Canarias, como la que se observa en la figura 3.6. En cada página, hay una tabla con diez filas que corresponden con diez firmantes, y en las que se recogen sus datos.

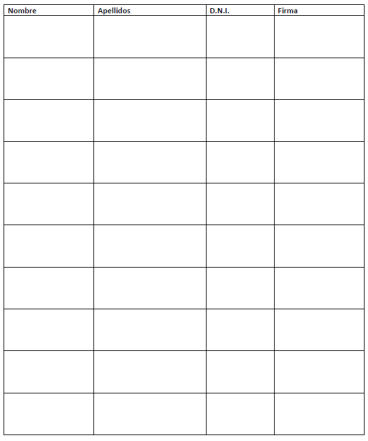

Nombre	Apellidos	D.N.I.	Firma

Figura 3.6: Plantilla para la entrada de datos

Para el desarrollo y evaluación de la aplicación, nos limitaremos al DNI y la firma. Estos datos hay que generarlos, por lo que cabe distinguir dos posibles fuentes para los mismos. Por un lado, tenemos la generación de datos de manera manual, es decir, pedir a distintas personas que anoten un DNI aleatorio y cualquier firma, evitando recolectar datos reales que puedan relacionar a dichas personas. Y por otro lado, tenemos la generación de datos manuscritos de manera sintética, es decir, que no son realmente manuales. Principalmente se ha optado por la generación sintética de los datos porque permiten generar una mayor cantidad de datos, se ha creado un *script* para la realización de esta tarea, cuyo funcionamiento se describe en el siguiente apartado.

3.3.1. Generación sintética de los datos manuscritos

La generación sintética de los datos tal y como se ha planteado para este proyecto supone el uso de unos conjuntos de datos de caracteres individuales manuscritos. En primer lugar, para el caso de las letras de DNI manuscritas se emplea el *dataset* [3] y cuya fuente es EMNIST, que contiene en un fichero CSV la representación de los píxeles de imágenes de letras individuales de 28x28 y en escala de grises. En segundo lugar, respecto a los dígitos del 0-9 se emplea el *dataset* [2] cuyo origen es también EMNIST, y que contiene otro fichero con extensión CSV y con las imágenes en el mismo formato. En ambos casos y mediante un *script* se ha guardado en una estructura de carpetas las imágenes con extensión PNG a partir de los citados ficheros en formato CSV. Esto con el fin de seleccionar sólo algunas de las imágenes para cada carácter, 12 por carácter, puesto que muchas de ellas son incluso ilegibles para el ser humano. Como resultado, el conjunto de datos a utilizar en la generación de datos manuscritos sintéticos es una serie de imágenes de caracteres individuales, en una estructura de directorios, y que se combinan para formar imágenes de DNI completos. Para las firmas manuscritas se hace uso del conjunto de datos [18], que contiene también una estructura de carpetas e imágenes con extensión JPG.

El flujo del *script* de generación de datos comienza preguntando cuántas páginas, de diez firmante cada una, se quieren obtener. Posteriormente, se ejecutan tantas iteraciones como páginas se han solicitado, en cada iteración se generarán por un lado todos los DNI de la página en formato PIL de representación de imágenes en Python mediante el módulo *pillow*. Y por otro lado, se obtiene también el conjunto de imágenes de firma en el mismo formato y que irán asociadas a cada DNI.

Para la obtención de todas las imágenes se ejecutan diez iteraciones por cada página puesto que tal y como se ha indicado se van a generar los datos sintéticos de diez firmantes por página siguiendo la plantilla mostrada en la imagen anterior. En cada una de esas diez iteraciones se ejecutan los siguientes pasos:

1. Generación aleatoria de los ocho números del DNI.
2. Obtención de la letra de control correspondiente a los números generados para el DNI, siguiendo el algoritmo de DNI español [10].
3. Obtención de manera aleatoria de la ruta de una de las imágenes que representan una firma manuscrita.

Posteriormente, una vez se tienen los dígitos y la letra correspondiente generados, se selecciona una de las imágenes que representan a cada carácter de manera aleatoria y

se juntan en una sola imagen en formato PIL. Finalmente, una vez se han generado las diez imágenes en formato PIL de cada uno de los DNI y de las firmas se devuelven y se almacenan.

Tras la generación de los datos para todas las páginas se procede a crear el correspondiente documento PDF lo más similar posible a la plantilla proporcionada por el Parlamento de Canarias, a partir de las imágenes almacenadas con los DNI y firmas para los diez firmantes por página. Puede observarse en la figura 3.7 un ejemplo de una página de datos sintéticos generada.

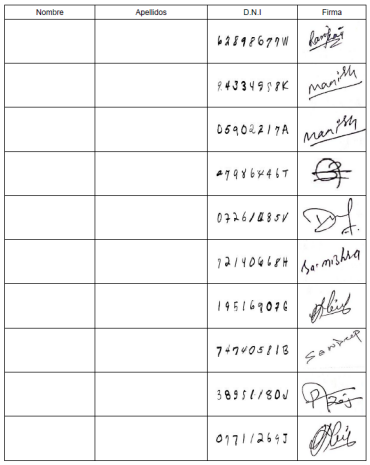


Figura 3.7: Páginas de datos generados sintéticamente

Además del documento PDF, el *script* devuelve también un fichero con extensión .txt que contiene todos los DNI generados, uno por línea. Puesto que será necesario para la evaluación de las tasas de éxito de la aplicación para comparar los DNI reales con las predicciones.

Finalmente, cabe resaltar que todos los datos manuscritos generados sintéticamente se corresponden con DNIs que cumplen el formato correspondiente de ocho números seguidos de una letra, y que además son válidos porque la letra es la correspondiente a esos números. Asimismo, cada uno de los firmantes también tiene siempre alguna firma asignada.

Capítulo 4

Evaluación experimental

4.1. Descripción del proceso de evaluación

4.1.1. Métricas implementadas

La evaluación de la aplicación en sus tres versiones con cada una de las técnicas de inteligencia artificial se ha realizado mediante la implementación de una serie de métricas desde dos puntos de vista, y que se detallan a continuación.

Métricas desde la perspectiva de un problema de OCR

Las métricas desde la óptica de un problema de procesamiento de lenguaje natural (NPL) están centradas en analizar el desempeño de los distintos modelos en el reconocimiento óptico de caracteres (OCR). Por lo que se han calculado a nivel de caracteres comparando los números de DNI reales que estaban en los datos con los DNI predichos por los modelos, carácter a carácter.

- **Precision**: mide la proporción de predicciones correctas positivas (caracteres correctos) sobre todas las predicciones positivas. Se centra en cuántas de las predicciones positivas hechas por el modelo son realmente correctas.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

  Donde **TP** (true positives) se corresponden con el número caracteres correctos y **FP** (false positives) con el número de caracteres incorrectamente reconocidos.

- **Accuracy**: mide la proporción de predicciones correctas (tanto positivas como negativas) sobre el total de predicciones. Es una métrica general de rendimiento del modelo.

$$\text{Accuracy} = \frac{\text{TP}}{\text{Número total de caracteres del DNI real}}$$

- **Recall**: mide la proporción de predicciones correctas positivas sobre todas las instancias reales positivas. Se centra en cuántas de las instancias positivas reales fueron identificadas correctamente por el modelo.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Donde **FN** (false negatives) se corresponde con el número de caracteres no reconocidos.

- **F1-Score**: es la media armónica de la precisión y la recuperación.

$$\text{F1-Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- **Tasa de error de caracteres (CER):** mide el número de errores a nivel de carácter, que incluye las inserciones, eliminaciones y sustituciones necesarias para convertir la predicción del modelo en el DNI correcto.

$$\text{CER} = \frac{S + D + I}{N}$$

Donde **S** se corresponde con el número de sustituciones, **D** con el número de eliminaciones, **I** con el número de inserciones y **N** con el número total de caracteres del DNI real.

Métricas desde la perspectiva de un problema de clasificación de *Machine Learning (ML)*

Las métricas desde el punto de vista de un problema de clasificación de ML se centran en evaluar el desempeño en la clasificación de los DNI predichos respecto de los reales.

- **Precision**: mide la proporción de DNIs correctamente reconocidos entre todos los DNIs que el sistema identificó como correctos.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Donde **TP** (true positives) se corresponde con el número de DNIs correctamente identificados y **FP** (false positives) con el número de DNIs incorrectamente identificados como correctos.

- **Accuracy**: mide la proporción de todas las predicciones correctas respecto al total de predicciones.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

Donde **TN** (true negatives) se corresponde con el número de DNIs correctamente identificados como negativos y **FN** (false negatives) con el número de DNIs no reconocidos correctamente.

- **Recall**: mide la proporción de DNIs correctamente reconocidos entre todos los DNIs que deberían haber sido reconocidos.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

- **F1-Score**: es la media armónica de la precisión y la recuperación.

$$\text{F1-Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

4.1.2. Características de las pruebas realizadas

Las pruebas se han realizado con dos ejemplos de documento PDF, generados sintéticamente tal y como se expuso en el apartado 3.3.1. Los documentos tenían una y 1000 páginas de datos sintéticos. En ambos casos, los DNI son correctos por cumplir el formato y la letra de control es la correspondiente. Y además, cada uno de los DNI siempre tiene asociada su correspondiente firma. Por lo que se debería poder validar tanto el DNI como la presencia de la firma. Esto permite analizar directamente los resultados de los distintos modelos, puesto que no existen errores en los datos originales. Debido a que si nos situamos en el caso de uso de este trabajo, algunos firmantes podrían introducir errores en los caracteres del DNI o no firmar, por lo que la firma no podría ser validada.

Nº páginas	Nº DNIs	Nº firmas
1	10	10
1.000	10.000	10.000

Tabla 4.1: Relación entre el número de páginas de los ejemplos y el número de DNIs y firmas

4.1.3. Entorno de ejecución de la evaluación

La evaluación de los distintos modelos se ha realizado en un ordenador portátil con las siguientes características.

Modelo	MSI Katana GF66 11UC
CPU	11th Gen Intel(R) Core(TM) i7-11800H @ 2.30GHz 2.30 GHz
GPU	NVIDIA GeForce RTX 3050 Laptop GPU
RAM	16 GB
Sistema Operativo	Windows 11

Tabla 4.2: Especificaciones del dispositivo en el que se han ejecutado las pruebas de evaluación

La evaluación se ha llevado a cabo mediante la realización de las pruebas con los dos documentos PDF para cada uno de los modelos. En las tablas 4.3, 4.4, 4.5, 4.6 y 4.7, a continuación, se muestran los tiempos de ejecución de las pruebas.

TrOCR sin *fine-tuning*	
Ejemplo	**Duración del análisis**
1 Página	00:00:04
1000 Páginas	01:52:22

Tabla 4.3: Tiempos de ejecución para la aplicación con el modelo de TrOCR sin *fine-tuning*

TrOCR *fine-tuned* con 450 imágenes	
Ejemplo	**Duración del análisis**
1 Página	00:00:06
1000 Páginas	05:25:34

Tabla 4.4: Tiempos de ejecución para la aplicación con el modelo de TrOCR *fine-tuned* con 450 imágenes

TrOCR *fine-tuned* con 900 imágenes	
Ejemplo	**Duración del análisis**
1 Página	00:00:07
1000 Páginas	03:36:51

Tabla 4.5: Tiempos de ejecución para la aplicación el modelo de TrOCR *fine-tuned* con 900 imágenes

La diferencia de tiempo en la ejecución de la prueba con el documento PDF de 1000 páginas para el modelo de TrOCR ajustado con 450 imágenes, frente al modelo TrOCR entrenado con 900 imágenes se debe a que el ordenador portátil se encontraba realizando otras tareas en simultáneo. Sin embargo, en pruebas anteriores se había obtenido una duración del análisis con el modelo ajustado con 450 imágenes de unas tres horas.

Redes neuronales convolucionales	
Ejemplo	**Duración del análisis**
1 Página	00:00:02
1000 Páginas	00:14:44

Tabla 4.6: Tiempos de ejecución para la aplicación con redes neuronales convolucionales

Máquinas de soporte vectorial	
Ejemplo	**Duración del análisis**
1 Página	00:00:03
1000 Páginas	00:27:44

Tabla 4.7: Tiempos de ejecución para la aplicación con máquinas de soporte vectorial

4.2. Resultados de la evaluación con el resultado directamente devuelto por los modelos

A continuación, se muestran los resultados de la evaluación, desde el punto de vista de un problema de OCR y desde la perspectiva de un problema de clasificación de *Machine Learning* para los DNIs devueltos directamente por los modelos, es decir, sin realizar ningún post-procesado.

4.2.1. Métricas desde el punto de vista de un problema de OCR

Modelo	Ejemplo	Accuracy	Precision	Recall	F1	CER
TrOCR sin fine-tuning	1 Página	81.11 %	82.02 %	98.65 %	89.75 %	18.89 %
TrOCR sin fine-tuning	1000 Páginas	82.97 %	83.93 %	98.64 %	90.69 %	17.03 %
	Media	82.04 %	82.98 %	98.65 %	90.13 %	17.96 %

Tabla 4.8: Resultados de las métricas OCR para el modelo TrOCR sin *fine-tuning* con los resultados directamente del modelo

Modelo	Ejemplo	Accuracy	Precision	Recall	F1	CER
TrOCR fine-tuned con 450 imágenes	1 Página	98.89 %	100 %	98.89 %	99.44 %	1.11 %
TrOCR fine-tuned con 450 imágenes	1000 Páginas	98.81 %	99.31 %	99.50 %	99.40 %	1.18 %
	Media	98.85 %	99.66 %	99.20 %	99.42 %	1.15 %

Tabla 4.9: Resultados de las métricas OCR para el modelo TrOCR *fine-tuned* con 450 imágenes con los resultados directamente del modelo

Modelo	Ejemplo	Accuracy	Precision	Recall	F1	CER
TrOCR fine-tuned con 900 imágenes	1 Página	97.78 %	98.88 %	98.88 %	98.88 %	2.22 %
TrOCR fine-tuned con 900 imágenes	1000 Páginas	98.42 %	98.85 %	99.55 %	99.20 %	1.58 %
	Media	98.10 %	98.87 %	99.22 %	99.04 %	1.90 %

Tabla 4.10: Resultados de las métricas OCR para el modelo TrOCR *fine-tuned* con 900 imágenes con los resultados directamente del modelo

Modelo	Ejemplo	Accuracy	Precision	Recall	F1	CER
CNN	1 Página	74.44 %	75.28 %	98.53 %	85.35 %	25.56 %
CNN	1000 Páginas	71.24 %	71.88 %	98.77 %	83.21 %	28.76 %
	Media	72.84 %	73.58 %	98.65 %	84.28 %	27.16 %

Tabla 4.11: Resultados de las métricas OCR para las redes CNN con los resultados directamente del modelo

Modelo	Ejemplo	Accuracy	Precision	Recall	F1	CER
SVM	1 Página	44.44 %	44.44 %	100 %	61.54 %	55.56 %
SVM	1000 Páginas	48.02 %	48.65 %	97.36 %	64.88 %	51.98 %
	Media	46.23 %	46.55 %	98.68 %	63.21 %	53.77 %

Tabla 4.12: Resultados de las métricas OCR para las máquinas de soporte vectorial con los resultados directamente del modelo

Los resultados obtenidos muestran claramente un mejor desempeño de los modelos con *transformers* a la hora de extraer los caracteres de los DNIs. Frente a las redes neuronales convolucionales, que aunque no tienen resultados tan bajos si que contrastan bastante con los de TrOCR, lo cuál afecta a las métricas desde el punto de vista de un problema de clasificación de *machine learning* y que veremos en el siguiente apartado. Y también, en comparación con las máquinas de soporte vectorial, en las que los resultados descienden significativamente y que tiene sus consecuencias luego a la hora de la clasificación de un DNI completo.

Aunque los modelos TrOCR son mejores vistos los resultados, si los analizamos en profundidad hay diferencias entre los 3 casos, puesto que se ve una mejoría al ajustar el modelo. Sin embargo, se podría pensar que cuantas más imágenes se utilizan para el *fine-tuning* mejores resultados se obtienen, pero en este caso se observa como frente al modelo ajustado con 450 imágenes, el de 900 imágenes obtiene resultados un poco inferiores.

4.2.2. Métricas desde el punto de vista de un problema de clasificación de ML

Modelo	Ejemplo	Accuracy	Precision	Recall	F1
TrOCR sin fine-tuning	1 Página	10 %	100 %	10 %	18.18 %
TrOCR sin fine-tuning	1000 Páginas	5.08 %	96.21 %	5.09 %	9.67 %
	Media	7.54 %	98.11 %	7.55 %	13.93 %

Tabla 4.13: Resultados de las métricas para clasificación ML con los DNI devueltos directamente por el modelo TrOCR sin *fine-tuning*

Modelo	Ejemplo	Accuracy	Precision	Recall	F1
TrOCR fine-tuned con 450 imágenes	1 Página	90 %	100 %	90 %	94.74 %
TrOCR fine-tuned con 450 imágenes	1000 Páginas	90.48 %	96.06 %	93.97 %	95 %
	Media	90.24 %	98.03 %	91.99 %	94.87 %

Tabla 4.14: Resultados de las métricas para clasificación ML con los DNI devueltos directamente por el modelo TrOCR *fine-tuned* con 450 imágenes

Modelo	Ejemplo	Accuracy	Precision	Recall	F1
TrOCR fine-tuned con 900 imágenes	1 Página	80 %	88.89 %	88.89 %	88.89 %
TrOCR fine-tuned con 900 imágenes	1000 Páginas	87.12 %	91.71 %	94.56 %	93.12 %
	Media	83.56 %	90.30 %	91.73 %	91.01 %

Tabla 4.15: Resultados de las métricas para clasificación ML con los DNI devueltos directamente por el modelo TrOCR *fine-tuned* con 900 imágenes

Modelo	Ejemplo	Accuracy	Precision	Recall	F1
CNN	1 Página	0 %	0 %	0 %	0 %
CNN	1000 Páginas	0.8 %	0.84 %	15.12 %	1.59 %
	Media	0.4 %	0.42 %	7.56 %	0.8 %

Tabla 4.16: Resultados de las métricas para clasificación ML con los DNI devueltos directamente por las redes CNN

Modelo	Ejemplo	Accuracy	Precision	Recall	F1
SVM	1 Página	0 %	0 %	0 %	0 %
SVM	1000 Páginas	0.04 %	0.04 %	0.62 %	0.08 %
	Media	0.02 %	0.02 %	0.31 %	0.04 %

Tabla 4.17: Resultados de las métricas para clasificación ML con los DNI devueltos directamente por las máquinas de soporte vectorial

Los resultados desde el punto de vista de la clasificación de los DNI visto como un problema de *Machine Learning*, tal y como se comentó previamente se ven bastante influidos por la capacidad de los distintos modelos de extraer los caracteres. Ya que como se puede ver mientras que el modelo TrOCR ajustado con 450 imágenes supera el 90 % de media de accuracy, el mismo modelo sin ajustar se queda en un 7.54 % de media por lo que aquí claramente se observa la diferencia. Pero destacan aún más los casos de las redes neuronales convolucionales y las máquinas de soporte vectorial. Porque tal y como indican los datos no son capaces de extraer prácticamente ningún DNI al completo, por lo que para el caso de uso no es demasiado útil. Estos resultados tan bajos en CNN y SVM se deben a que desde que falle uno de los caracteres de un DNI ya el DNI es incorrecto porque no es igual al real, y por tanto no valdría para dar un firmante como aceptado.

Los datos muestran además un caso bastante singular y es que como se puede ver en la tabla 4.13, para el ejemplo con una página. Mientras que la accuracy resulta en que el modelo solo es capaz de extraer correctamente y al completo el 10 % de los DNI, que en este caso se corresponde con un solo DNI puesto que el ejemplo tiene 10. Por el contrario, la precisión es de un 100 %, y esto tiene la siguiente explicación.

Teniendo en cuenta la fórmula para calcular la precisión:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Y considerando los siguientes valores reales y predichos:

DNI reales: [’99054213K’, ’26194467C’, ’04872860B’, ’06455523K’, ’88428214E’, ’29045054X’, ’00053950S’, ’66272204G’, ’76150839R’, ’41847731J’]

DNI predichos: [’99054213k.’, ’261944 67c.’, ’04872860B.’, ’064552k.’, ’88428214E’, ’29045054x.’, ’000539505.’, ’66272204G.’, ’76150839R.’, ’4184773 1.’]

Donde se obtiene:

- TP = 1
- FP = 0
- FN = 9
- TN = 0

Por lo tanto, si realizamos el cálculo resultaría lo siguiente:

$$\text{Precision} = \frac{1}{1 + 0} = 1$$

Tal y como se observa en los DNI predichos solo uno de ellos (88428214E), coincide exactamente con su correspondiente DNI real, por lo que los verdaderos positivos (TP) serían uno. Y como se puede ver los 9 DNIs restantes no cumplen con el formato, porque en algunos se introducen espacios en blanco o un punto final, pero sin embargo sus DNI reales si que son correctos. Por tanto, los 9 DNIs entrarían en la categoría de falso negativo, por lo que el resto de parámetros serían 0. Asimismo, si nos ceñimos al significado de la precisión, quiere decir que el 100 % de los DNI identificados como correctos son efectivamente correctos, lo cuál es cierto. Finalmente, cabe señalar que las

fases de post-procesado, tanto la eliminación de caracteres especiales como la corrección de errores se han implementado para tratar de resolver esos errores introducidos por el modelo de TrOCR sin ajuste, en esta caso los puntos finales, los espacios o las letras identificadas en minúsculas.

4.3. Resultados de la evaluación después de eliminar caracteres especiales

En este apartado se muestran los resultados de la evaluación, desde el punto de vista de un problema de OCR y desde la perspectiva de un problema de clasificación de *Machine Learning* para los DNIs tras haber aplicado la fase de eliminación de caracteres especiales del post-procesado.

4.3.1. Métricas desde el punto de vista de un problema de OCR

Modelo	Ejemplo	Accuracy	Precision	Recall	F1	CER
TrOCR sin fine-tuning	1 Página	95.56 %	98.85 %	96.63 %	97.73 %	4.44 %
TrOCR sin fine-tuning	1000 Páginas	95.36 %	97.54 %	97.71 %	97.63 %	4.64 %
	Media	95.46 %	98.20 %	97.17 %	97.68 %	4.54 %

Tabla 4.18: Resultados de las métricas OCR para el modelo TrOCR sin *fine-tuning* después de eliminar los caracteres especiales

Modelo	Ejemplo	Accuracy	Precision	Recall	F1	CER
TrOCR fine-tuned con 450 imágenes	1 Página	98.89 %	100 %	98.89 %	99.44 %	1.11 %
TrOCR fine-tuned con 450 imágenes	1000 Páginas	98.82 %	99.31 %	99.50 %	99.40 %	1.18 %
	Media	98.86 %	99.66 %	99.20 %	99.42 %	1.15 %

Tabla 4.19: Resultados de las métricas OCR para el modelo TrOCR *fine-tuned* con 450 imágenes después de eliminar los caracteres especiales

Modelo	Ejemplo	Accuracy	Precision	Recall	F1	CER
TrOCR fine-tuned con 900 imágenes	1 Página	97.78 %	98.88 %	98.88 %	98.88 %	2.22 %
TrOCR fine-tuned con 900 imágenes	1000 Páginas	98.42 %	98.85 %	99.55 %	99.20 %	1.58 %
	Media	98.10 %	98.87 %	99.22 %	99.04 %	1.9 %

Tabla 4.20: Resultados de las métricas OCR para el modelo TrOCR *fine-tuned* con 900 imágenes después de eliminar los caracteres especiales

Modelo	Ejemplo	Accuracy	Precision	Recall	F1	CER
CNN	1 Página	74.44 %	75.28 %	98.53 %	85.35 %	25.56 %
CNN	1000 Páginas	71.24 %	71.88 %	98.77 %	83.21 %	28.76 %
	Media	72.84 %	73.58 %	98.65 %	84.28 %	27.16 %

Tabla 4.21: Resultados de las métricas OCR para las redes CNN después de eliminar los caracteres especiales

Modelo	Ejemplo	Accuracy	Precision	Recall	F1	CER
SVM	1 Página	44.44 %	44.44 %	100 %	61.54 %	55.56 %
SVM	1000 Páginas	48.02 %	48.65 %	97.36 %	64.88 %	51.98 %
	Media	46.55 %	46.55 %	98.68 %	63.21 %	53.77 %

Tabla 4.22: Resultados de las métricas OCR para las máquinas de soporte vectorial después de eliminar los caracteres especiales

Los resultados obtenidos muestran como solo uno de los modelos varía sus resultados respecto al apartado 4.2.1, en el que se analizaban los DNI predichos directamente por el modelo. Esto se debe a que el citado modelo TrOCR sin ajuste introduce algunos caracteres especiales (puntos, espacios en blanco, guiones o comas), que no están en los DNI reales. Por lo que al incorporar esta fase de eliminación de dichos caracteres los datos mejoran considerablemente.

Los demás modelos no se ven afectados por esta eliminación de caracteres puesto que al haber sido entrenados únicamente con letras y números no pueden producir predicciones con otros caracteres. Por lo que para utilizar estos modelos en los que la variedad de caracteres es más limitada sería conveniente eliminar esta fase, puesto que no aporta una mejoría en los resultados.

4.3.2. Métricas desde el punto de vista de un problema de clasificación de ML

Modelo	Ejemplo	Accuracy	Precision	Recall	F1
TrOCR sin fine-tuning	1 Página	70 %	100 %	70 %	82.35 %
TrOCR sin fine-tuning	1000 Páginas	63.06 %	98.01 %	63.88 %	77.35 %
	Media	66.53 %	99.01 %	66.94 %	79.85 %

Tabla 4.23: Resultados de las métricas para clasificación ML con los DNI después de eliminar los caracteres especiales para el modelo TrOCR sin *fine-tuning*

Modelo	Ejemplo	Accuracy	Precision	Recall	F1
TrOCR fine-tuned con 450 imágenes	1 Página	90 %	100 %	90 %	94.74 %
TrOCR fine-tuned con 450 imágenes	1000 Páginas	90.50 %	96.06 %	93.99 %	95.01 %
	Media	90.25 %	98.03 %	92 %	94.88 %

Tabla 4.24: Resultados de las métricas para clasificación ML con los DNI después de eliminar los caracteres especiales para el modelo TrOCR *fine-tuned* con 450 imágenes

Modelo	Ejemplo	Accuracy	Precision	Recall	F1
TrOCR fine-tuned con 900 imágenes	1 Página	80 %	88.89 %	88.89 %	88.89 %
TrOCR fine-tuned con 900 imágenes	1000 Páginas	87.12 %	91.71 %	94.56 %	93.12 %
	Media	83.56 %	90.30 %	91.73 %	91.01 %

Tabla 4.25: Resultados de las métricas para clasificación ML con los DNI después de eliminar los caracteres especiales para el modelo *fine-tuned* con 900 imágenes

Modelo	Ejemplo	Accuracy	Precision	Recall	F1
CNN	1 Página	0 %	0 %	0 %	0 %
CNN	1000 Páginas	0.80 %	0.84 %	15.12 %	1.59 %
	Media	0.40 %	0.42 %	7.56 %	0.80 %

Tabla 4.26: Resultados de las métricas para clasificación ML con los DNI después de eliminar los caracteres especiales para las redes CNN

Modelo	Ejemplo	Accuracy	Precision	Recall	F1
SVM	1 Página	0 %	0 %	0 %	0 %
SVM	1000 Páginas	0.04 %	0.04 %	0.62 %	0.08 %
	Media	0.02 %	0.02 %	0.31 %	0.04 %

Tabla 4.27: Resultados de las métricas para clasificación ML con los DNI después de eliminar los caracteres especiales para las máquinas de soporte vectorial

En las métricas desde el punto de vista de ML sucede algo similar al caso de las métricas OCR, y esto es que los resultados no cambian respecto al apartado 4.2.2, a excepción del modelo TrOCR sin ajustes. Para el caso del citado modelo se observa un significativo aumento del porcentaje de DNIs completos que la aplicación es capaz de extraer a partir del post-procesado aplicado a las predicciones. Como por ejemplo para la prueba realizada con el ejemplo de una página, que pasa del 10 % sin la eliminación de caracteres especiales al 70 %, que en números absolutos de DNI, serían uno y siete correctamente extraídos al completo, respectivamente.

4.4. Resultados de la evaluación después de la corrección de errores

En este apartado se muestran los resultados de la evaluación, desde el punto de vista de un problema de OCR y desde la perspectiva de un problema de clasificación de *Machine Learning* para los DNIs tras haber aplicado la fase de corrección de errores del post-procesado.

4.4.1. Métricas desde el punto de vista de un problema de OCR

Modelo	Ejemplo	Accuracy	Precision	Recall	F1	CER
TrOCR sin fine-tuning	1 Página	96.67 %	100 %	96.67 %	98.31 %	3.33 %
TrOCR sin fine-tuning	1000 Páginas	96.38 %	98.58 %	97.74 %	98.15 %	3.62 %
	Media	96.53 %	99.29 %	97.21 %	98.23 %	3.48 %

Tabla 4.28: Resultados de las métricas OCR para el modelo TrOCR sin *fine-tuning* después de la corrección de errores

Modelo	Ejemplo	Accuracy	Precision	Recall	F1	CER
TrOCR fine-tuned con 450 imágenes	1 Página	98.89 %	100 %	98.89 %	99.44 %	1.11 %
TrOCR fine-tuned con 450 imágenes	1000 Páginas	98.83 %	99.32 %	99.50 %	99.41 %	1.17 %
	Media	98.86 %	99.66 %	99.20 %	99.43 %	1.14 %

Tabla 4.29: Resultados de las métricas OCR para el modelo TrOCR *fine-tuned* con 450 imágenes después de la corrección de errores

Modelo	Ejemplo	Accuracy	Precision	Recall	F1	CER
TrOCR fine-tuned con 900 imágenes	1 Página	97.78 %	98.88 %	98.88 %	98.88 %	2.22 %
TrOCR fine-tuned con 900 imágenes	1000 Páginas	98.42 %	98.85 %	99.55 %	99.20 %	1.58 %
	Media	98.10 %	98.87 %	99.22 %	99.04 %	1.90 %

Tabla 4.30: Resultados de las métricas OCR para el modelo TrOCR *fine-tuned* con 900 imágenes después de la corrección de errores

Modelo	Ejemplo	Accuracy	Precision	Recall	F1	CER
CNN	1 Página	74.44 %	75.28 %	98.53 %	85.35 %	25.56 %
CNN	1000 Páginas	71.24 %	71.88 %	98.77 %	83.21 %	28.76 %
	Media	72.84 %	73.58 %	98.65 %	84.28 %	27.16 %

Tabla 4.31: Resultados de las métricas OCR para las redes CNN después de la corrección de errores

Modelo	Ejemplo	Accuracy	Precision	Recall	F1	CER
SVM	1 Página	44.44 %	44.44 %	100 %	61.54 %	55.56 %
SVM	1000 Páginas	48.02 %	48.65 %	97.36 %	64.88 %	51.98 %
	Media	46.23 %	46.55 %	98.68 %	63.21 %	53.77 %

Tabla 4.32: Resultados de las métricas OCR para las máquinas de soporte vectorial después de la corrección de errores

Los resultados obtenidos tras la eliminación de caracteres especiales mejoraron bastante el desempeño de la aplicación con el modelo de TrOCR sin ajustes. Sin embargo, se han podido mejorar un poco más añadiendo otra fase adicional al post-procesado, la corrección de errores. Puesto que el citado modelo confunde en ocasiones ciertos números con letras y viceversa, entonces se ha tratado de corregir esos errores, para mejorar el desempeño de la aplicación. Cabe resaltar que esta fase se ha añadido porque las pruebas se han realizado con datos correctos, por lo que se ha podido comprobar que es el propio modelo de TrOCR el que introduce esos pequeños errores y confusiones. Los demás modelos, no experimentan ninguna variación en cuanto a sus resultados, lo que puede deberse a que por ejemplo en el caso de las CNN y los SVM no se pueden confundir letras con números porque los modelos se han entrenado de manera independiente para los números por un lado, y las letras por otro lado, entonces no habría posibilidad de realizar correcciones.

4.4.2. Métricas desde el punto de vista de un problema de clasificación de ML

Modelo	Ejemplo	Accuracy	Precision	Recall	F1
TrOCR sin fine-tuning	1 Página	80 %	100 %	80 %	88.89 %
TrOCR sin fine-tuning	1000 Páginas	71.96 %	94.26 %	75.26 %	83.69 %
	Media	75.98 %	97.13 %	77.63 %	86.29 %

Tabla 4.33: Resultados de las métricas para clasificación ML con los DNI después la corrección de errores para el modelo TrOCR sin *fine-tuning*

Modelo	Ejemplo	Accuracy	Precision	Recall	F1
TrOCR fine-tuned con 450 imágenes	1 Página	90 %	100 %	90 %	94.74 %
TrOCR fine-tuned con 450 imágenes	1000 Páginas	90.60 %	96.07 %	94.09 %	95.07 %
	Media	90.30 %	98.04 %	92.05 %	94.91 %

Tabla 4.34: Resultados de las métricas para clasificación ML con los DNI después la corrección de errores para el modelo TrOCR *fine-tuned* con 450 imágenes

Modelo	Ejemplo	Accuracy	Precision	Recall	F1
TrOCR fine-tuned con 900 imágenes	1 Página	80 %	88.89 %	88.89 %	88.89 %
TrOCR fine-tuned con 900 imágenes	1000 Páginas	87.12 %	91.71 %	94.56 %	93.12 %
	Media	83.56 %	90.30 %	91.73 %	91.01 %

Tabla 4.35: Resultados de las métricas para clasificación ML con los DNI después la corrección de errores para el modelo *fine-tuned* con 900 imágenes

Modelo	Ejemplo	Accuracy	Precision	Recall	F1
CNN	1 Página	0 %	0 %	0 %	0 %
CNN	1000 Páginas	0.8 %	0.84 %	15.12 %	1.59 %
	Media	0.4 %	0.42 %	7.56 %	0.8 %

Tabla 4.36: Resultados de las métricas para clasificación ML con los DNI después la corrección de errores para las redes CNN

Model	Ejemplo	Accuracy	Precision	Recall	F1
SVM	1 Página	0 %	0 %	0 %	0 %
SVM	1000 Páginas	0.04 %	0.04 %	0.62 %	0.08 %
	Media	0.02 %	0.02 %	0.31 %	0.04 %

Tabla 4.37: Resultados de las métricas para clasificación ML con los DNI después la corrección de errores para las máquinas de soporte vectorial

En las métricas desde el punto de vista de un problema de clasificación de ML, tras las corrección de errores, se produce la misma situación que en el apartado 4.3.2. Puesto que tal y como se puede observar en los datos, el modelo TrOCR sin ajustes ve aumentado su accuracy del 70 % tras eliminación de caracteres especiales al 80 % después de corregir también los errores. Por el contrario, los demás modelos mantienen los mismos resultados.

4.5. Evaluación de la detección de la firma

La evaluación de la detección de la presencia de las firmas se ha realizado también con las mismas pruebas que para el caso de las métricas. Y ha dado como resultado un 100 % de firmas detectadas en todas las pruebas realizadas. Esto quiere decir que el modelo *zero-shot* para la clasificación de imágenes [23] es muy bueno. Cabe destacar además que se realizó alguna prueba con imágenes vacías sin ninguna firma, y el modelo fue capaz de predecir que efectivamente no había ninguna firma en la imagen.

Capítulo 5

Conclusiones y líneas futuras

5.1. Conclusiones

En conclusión, este trabajo tuvo como objetivo el análisis y desarrollo de un aplicativo para reemplazar el método manual de verificación de datos manuscritos en la recogida de firmas. Para ello, se desarrollaron tres versiones de la aplicación, cada una utilizando una técnica de inteligencia artificial diferente en la fase de reconocimiento óptico de caracteres: los *transformers*, las redes neuronales convolucionales y las máquinas de soporte vectorial. Se evaluó el desempeño de cada técnica y se encontró que el modelo de TrOCR ajustado con 450 imágenes obtuvo el mejor desempeño, con una accuracy (exactitud) media del 98.85 % en las métricas OCR y un 99.66 % de precisión, lo que indica una alta fiabilidad, ya que prácticamente el 100 % de los caracteres detectados como correctos lo son realmente. Además, el modelo alcanzó un 90.24 % de accuracy promedio en la clasificación de los DNIs y un 98.03 % de precisión en términos de un problema de clasificación de *Machine Learning*, destacando a los *transformers* como la técnica más efectiva entre las analizadas cuando se aplica un ajuste fino con 450 imágenes.

En resumen, este trabajo demuestra que el uso de técnicas avanzadas de inteligencia artificial, particularmente los *transformers*, puede agilizar los procedimientos administrativos, y en particular en el caso de uso de este trabajo mejorando el proceso de verificación de los datos manuscritos de los electores en una recogida de firmas.

5.2. Líneas futuras

Para futuras investigaciones, sería interesante analizar si el incremento del número de imágenes de entrenamiento en el modelo de *transformers* para OCR podría mejorar aún más los resultados obtenidos. En caso de no observar mejoras, sería fundamental estudiar las razones detrás de esta observación, dado que en este trabajo se encontró que el modelo ajustado con 450 imágenes produjo mejores resultados que el ajustado con 900 imágenes, lo cual es contrario a lo esperado. Esto sugiere la necesidad de una investigación más profunda para entender los factores que afectan el rendimiento del modelo al aumentar el volumen de datos de entrenamiento.

Además, también se podría realizar una evaluación en entornos reales del desempeño de la aplicación. Puesto que las variaciones en la calidad de las imágenes y la escritura puede ayudar a determinar la robustez del aplicativo en condiciones reales.

Capítulo 6

Summary and Conclusions

6.1. Conclusions

In conclusion, this Bachelor's Thesis aimed to analyze and develop an application to replace the manual method of verifying handwritten data in signature collection. For this purpose, three versions of the application were developed, each of them using a different artificial intelligence technique in the optical character recognition phase: transformers, convolutional neural networks, and support vector machines. The performance of each technique was evaluated, and it was discovered that the TrOCR model fine-tuned with 450 images achieved the best performance, with an average OCR accuracy of 98.85 % and a precision of 99.66 %, indicating high reliability since nearly 100 % of the characters detected as correct are indeed correct. Additionally, the model achieved an average accuracy of 90.24 % in DNI classification and a precision of 98.03 % in terms of a machine learning classification problem, highlighting transformers as the most effective technique among those analyzed when fine-tuned with 450 images.

In summary, this Bachelor's Thesis demonstrates that the use of advanced artificial intelligence techniques, particularly transformers, can streamline administrative procedures, and in the case of this work, improve the process of verifying handwritten electors data in a signature collection.

6.2. Future lines

For future research, it would be interesting to analyze whether increasing the number of training images in the *transformers* model for OCR could further improve the obtained results. If no improvements are observed, it would be essential to study the reasons behind this observation, given that in this work, the model trained with 450 images produced better results than the one trained with 900 images, which is contrary to expectations. This suggests the need for deeper investigation to understand the factors affecting the model's performance as the volume of training data increases.

Additionally, an evaluation of the application's performance in real-world environments could also be conducted. Since variations in image quality and handwriting could help determine the robustness of the application under real conditions.

Capítulo 7

Presupuesto

7.1. Distribución de tareas y costos

A continuación, en la tabla 7.1, se muestra un presupuesto desglosado por tareas y con los correspondientes precios por horas, para la realización de este trabajo.

Tarea	Horas	Precio/h	Coste
Pruebas previas	6	15,00 €	90,00 €
Estado del arte	25	35,00 €	875,00 €
Desarrollo de la aplicación	160	60,00 €	9.600,00 €
Generación sintética de los datos	16	35,00 €	560,00 €
Evaluación de resultados	15	30,00 €	450,00 €
Redacción de la memoria y otras tareas	78	40,00 €	3.120,00 €
Horas totales	300 horas	Coste total	14.695,00 €

Tabla 7.1: Presupuesto distribuido por tareas y número de horas

Bibliografía

[1] Reglamento del parlamento de canarias. URL https://www.parcan.es/pub/reglamento.pdf.

[2] Conjunto de datos con dígitos manuscritos. URL https://www.kaggle.com/datasets/oddrationale/mnist-in-csv?select=mnist_train.csv.

[3] Conjunto de datos con letras manuscritas. URL https://www.kaggle.com/datasets/sachinpatel21/az-handwritten-alphabets-in-csv-format/data?select=A_Z+Handwritten+Data.csv.

[4] Scikit-learn - recognizing hand-written digits. URL https://scikit-learn.org/stable/auto_examples/classification/plot_digits_classification.html.

[5] Reconocimiento de la estructura de una tabla con transformers. URL https://huggingface.co/bilguun/table-transformer-structure-recognition.

[6] C. Aggarwal. *An Introduction to Neural Networks*, pages 1–27. Springer International Publishing, Cham, 2023. ISBN 978-3-031-29642-0. doi: 10.1007/978-3-031-29642-0_1. URL https://doi.org/10.1007/978-3-031-29642-0_1.

[7] C. Aggarwal. *Convolutional Neural Networks*, pages 305–360. Springer International Publishing, Cham, 2023. ISBN 978-3-031-29642-0. doi: 10.1007/978-3-031-29642-0_9. URL https://doi.org/10.1007/978-3-031-29642-0_9.

[8] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik. Emnist: an extension of mnist to handwritten letters, 2017.

[9] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967. doi: 10.1109/TIT.1967.1053964.

[10] M. del Interior Gobierno de España. Cálculo del dígito de control del nif/-nie. URL https://www.interior.gob.es/opencms/ca/servicios-al-ciudadano/tramites-y-gestiones/dni/calculo-del-digito-de-control-del-nif-nie/.

[11] S. Fernández, C. Javier, and V. S. Consuegra. Reconocimiento óptico de caracteres (ocr). *Univ.Carlo*, 3(7):2008, 2008.

[12] A. Géron. *Understanding support vector machines*. O'Reilly Media, Inc, 2017.

[13] IBM. ¿qué son las redes neuronales? URL https://www.ibm.com/es-es/topics/neural-networks.

[14] N. Islam, Z. Islam, and N.Ñoor. A survey on optical character recognition system. *arXiv preprint arXiv:1710.05703*, 2017.

[15] M. Li, T. Lv, L. Cui, Y. Lu, D. Florencio, C. Zhang, Z. Li, and F. Wei. Trocr: Transformer-based optical character recognition with pre-trained models, 2021. URL https://huggingface.co/microsoft/trocr-large-handwritten.

[16] D. R. Luna. Svm. URL https://github.com/DavidReveloLuna/Machine-Learning/blob/master/3_5_M%C3%A1quinas_de_Soporte_Vectorial_SVM.ipynb.

[17] E. S. Olivas. *Sistemas de Aprendizaje Automatico*. Paracuellos de Jarama, Madrid: RaMa, Paracuellos de Jarama, Madrid, 2023. Çontenidos adaptados al Curso de Especializacion en Inteligencia Artificial y Big Data-Cover.; ID: $\mathrm{TN}_{c}di_{e}libro_{b}ooks_{E}LB235049$.

[18] S. Pokharel, S. Giri, and P. Sharma. Conjunto de datos con firmas manuscritas, 2022. URL https://www.kaggle.com/dsv/4200766.

[19] J. C. Roma, T. L. Bagén, and A. B. Rué. *Deep Learning : Principios y Fundamentos*. Editorial UOC, Barcelona, 2020. ISBN 9788491806578. URL http://ebookcentral.proquest.com/lib/bull-ebooks/detail.action?docID=7025971. ID: 7025971.

[20] P. Singh and S. Budhiraja. Feature extraction and classification techniques in ocr systems for handwritten gurmukhi script–a survey. *International Journal of Engineering Research and Applications (IJERA)*, 1(4):1736–1739, 2011.

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.Ñ. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. 2017. URL https://arxiv.org/pdf/1706.03762.pdf.

[22] N. Watson. *Understanding Convolutional Neural Networks (CNNs)*. Infinite Skills, 2017.

[23] X. Zhai, B. Mustafa, A. Kolesnikov, and L. Beyer. Sigmoid loss for language image pre-training, 2023. URL https://huggingface.co/google/siglip-base-patch16-256-multilingual.