Implementación de reconocimiento óptico de caracteres para la digitalización de documentos

Camilo Andrés Sabogal Aristizábal

Tesis o trabajo de investigación presentada(o) como requisito parcial para optar al título de:

Especialista en Analítica y Ciencia de Datos

Asesor:

Sebastián Rodríguez Colina, Magíster en Ingeniería

Universidad de Antioquia
Facultad de Ingeniería
Especialización en Analítica y Ciencia de Datos
Medellín, Antioquia, Colombia
2022

Cita	(Sabogal Aristizábal, 2022)
Referencia **Estilo APA 7 (2020)**	Sabogal Aristizábal. (2022). *Implementación de reconocimiento óptico de caracteres para la digitalización de documentos,* [Trabajo de grado especialización]. Universidad de Antioquia, Medellín, Colombia.

Especialización en Analítica y Ciencia de Datos, Cohorte III.

TABLA DE CONTENIDOS

1. RESUMEN EJECUTIVO

La digitalización de documentos es una ciencia que permite traducir tipos de documentos escaneados o imágenes en datos analizables, editables y buscables. Esto es muy útil, ya que permite guardar datos históricos, jurídicos, culturales, científicos, etc. de forma segura y pueden ser de utilidad para investigaciones futuras. Sin embargo, digitalizarlos de forma manual es laborioso y puede tomar mucho tiempo lo que genera grandes costos. Este tipo de digitalizaciones se pueden clasificar en dos categorías: el reconocimiento de caracteres basado en reglas y el reconocimiento de caracteres basado en aprendizaje automático. Por las razones anteriores, es importante contar con herramientas que automaticen este proceso como el aprendizaje automático, ya que así se podrán digitalizar un mayor número de manuscritos en menor tiempo.
El proyecto persigue la automatización de la conversión de documentos escaneados a texto editable.
Se utilizaron datos de documentos escaneados, exactamente 600 archivos de palabras y documentos, de diferentes fuentes y se siguió una estrategia de mejora incremental para las iteraciones, con el objetivo de desarrollar un modelo de OCR para manuscritos que sea de utilidad al servicio de la comunidad.
Los principales obstáculos encontrados fueron la falta de una base de datos de entrenamiento robusta, adecuada y la variabilidad de los formatos de documentos escaneados.

Los notebooks del proyecto descrito en esta monografía pueden ser consultados en el siguiente repositorio de GitHub: https://github.com/CamiloSaboA-csv/digitalizador_manuscritos

2. DESCRIPCIÓN DEL PROBLEMA

2.1 PROBLEMA DE NEGOCIO

Dada la abundante presencia de los documentos escritos a mano en la gran mayoría de transacciones humanas, la digitalización de documentos tiene un valor práctico invaluable. El reconocimiento óptico de caracteres es una ciencia que permite traducir tipos de documentos escaneados o imágenes en datos analizables, editables y buscables. Durante la última década, los investigadores han utilizado herramientas de inteligencia artificial/aprendizaje automático para analizar documentos manuscritos e impresos para convertirlos a formato electrónico.

Los manuscritos son una fuente de información muy valiosa, ya que en ellos se consignan datos históricos, jurídicos, culturales, científicos, etc. que pueden ser de utilidad para investigaciones futuras. Sin embargo, digitalizarlos de forma manual es laborioso y puede tomar mucho tiempo lo que genera grandes costos, adicional es altamente dependiente del ojo humano, lo que puede inducir a errores si la imagen no tiene una buena calidad. Por esta razón, es importante contar con herramientas que automatizan este proceso, ya que así se podrán digitalizar un mayor número de manuscritos en menor tiempo.
Este proyecto pretende optimizar el proceso de digitalización de documentos a través de la clasificación de los caracteres en las imágenes de documentos, con el fin de reducir los tiempos de digitalización, reducir errores humanos y generar una copia digital de documentos físicos, teniendo efectos positivos la reducción de costos en la realización de esta actividad y múltiples beneficios agregados como la indexación de documentos físicos.

2.2 APROXIMACIÓN DESDE LA ANALÍTICA DE DATOS

El objetivo del modelo generado es clasificar los caracteres que se encuentren en el texto visible en archivos en formato de imagen como fotografías de documentos físicos o escaneos de manuscritos. Estos serán la entrada del modelo y se generan como salidas archivos en formato de texto manipulable esperando así una digitalización fiel del texto original pero con el valor agregado de permitir manipulación, copiado e indexado de palabras en el documento.

2.3 ORIGEN DE LOS DATOS

Se utilizaron datos con dos orígenes diferentes:

Datos sintéticos generados artificialmente: se creó un dataset a partir de imágenes generadas sintéticamente de manuscritos utilizando un módulo experimental de la librería "TextRecognitionDataGenerator" con el objetivo de nutrir el modelo con un balance de letras y palabras que no se encontraran tan sesgadas por la naturaleza humana, las palabras se seleccionan al azar del diccionario de un idioma español.
Datos propios: Se generó un dataset con 100 fotografías de textos escritos a mano por tres colaboradores con el propósito de nutrir el modelo con manuscritos propios.

2.4 MÉTRICAS DE DESEMPEÑO

Las métricas escogidas para la evaluación del desempeño del modelo son el Character Error Rate (**CER**) y el Word Error Rate (**WER**) en donde CER Indica el número de caracteres que se han reconocido incorrectamente y, por lo tanto, es útil para identificar errores, el WER por otro lado es una buena métrica para evaluar el rendimiento del modelo en términos de eficacia. Indica el número de palabras que se han reconocido incorrectamente.
La diferencia entre ambas métricas es que en el CER se considera que transiciones erróneas son igual de graves que los caracteres sustituidos, mientras que en el WER se considera que transiciones incorrectas son menos graves que los caracteres sustituidos.

Estas métricas nos permiten medir los errores que nuestro modelo puede tener.

- Las métricas usadas para medir el desempeño del modelo se fundamenta en dos fórmulas explicados a continuación:
    - CER, es la métrica que nos permite conocer el porcentaje de caracteres que se sustituyeron, eliminaron o se insertaron para convertir un escrito en otro y está definido por la siguiente fórmula.

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

      Con:
        S = Número de caracteres sustituidos
        D = Número de caracteres eliminados
        I = Número de caracteres insertados
        N = Número total de caracteres en el escrito original
    - WER, es la métrica que nos permite conocer el porcentaje de palabras que se sustituyeron, eliminaron o se insertaron para convertir un escrito en otro.

$$WER = \frac{S_w + D_w + I_w}{N_w}$$

Con:

S = Número de palabras sustituidas
D = Número de palabras eliminadas
I = Número de palabras insertadas
N = Número total de palabras en el escrito original

- La tasa de acierto: La tasa de acierto se refiere al porcentaje de veces en que el modelo de ML es capaz de identificar correctamente un carácter en un manuscrito digitalizado.
- La velocidad de procesamiento: La velocidad de procesamiento se refiere al tiempo que el modelo de ML requiere para identificar caracteres en un manuscrito digitalizado.

Teniendo en cuenta las métricas expuestas anteriormente se definen los siguientes intervalos:

- Una buena precisión del reconocimiento con buen desempeño se tendría con un CER menor al 10 % y un WER menor al 20%
- Una mala precisión del reconocimiento con mal desempeño se tendría al obtener un CER mayor al 10 % y un WER mayor al 20%, lo que nos daría como resultado una digitalización de mala calidad.

Es preciso mencionar que el WER generalmente se encuentra correlacionado con el CER, exceptuando los casos en donde las tasas de error son excesivamente altas (CER>80%), aunque se espera que el valor absoluto del WER sea más alto que el valor del CER, en los casos en los que el WER se encuentre en el intervalo de mala precisión y el CER se encuentre en el intervalo de buena precisión, se tomará como un reconocimiento aceptable pero de baja calidad.
Estos intervalos se definieron de esta manera tomando como referencia de apoyo el artículo publicado por Simeon Keates (Keates, 2017) , en donde habla de la aceptación de las métricas, teniendo en cuenta este artículo y conociendo la naturaleza del negocio, se decidió apoyar la información técnica realizando una encuesta de percepción a cinco desarrolladores de ML en donde se pregunto que cual era la cantidad de palabras y caracteres errados que considera tolerable en una digitalización, realizando una evaluación subjetiva entre estos datos obtenidos y diferentes artículos, se concluyó que las métricas de aceptación eran las antes mencionadas .
Por lo tanto se espera que el modelo logre digitalizar el 80% de las palabras con un CER menor al 10%.

3. DATOS

3.1 DATOS ORIGINALES

Todos los archivos se encuentran en formato .png y hacen referencia a fotografías de palabras y se encuentran distribuidos de la siguiente manera.

Dataset	Subdivisión 1	Cantidad imágenes	Peso [MB]
Datos sinteticos	Train	500	12.3
	Test	50	1.54
	Validation	50	1.54
Datos personales	Train	150	1.4
	Test	15	0.17
	Validation	15	0.17

Los datos no presentan ninguna restricción de acceso puesto que fueron generados de manera simultánea, las resoluciones de las imágenes son variables debido a las variaciones en la longitud de las palabras, sin embargo en el dataset de imágenes generadas sintéticamente las imágenes tienen una medida constante de 64 píxeles de altura.

3.2 DATASETS

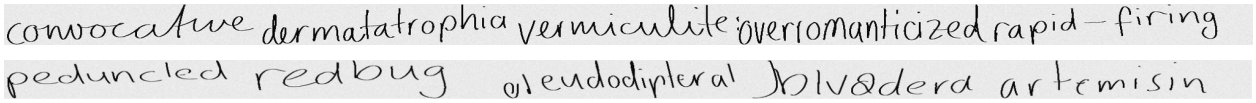
Para construir los datos de entrenamiento y validación se realizó un proceso poco usual debido a que se está utilizando un algoritmo de entrenamiento ya definido por la herramienta a utilizar, el inconveniente se genera debido a que el input requerido del modelo son palabras individuales y una parte de nuestro dataset consta de manuscritos completos, por lo que para el entrenamiento, se tomaron las palabras individuales del dataset.
El número de palabras a tomar en cuenta para los datasets de entrenamiento y validación se seleccionó manualmente, debido a que al aumentar el número de palabras se incrementaron los errores en la clasificación, sin embargo, al disminuir el número se reducían los errores, pero el número de palabras se volvía insuficiente para el entrenamiento.

3.3 DESCRIPTIVA

Como se mencionó en 2.3 se utilizaron dos orígenes diferentes:
Datos generados artificialmente: se creó un dataset a partir de imágenes generadas sintéticamente utilizando la librería “TextRecognitionDataGenerator”(Belval, 2020) con el objetivo de nutrir el modelo con un balance de letras y palabras que no se encontraran tan sesgadas por la naturaleza humana, las palabras se seleccionan al azar del diccionario de un idioma específico. Luego, se generará una imagen de esas palabras usando la fuente, el fondo y las modificaciones seleccionadas, en este caso (ancho del trazo, borrosidad del fondo, deformación del texto, variación del fondo, inclinación del texto y espaciado entre caracteres) de manera aleatoria.

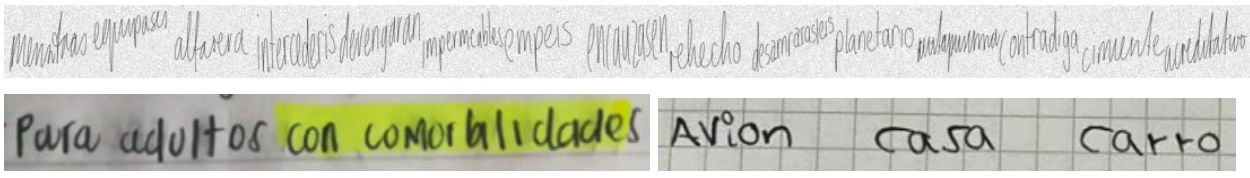
Se generaron fotografías de textos con diferentes tamaños, diferentes tipografías, diferentes ángulos de inclinación, diferentes fondos, diferentes espaciamientos y diferentes niveles de difuminación.
La division de este dataset consta de:

- 500 palabras generadas sintéticamente emulando escritura a mano.
- 100 oraciones de 15 palabras generadas sintéticamente emulando escritura a mano.
- 200 oraciones de texto sintético con diferentes fuentes.



Datos propios: Se generó un dataset con 100 fotografías de textos escritos a mano por tres colaboradores con la finalidad de nutrir el modelo con manuscritos propios, posterior a esto, se genero un segundo dataset extraído a partir de estos textos, estos dataset se pueden describir de la siguiente manera:

- Oraciones : Se fotografiaron 20 oraciones con diferente cantidad de palabras.
- Palabras: Se extrajeron 100 palabras como imágenes individuales de las oraciones escritas a mano inicialmente, esto con la finalidad de facilitar el entrenamiento del modelo de EasyOCR ya que cuenta con la limitante de únicamente recibir palabras como entrada.



Posterior a la distinción de cada uno de estos datasets, se realizaron algunas gráficas de manera exploratoria con el fin de identificar información valiosa:

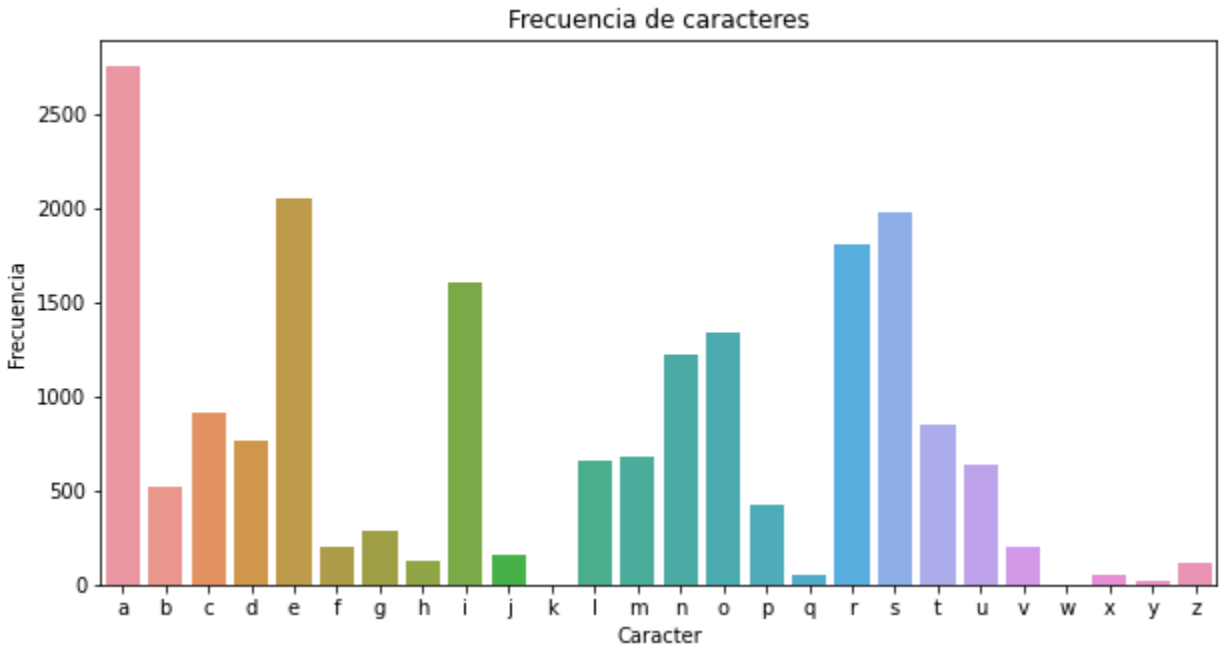


Figura 1. *Tabla de frecuencia de los caracteres en el dataset*

En la gráfica se evidencia que hay un desbalance en la cantidad de caracteres del dataset y se muestra que hay una baja cantidad de caracteres como la **k, q, y, w** y **x.**

También se generó interés en conocer la longitud de las palabras que se encontraban en el dataset y se pudo observar que la gran mayoría de las palabras tienen entre siete y trece caracteres.

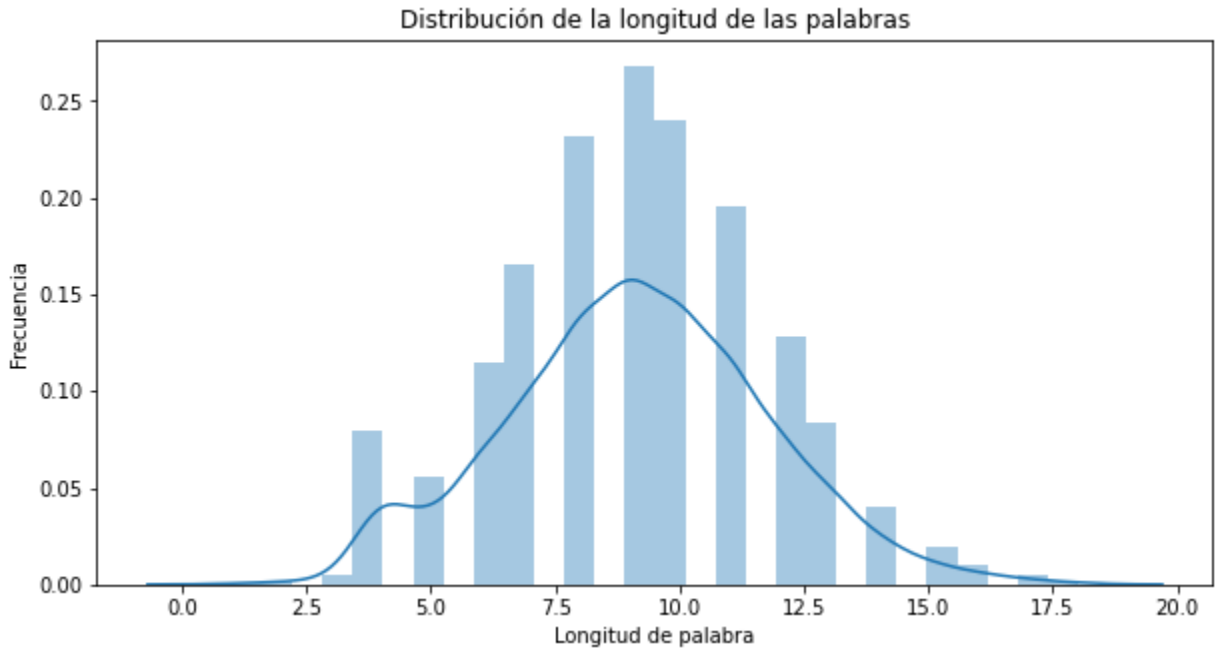


Figura 2. *Longitud de las palabras dentro del dataset*

4. PROCESO DE ANALÍTICA

4.1 PIPELINE PRINCIPAL

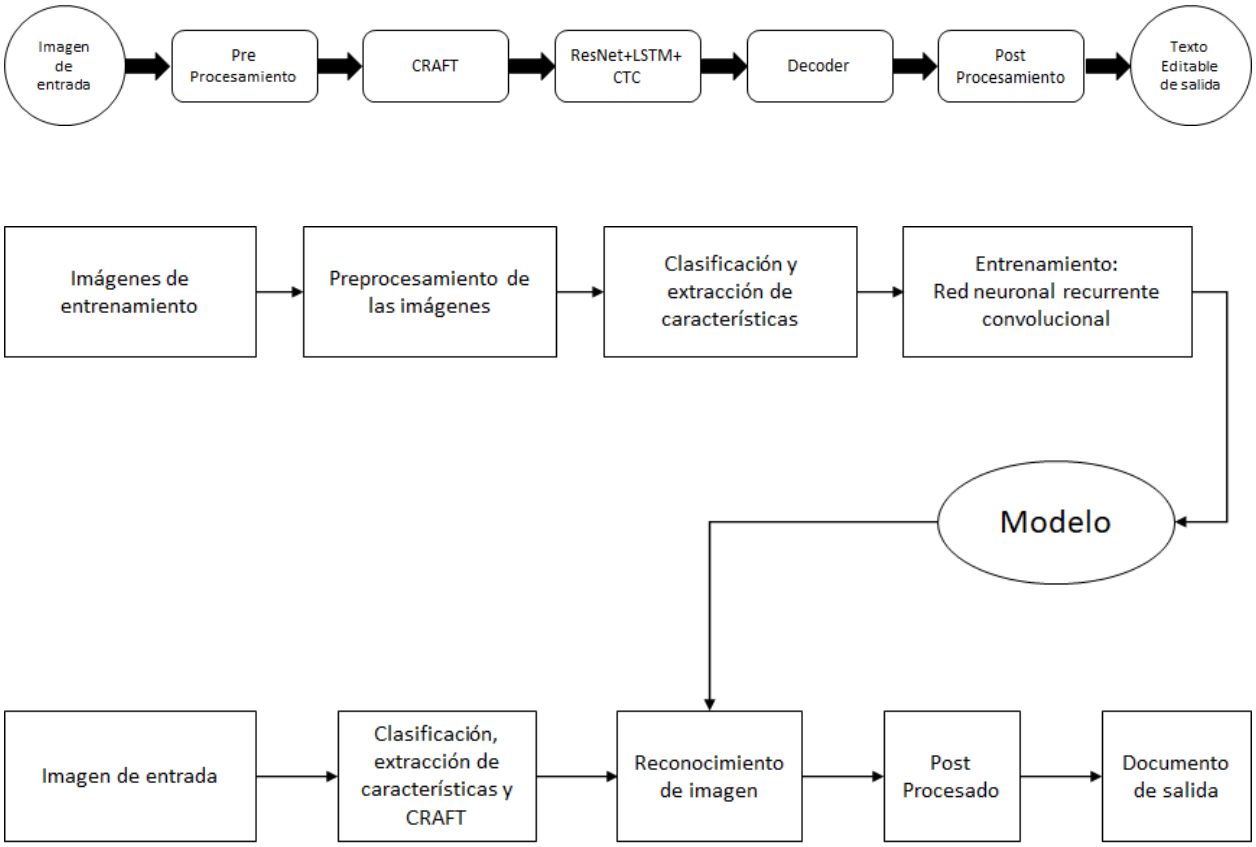


Figura 3. *Flujo de entrenamiento general para el método propuesto. El entrenamiento se realiza mediante imágenes tanto reales como sintéticas.*

4.2 PREPROCESAMIENTO

Primero, utilizamos una clase para alinear y ordenar imágenes en un lote. tomando tres argumentos: la altura y el ancho de la imagen de salida. Adicional se definió un booleano que especifica si se conservará el ratio de aspecto de la imagen de entrada o si se rellenará con píxeles para que coincida con el tamaño especificado. Posterior a esto, tomamos como argumento el lote de imágenes y las transformamos. Se inicia calculando el tamaño máximo de la imagen redimensionada y el número de canales de entrada. A continuación, se define un objeto de transformación llamado NormalizePAD. NormalizePAD se usa para aplicar una normalización al lote de imágenes. Por último, se recorre la lista de imágenes, se redimensiona cada imagen según el tamaño especificado y se aplica la transformación NormalizePAD. Las imágenes

transformadas se agregan a una nueva lista. Finalmente, se devuelven las imágenes transformadas.
También, se determina el contraste actual en una escala de grises por medio de una función con la finalidad de poder transformar la imagen al contraste definido para facilitar el procesamiento, en este caso el contraste objetivo por defecto es 0.4.
Ya para finalizar la etapa de preprocesamiento, se transforman las imágenes en mapas de bits antes de enviarlas a la red neuronal utilizando la librería PIL.

4.3 MODELOS

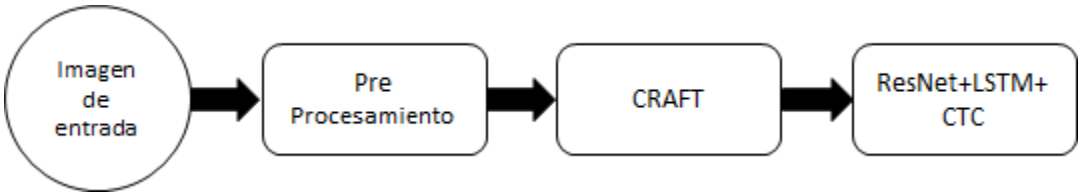


Figura 4. *Procedimiento inicial de 1. Preprocesamiento, 2. segmentación de caracteres para lograr una división de las palabras y los caracteres y 3. Modelo de reconocimiento.*

La arquitectura del modelo consta de tres partes:
1) capas convolucionales, que extraen un conjunto de características secuencia de la imagen de entrada;
2) capas recurrentes, que predicen una distribución de etiquetas para cada región;
3) capa de transcripción, que traduce las predicciones por cuadro en la secuencia de etiqueta final.

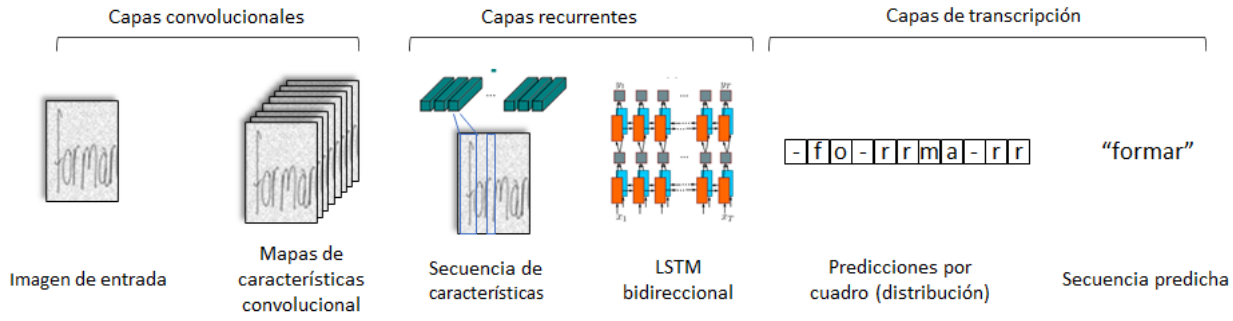


Figura 5. *Arquitectura de red. La arquitectura consta de tres partes: 1. capas convolucionales, que extraen un conjunto de características secuenciales de la imagen de entrada; 2. capas recurrentes, que predicen una distribución de etiquetas para cada cuadro; 3. capa de transcripción, que traduce las predicciones por cuadro en la secuencia de etiqueta final.*

De manera sencilla y general, el funcionamiento del modelo es el siguiente, posterior al preprocesamiento de los datos, se utiliza el modelo CRAFT (Baek et al., 2019) que se encarga de realizar la detección del texto dentro de la imagen y delimitando el área o la caja de las diferentes palabras que se encuentren en la imagen, luego, continuamos con el entrenamiento del modelo de reconocimiento de caracteres. En primer lugar, obtiene el lote de datos de entrenamiento, que incluye las imágenes y etiquetas. Luego
codifica las etiquetas usando el converter que puede ser con el criterio CTC o el Attention. En segundo lugar, alimenta las imágenes y las etiquetas codificadas al modelo. Si el converter es 'CTC', el modelo devuelve una matriz de predicciones. Se calcula el costo de la predicción usando la función criterion y se realiza el backpropagation. En tercer lugar, el optimizador actualiza los pesos del modelo.

Modelo de detección - CRAFT

Figura 6. *Visualización de la detección a nivel de carácter mediante CRAFT. 1. Imagen de entrada. 2. Mapas de calor predichos por nuestro marco propuesto. 3. Detección de resultados para textos de varias formas. Adaptado de (Baek et al., 2019).*

El modelo CRAFT se encarga de la detección de texto, CRAFT de las siglas (character-Región Awareness) nos permite la segmentación del texto en la imagen, el modelo opera detectando y delimitando el área de texto de los caracteres que se encuentran presentes en una imagen y se encuentra basado en redes neuronales.

El módulo CRAFT, es el primero módulo, este implementa una red neuronal convolucional que se usa para la detección de objetos en imágenes. La red está basada en la arquitectura de VGG-16 y se compone de una serie de módulos de convolución que se encargan de extraer características de la imagen de entrada. Cada uno de los módulos, consiste en una secuencia de 6 capas. La primera es una capa de convolución 1D, seguida de una capa de batch normalization, una capa de activación ReLU, otra capa de convolución 1D y batch normalization, y finalmente otra capa de activación ReLU.
Estas características se usan después para generar un mapa de predicción que indica la presencia de objetos en la imagen.

Como salida, este modelo genera un mapa de puntajes de afinidad y un mapa de puntajes de región como se muestra en la siguiente figura.

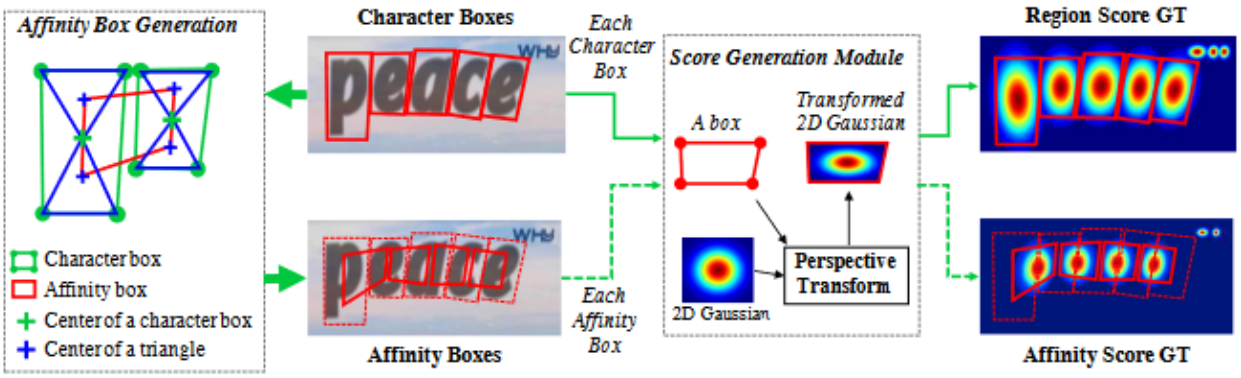


Figura 7. *Procedimiento de generación de datos reales. Generamos etiquetas de verdad sobre el terreno a partir de una imagen sintética que tiene anotaciones a nivel de carácter. Tomado de (Baek et al., 2019).*

En resumen, CRAFT es un detector de caracteres que funciona de una manera muy óptima para la detección de caracteres individuales en casi todos los alfabetos existententes, con buenos resultados incluso cuando los caracteres presentan grandes deformaciones, interferencias y diferentes formas de texto como inverso u horizontal, generando contornos muy acertados, este modelo ha demostrado un rendimiento de última generación en muchos de los datasets públicos de imágenes con texto y se podría catalogar como uno de los mejores detectores de caracteres para entornos generales.

Modelo de entrenamiento del reconocimiento

Para el modelo de reconocimiento de caracteres, se utilizaron las librerías torch y torchvision para cargar las imágenes y las capas del modelo.El modelo se entrena usando el criterio CTC Loss y el optimizador Adam.
El modelo toma como argumentos el dataset de entrenamiento, el criterio de pérdida, el optimizador y el número de épocas. El entrenamiento consta de iteraciones sobre el dataset para obtener los mini-batches de datos, para cada mini-batch, se extraen las imágenes y las etiquetas similar a (He et al., 2015), después, el modelo se ejecuta en modo autocast para obtener las predicciones, luego, se calcula la función de pérdida y se realiza el backpropagation y por último, se actualiza el optimizador y se guarda el modelo.

5. METODOLOGÍA

5.1 BASELINE

Para el baseline se ejecutó el flujo de trabajo antes mencionado con el modelo de reconocimiento por defecto de la librería Easy OCR ingresando como entrada una imagen de un documento manuscrito sin modificar ningún parámetro, como era previsto el modelo no se acercó al resultado esperado, se obtuvieron las métricas presentadas en el siguiente gráfico.

MODELO	**Dataset propio**		**Dataset sintetico**	
Metrica	**CER**	**WER**	**CER**	**WER**
BaseLine	92.52318911	99.75797723	82.342437	100

Se evidencio que el modelo por defecto de Easy OCR no realizó un reconocimiento acertado de ninguna de las palabra y adicional erro en la gran mayoría de los caracteres mostrando una baja capacidad de reconocimiento sobre datos escritos a mano.

5.2 VALIDACIÓN

El dataset original se dividió en tres partes y de la siguiente manera:
Datos de entrenamiento: se refiere a los datos que sirven para transportar el aprendizaje de la máquina de manera que se pueda generalizar, en este caso se utilizaron el 80% de los datos del dataset original.
Datos de test: Se refiere a los datos que sirven para comprobar si se puede generalizar el aprendizaje obtenido en el entrenamiento, en este caso se utilizaron el 10% de los datos del dataset original.
Datos de validación: se refiere a los datos que sirven para comprobar si se pueden obtener resultados similares al test, en este caso se utilizaron el 10% de los datos del dataset original.

Para el proceso de validación, se utilizaron dos métodos debido a que la librería de EasyOCR ya tiene una funcion de validacion predeterminada, la ICDAR2019 Normalized Edit Distance (De La Higuera & Micó, 2008), que es una métrica de evaluación utilizada para medir el rendimiento del modelo, esta métrica se basa en la métrica de edit distance, que mide el número de operaciones necesarias para transformar una cadena de caracteres en otra normalizando el número de operaciones de edición en función del tamaño de las cadenas de caracteres, lo que permite una comparación justa y por otro lado tenemos las métricas WER y CER mencionadas anteriormente que nos permite de manera similar obtener el error a partir del número de caracteres y palabras errados en cada salida del modelo.

5.3 Data augmentation

Se utilizó la metodología de data augmentation sobre una parte del dataset, los datos escogidos fueron las palabras que contenían los caracteres que menos se repetían en el dataset, en este caso, las palabras que contenían alguno de los caracteres **k, q, y, w** y **x**. A partir de este proceso de data augmentation se generaron imágenes adicionales para complementar el dataset final.
dando así un total de cuatro imágenes por cada manuscrito.

Para generar el proceso de data augmentation se utilizó un proceso offline para así poder seleccionar los datos , con la cual se realizaron:
- rotaciones en el rango de -10 a 10 grados
- escalamientos de imágenes en el rango de 0.98 a 1.02
- traslaciones de imágenes en el rango de -1 a 1 pixeles
- cambios de brillo en el rango de -0.1 a 0.1

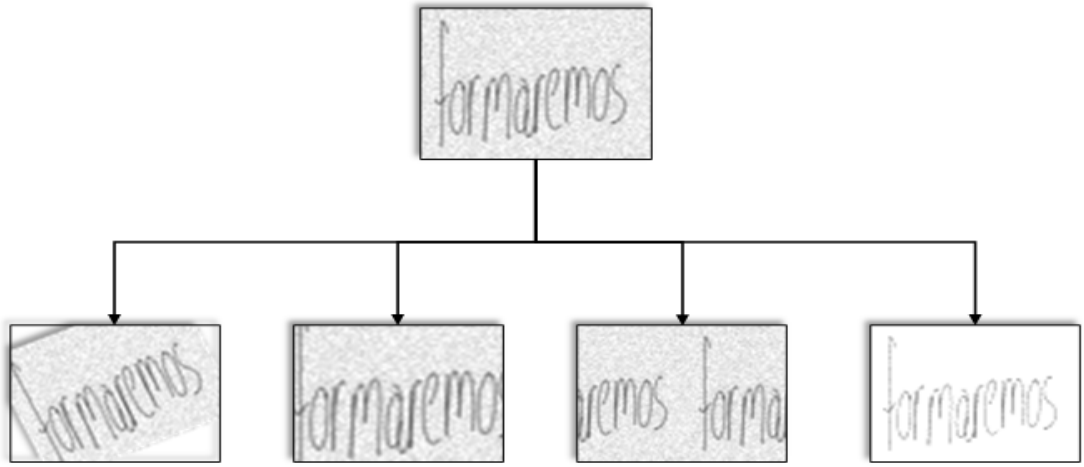


Figura 8. *Gráfico ilustrativo de las transformaciones para aumentar nuestro dataset haciendo uso del método de data augmentation.*

5.3 ITERACIONES y EVOLUCIÓN

Cada una de las iteraciones ejecutadas se realizan teniendo en cuenta la premisa que el dataset sea de alguna manera más robusto respecto a la iteración anterior.

Las iteraciones se ejecutaron en el siguiente orden:
Iteración 1 (Baseline)
La primera iteración consiste en el proceso mencionado en 5.1 BASELINE

Iteración 2
Se genera un modelo a partir de la arquitectura base utilizando únicamente los datos propios generados de manera manual.

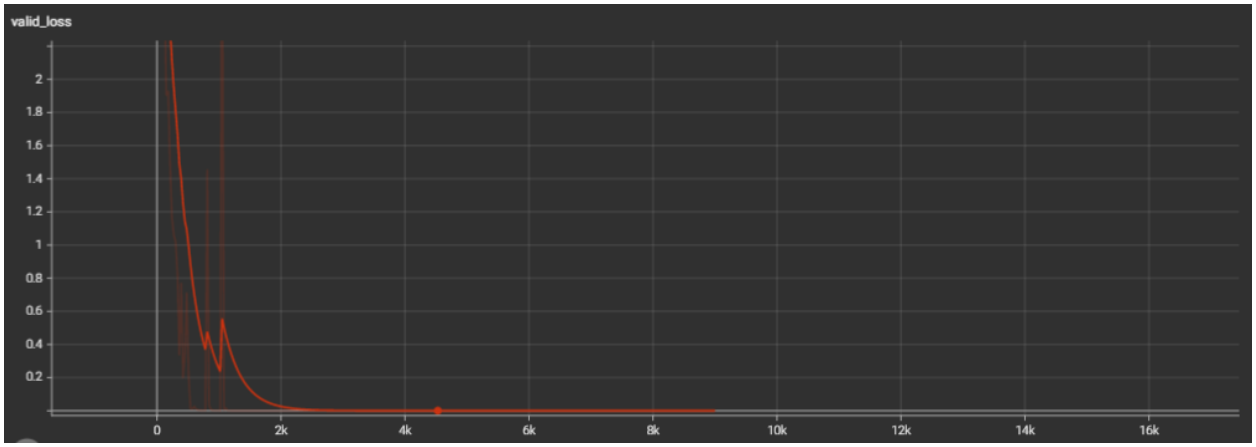


Figura 9. *Gráfica de la función de pérdida del modelo general cuando se entrenó únicamente con la base de datos propia.*

Iteración 3
Se genera un modelo a partir de la arquitectura base utilizando únicamente los datos generados de manera sintética.

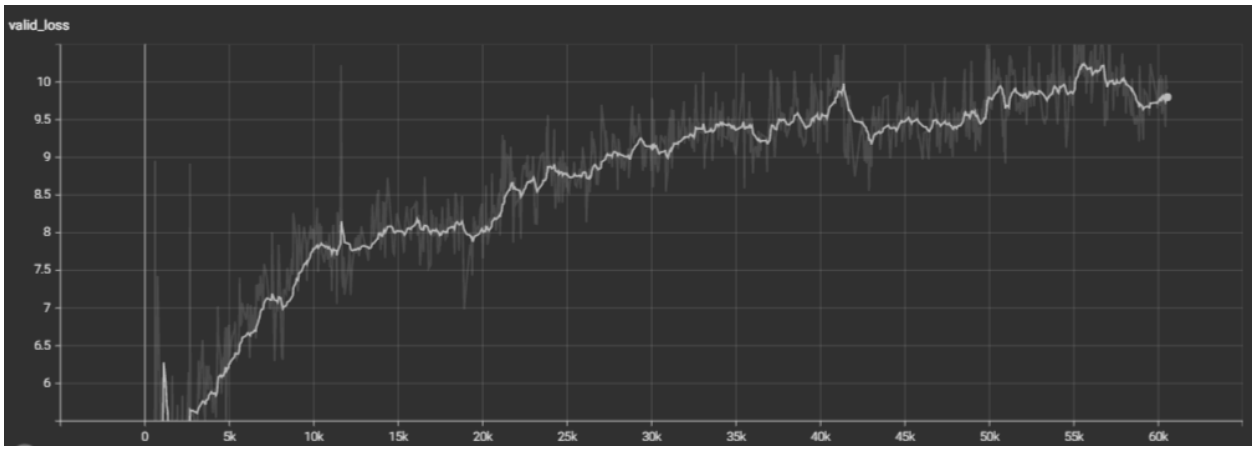


Figura 10. *Gráfica de la función de pérdida del modelo general cuando se entrenó únicamente con la base de datos sintéticos.*

Iteración 4

Se reentrena el modelo aplicando la técnica de fine tuning y la técnica de data augmentation sobre las palabras que contienen los caracteres con una baja cantidad de datos mencionados en 5.3 Data augmentation, esta técnica se aplica sobre el dataset comprendido por los datos sintéticos más los datos propios.

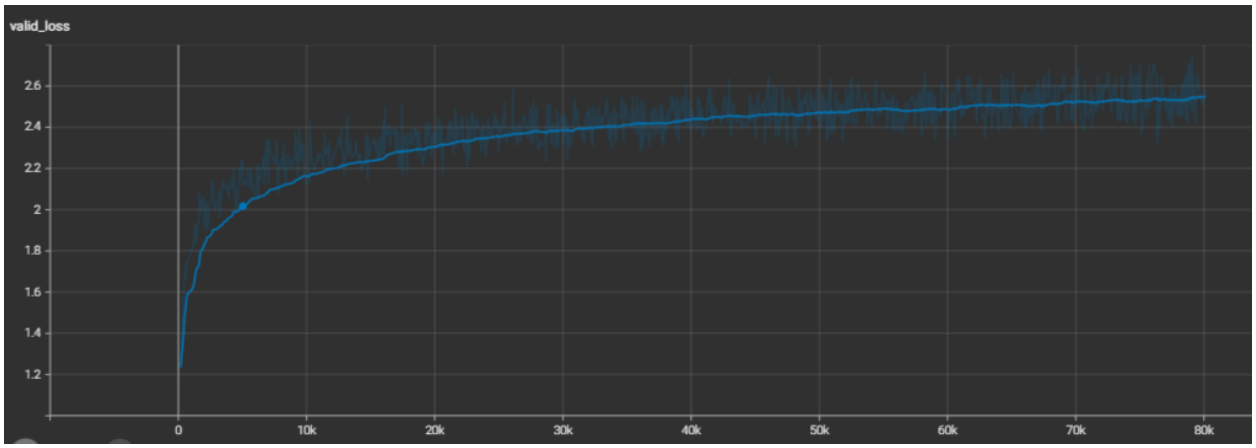


Figura 11. *Gráfica de la función de pérdida del modelo final*

5.4 HERRAMIENTAS

Para el desarrollo del proyecto se utilizaron las siguientes herramientas, separadas en las siguientes clases:

Para las pruebas, implementación y despliegue, se utilizó **Docker**.

Para el desarrollo en forma de RESTful API, se utilizó el framework de Python FastAPI.

Para el desarrollo y construcción del modelo se utilizaron las siguientes librerías de python:

EasyOCR, Cuda-python, torch, torchvision>=0.5, opencv-python-headless, scipy, numpy, Pillow, scikit-image, python-bidi, PyYAML y fastwer.

6. RESULTADOS

6.1 MÉTRICAS

MODELO	Dataset	Métricas con dataset propio		Métricas con dataset sintético	
		CER [%]	WER[%]	CER [%]	WER [%]
Modelo 1 entrenado con dataset de manuscritos propios	Train	76.289	98.020	85.377	100.000
	Test	82.130	99.224	84.921	100.000
	Validation	86.272	99.889	89.594	100.000
Modelo 2 entrenado con dataset sintético	Train	94.595	100.000	80.842	97.515
	Test	95.595	100.000	78.585	99.888
	Validation	97.355	100.000	86.345	99.752
Modelo 3 entrenado con dataset sintéticos + datos propios + data augmentation	Train	18.043	61.771	37.755	88.875
	Test	12.755	53.442	37.338	86.554
	Validation	16.973	55.173	46.836	90.341

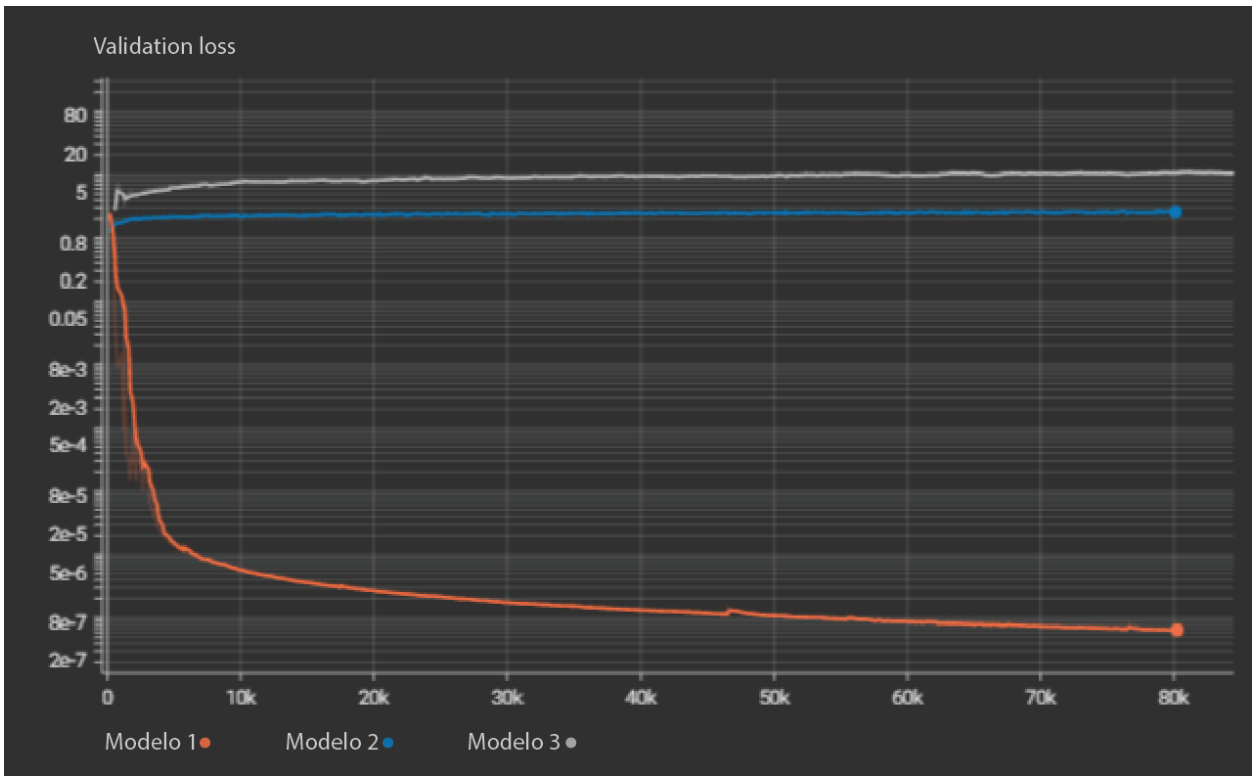


Figura 12. *Gráfica de la función de pérdida de los tres modelos*

Los resultados de los modelos nos muestran que el modelo 1 entrenado con datos de manuscritos propios tiene un rendimiento muy bajo en el WER, principalmente frente al dataset de datos sintéticos, esto es probablemente debido a que el dataset es pequeño y no contiene suficientes datos para aprender los patrones necesarios para el reconocimiento de los caracteres embebidos en largas frases. El modelo 2 entrenado con datos sintéticos tiene un rendimiento mejor, esto se debe a que el dataset sintético es mucho más grande y contiene más ejemplos de los caracteres que se están tratando de reconocer, lo que le da un mejor desempeño cuando se evalúa con un dataset generado sintéticamente, sin embargo al evaluar los datos propios que contienen un componente humano muy alto, predice los caracteres muy mal. El modelo 3, que fue entrenado con datos sintéticos y propios, así como con data augmentation, tiene el mejor rendimiento. Esto se debe a que el modelo tiene más datos para aprender, lo que le permite aprender mejor los patrones necesarios para el reconocimiento de los caracteres.

6.3 CONSIDERACIONES DE PRODUCCIÓN

Para la puesta en producción del modelo, se planteó el siguiente esquema:

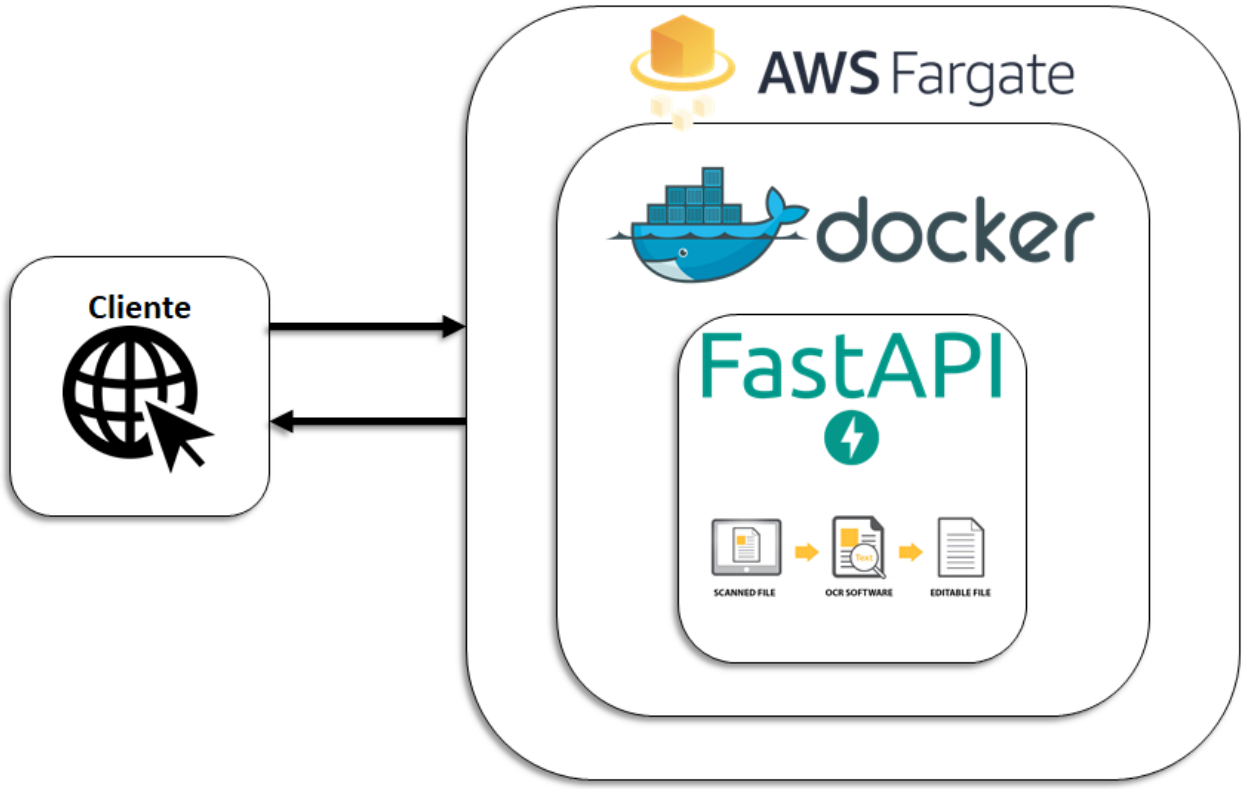


Figura 13. *Arquitectura planteada para despliegue del modelo como servicio.*

Para la puesta en producción del modelo, se propone una arquitectura teniendo como pilar la tecnología AWS Fargate y Docker.

AWS Fargate es un compute engine para Amazon ECS y EKS que aísla y abstracta el servidor, eliminando la necesidad de administrar clústeres de contenedores. Al usar Fargate, nos centramos en el código del digitalizador de caracteres y dejar que AWS se encargue de la infraestructura subyacente.

Para la imagen de nuestra aplicación utilizamos Docker, esta es una plataforma de aplicaciones portátil que le permite desplegar aplicaciones en cualquier entorno, ya sea en la nube, en un servidor local o en una PC, en este caso el objetivo es desplegarla en AWS Fargate.

Como framework principal de nuestra aplicación utilizamos Fast API, sin duda la mejor opción para el desarrollo de aplicaciones web basadas en Python. Fast API nos permite exponer nuestro servicio en un endpoint de fácil acceso para las personas interesadas que simplemente haciendo un POST de la imagen o manuscrito a digitalizar y un REQUEST, pueden obtener el texto editable.

Teniendo clara la arquitectura planteada para el despliegue de nuestra aplicación, se decidió revisar los siguientes factores de importancia.

1. Costo: El costo de ejecutar un modelo de machine learning en AWS fargate puede ser significativo, sin embargo teniendo en cuenta los precios actuales y conociendo las particularidades de nuestro modelo, podemos calcular un costo aproximado de 0.0135 USD por cada hora de uso de nuestro modelo y teniendo en cuenta las pruebas de tiempo de nuestro modelo, podríamos dar un costo aproximado de 0.0000075 USD por cada una de las palabras digitalizadas, asumiendo que solo se digitalizara una palabra, sin embargo a mayor cantidad de palabras que tenga cada manuscrito, menor es el costo por palabra del despliegue del modelo.

2. Tiempo de ejecución: AWS fargate puede ejecutar un modelo de machine learning de forma muy rápida. Sin embargo, es importante tener en cuenta el tiempo de ejecución del modelo antes de desplegarlo, por esta razón, se corrieron pruebas de tiempo en el modelo y se obtuvieron los siguientes datos:

Una palabra : 1.86 s ± 84.2 ms per loop

Manuscrito con veinte palabras: 2.05 s ± 74.4 ms per loop.

Manuscrito con cincuenta palabras: 2.23 s ± 49.4 ms per loop

Manuscrito con cien palabras: 2.56 s ± 128 ms per loop

Manuscrito con quinientas palabras: 3.88 s ± 64.8 ms per loop

3. Fiabilidad: AWS fargate ofrece un nivel de fiabilidad muy alto para los modelos de machine learning, sin embargo hay que realizar una constante actualización de la aplicación con la finalidad de que siempre funcione correctamente, por lo tanto se propone realizar un componente fuerte de pruebas unitarias y pruebas integrales con el objetivo de cubrir la mayor cantidad de brechas que pueda tener nuestra aplicación a desplegar.

4. Escalabilidad: Una gran ventaja de AWS fargate, es que nos permite escalar el tamaño y la complejidad de un modelo de machine learning de forma muy sencilla, sin embargo se propone plantear un plan de escalabilidad adecuado, siempre teniendo en cuenta el crecimiento de los costos por ejecución y almacenamiento.

7. CONCLUSIONES

El desbalance de los datos evidenciado en la poca presencia de algunos caracteres más usados es un problema que se debe trabajar, ya que al tener un desbalance tan marcado como el que se muestra en la **Figura 1,** no se puede asegurar que el modelo "aprenda" de manera confiable estos caracteres.

Se evidencio que es de suma importancia entrenar el modelo de OCR con múltiples tipografías a mano alzada debido a que es un problema común que diferentes personas escriben caracteres diferentes de una manera similar, esto se vio evidenciado en casos en donde en tipografía 1, se presentaba el carácter **a** y en la tipografía 2 se presentaba el carácter **q**, pero visualmente eran muy similares y al no tener más variedad en el dataset, este problema continuó persistiendo.

También es importante identificar que el modelo en ocasiones presenta problemas para separar palabras, ya que al momento de evaluar el modelo, este unía dos frases diferentes y en algunos casos separaba palabras debido al espaciamiento entre algunos caracteres producto de la naturaleza del escritor.

Se evidencia una buena confianza en la cantidad de caracteres que lograba determinar el modelo final, entrenado con fine tuning logrando solucionar un gran problema que presentaban los modelos iniciales en donde la cantidad de caracteres sin importar si eran errados o no, no era la misma cantidad de caracteres de la palabra o frase inicial.

Al implementar data augmentation el modelo disminuye su velocidad de entrenamiento sustancialmente ya que la cantidad de datos aumenta en gran medida, sin embargo es una técnica que nos permite reducir un poco el overfitting del modelo y se evidencia mucho cuando el dataset no es muy grande, como lo era nuestro dataset de datos propios.

La evolución del modelo de OCR muestra una capacidad de aprendizaje creciente, sin embargo se recomienda continuar entrenando con una base de datos más robusta y con una máquina con una mejor de cómputo con el fin de optimizar los tiempos.

8. BIBLIOGRAFIA

Baek, Y., Lee, B., Han, D., Yun, S., & Lee, H. (2019). Character Region Awareness for Text Detection. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2019-June, 9357–9366. https://doi.org/10.48550/arxiv.1904.01941

Belval, E. (2020, May 11). TextRecognitionDataGenerator: A synthetic data generator for text recognition. https://github.com/Belval/TextRecognitionDataGenerator

de La Higuera, C., & Micó, L. (2008). A contextual normalised edit distance. Proceedings - International Conference on Data Engineering, 354–361. https://doi.org/10.1109/ICDEW.2008.4498345

JaidedAI. (2021). EasyOCR: Ready-to-use OCR with 80+ supported languages and all popular writing scripts including Latin, Chinese, Arabic, Devanagari, Cyrillic and etc. https://github.com/JaidedAI/EasyOCR

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2016-December, 770–778. https://doi.org/10.48550/arxiv.1512.03385

Keates, S. (2017). Measuring acceptable input: What is “good enough”? Universal Access in the Information Society, 16(3), 713–723. https://doi.org/10.1007/S10209-016-0498-4

Shi, B., Bai, X., & Yao, C. (2015). An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(11), 2298–2304. https://doi.org/10.48550/arxiv.1507.05717