

PROYECTO DE VISIÓN POR COMPUTADOR

TEXT DETECTION AND RECOGNITION IN THE WILD



Autores:

Laura García García
Victoria Nikitina
Zuleika María Redondo García
Iñigo Zárate Rico

Visión por computador
Grado en Ingeniería robótica
Universidad de Alicante
Curso 2021-2022

ÍNDICE

1. Introducción y objetivos -----	02
2. Metodologías -----	03
3. Datos utilizados -----	10
4. Experimentos y resultados -----	10
4.1 Método de detección -----	11
4.2 Métodos de reconocimiento -----	15
4.3 Métodos de detección + reconocimiento -----	17
5. Modelo ResNet -----	24
6. Conclusiones -----	29
7. Bibliografía -----	30

1. INTRODUCCIÓN Y OBJETIVOS

La extracción del texto en las imágenes presenta numerosas aplicaciones como el reconocimiento de documentos de identidad como el DNI o el pasaporte, número de matrículas, señales, entre otras muchas. Sin embargo, aunque a simple vista parece una labor relativamente sencilla para el ojo humano, no lo es para nada cuando las ha de realizar una computadora. Esta tarea la podemos subdividir en otras dos, por un lado está la detección de texto, que consiste en marcar la zona de la imagen donde se encuentra, y por otra parte está el reconocimiento del mismo, que se basa en identificar qué es lo que pone.

Antes de profundizar más en por qué estas son dos tareas arduas, debemos tener en cuenta que el tipo de texto que se puede pretender reconocer puede ser de dos formas: Estructurado y no estructurado. Por un lado, el texto estructurado es aquel que aparece en los documentos mecanografiados, con una fuente estándar, y un fondo en blanco. Por el contrario, el no estructurado, es el que podemos encontrar en escenas cotidianas, por ejemplo carteles, señales, etc, este tipo de texto se caracteriza por tener una fuente no estándar, en fondos diversos y sin ningún tipo de estructura como tal. Para el primer caso de texto, llevar a cabo tanto la detección como el reconocimiento no resulta una tarea demasiado compleja. No obstante, esto se complica mucho más con el segundo tipo, es lo que se conoce como el problema de *“text detection and recognition in the wild”*, y será sobre el que nos vamos a centrar, aunque también analizaremos ejemplos del primer tipo de texto.

La dificultad de la detección y reconocimiento del texto en la naturaleza radica en que en este tipo de situaciones el texto puede aparecer con diferentes tipos de colores, tipografías, formas, orientaciones y tamaños. Además, de que los fondos sobre los que se puede encontrar puede dificultar estas tareas, y también hay que tener en cuenta otros factores como la luminosidad y la resolución de las imágenes. Por todo esto, estas tareas resultan bastante complicadas. [9]

El objetivo de este informe es, por tanto, realizar un estudio e investigación sobre las diferentes metodologías que existen para llevar a cabo este cometido, además de comparar y concluir cuáles son los métodos que dan mejores resultados.

Todo el código de los métodos usados en este trabajo se han recopilado en un repositorio de github. De este modo se ha podido ir coordinando el trabajo y se encuentra todo agrupado y explicada la forma de ejecutar cada método. El enlace al repositorio es el siguiente: https://github.com/Inigo183/scene_text_recognition

2. METODOLOGÍAS

Para llevar a cabo el cometido que nos atañe, que es extraer texto de imágenes, primero, se detectan las apariencias del texto en la imagen, en nuestro caso, texto en la naturaleza.

Después de detectar el nivel de línea / palabra, podemos elegir entre varias soluciones, que provienen de tres enfoques principales:

1. Técnicas clásicas de visión artificial.
2. Aprendizaje profundo especializado.
3. Enfoque estándar de aprendizaje profundo (detección).

-Técnicas clásicas de visión artificial

Los pasos a seguir serían:

1. Aplica filtros para que los personajes se destaquen del fondo.
2. Aplique la detección de contorno para reconocer los caracteres uno por uno.
3. Aplicar clasificación de imágenes para identificar a los caracteres.

Tanto si se aplican mejor los filtros y la detección, el tercer paso, será mucho más sencillo.

Sin embargo, cabe destacar que la detección correcta de las palabras, es una tarea complicada, sobre todo, cuando la imagen está movida y/o las letras muy juntas.

-Aprendizaje profundo especializado

Este es un método más utilizado que el anterior dado que proporciona mejores resultados, los más comunes son EAST, CRNN, que se comentan posteriormente.

-Enfoque estándar de aprendizaje profundo (detección).

Después de detectar las "palabras" podemos aplicar enfoques estándar de detección de aprendizaje profundo, como SSD, YOLO y Mask RCNN.

Sin embargo, SSD y otros modelos de detección enfrentan desafíos cuando se trata de clases densas y similares. Es decir, les cuesta mucho más detectar texto o números que objetos, que a priori parece ser una tarea más desafiante. [16]

2.1. DETECCIÓN

Dentro de los métodos de detección podemos hacer una distinción entre los que usan y no usan deep learning.

2.1.1. Sin deep learning

Maximal Stable Extremal Regions (MSER)

Propuesto por primera vez en 2004 por Matias *et al*, se trata de un método de detección de regiones (*blobs*) en la imagen, en el que aplicamos todos los umbrales posibles (*sliding threshold*), para combinar los píxeles en manchas más grandes.

Ahora vamos a definir una serie de conceptos. Por un lado el de *región extrema* o en inglés *extremal region*, ésta se trata de una región donde los píxeles que hay en su interior tienen una intensidad mayor o menor que la que presentan los que están situados en el borde de la región. Y por otro lado, las *maximal regions*, son todas aquellas regiones que conservan más o menos el mismo tamaño ante diversos umbrales. [4]

Una vez visto esto, podemos pasar a explicar en qué consiste el algoritmo MSER. En primer lugar, se aplican todos los umbrales posibles obteniendo así distintas imágenes binarias. Y para cada uno de estos umbrales se calculan las regiones que presentan píxeles con la misma intensidad en las imágenes binarias (regiones conectadas o *blobs*). Luego para cada una de ellas, se aplica una función, y si una región permanece estable durante una serie de niveles de gris se dice que es una región de interés. [14]

A pesar de que este método no fue desarrollado con el objetivo de realizar detección de texto si que es capaz de detectar caracteres. Cabe destacar que para su uso en la detección de texto se combina con la detección de bordes Canny, donde primero se aplica el algoritmo MSER para determinar las regiones donde están los caracteres y con Canny se eliminan los píxeles que están fuera de los límites formados por los bordes Canny. También, se puede usar MSER para la detección de texto combinándolo con el método de grafos. Además, se encuentra implementado en OpenCV. [15]

2.2.2. Con deep learning

EAST

EAST (*Efficient and Accuracy Scene Text detection*), se trata de un método muy robusto basado en deep learning que se usa exclusivamente para llevar a cabo la detección de texto, siendo capaz de encontrar cuadros delimitadores (*bounding boxes*) tanto horizontales como rotados tanto en imágenes como en vídeo. Además, de que se puede usar en combinación con cualquier método para reconocimiento de texto.

Se trata de un método que elimina pasos intermedios como la partición de palabras, o la formación de regiones de texto , incluyendo así solo en el post procesamiento la umbralización y NMS (sistema de gestión de red o *Network Management System*)¹. Por tanto, lleva a cabo las siguientes dos etapas:

En primer lugar, usa una red convolucional completa (FCN)², para generar directamente predicciones de líneas de texto o palabras. [1]

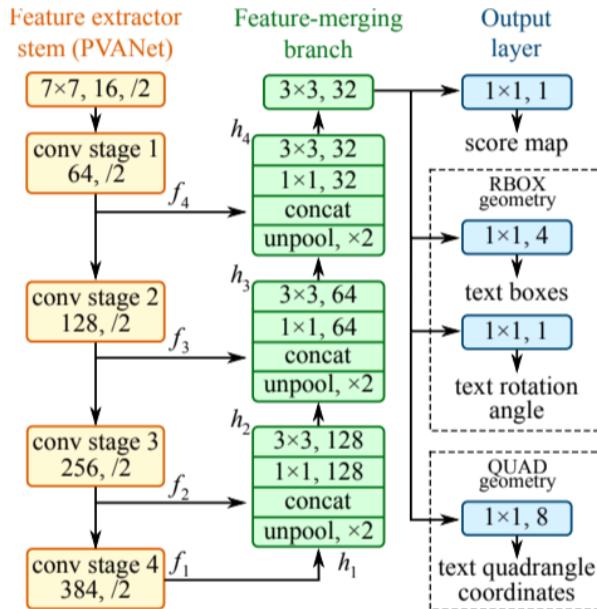


Figura 1 . Estructura del modelo FCN

Una vez que han tenido lugar las predicciones de texto, es decir, se han generado los *bounding boxes*, la salida se filtra haciendo uso de la técnica *Non Maximum Suppression* (NMS), que se trata de un algoritmo que selecciona una entidad entre todas las que hay, en este caso selecciona entre todas las *bounding boxes*, una que será la que luego en la imagen de salida marcará dónde está el texto. [5]

¹ **NMS:** Aplicación o conjunto de aplicaciones que permite a los administradores de red monitorizar o administrar todas sus operaciones de red haciendo uso únicamente de un sistema. Permitiendo así ahorrar dinero, tiempo, productividad y facilitando la detección de fallos. [2]

² **FCN:** Clasifica las imágenes a nivel de píxel. A diferencia de la CNN, puede aceptar imágenes de entrada que tengan cualquier tamaño, y usar una capa desconvolucional para la última capa convolucional. [3]



Figura 2. Aplicación de NMS

En la siguiente figura se muestra de forma esquemática el proceso descrito del método EAST:

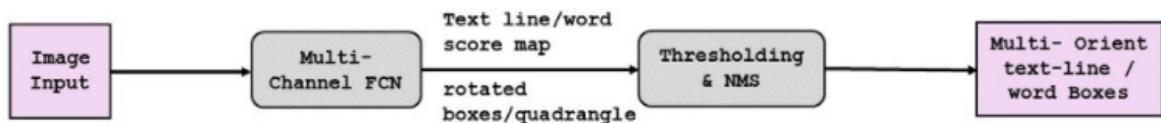


Figura 3. Estructura del algoritmo EAST

Cabe destacar que en OpenCV podemos encontrar este método ya pre-entrenado.

2.2. RECONOCIMIENTO

Una vez que el texto se ha detectado en la imagen, el siguiente paso es el reconocimiento del mismo.

CRNN

Las redes convolucionales recurrentes (CRNN), se trata de una combinación de tres cosas: redes convolucionales (CNN), redes recurrentes (RNN) y CTC (*Connectionist Temporal Classification*) para tareas de reconocimiento de secuencias basadas en imágenes, como reconocimiento de texto en escenas y reconocimiento óptico de caracteres (OCR).

Hay que destacar que la arquitectura que presenta esta red neuronal en comparación con otros sistemas anteriores usados para reconocer texto en escenas presenta una serie de características que la hacen diferente: [8]

- En la gran parte de los algoritmos que llevan a cabo esta tarea los componentes se entranan y ajustan por separado. Sin embargo, ésta es entrenable de extremo a extremo.

- No se ciñe a un léxico predefinido, además de que da buenos resultados tanto en imágenes con y sin texto.

Genera un modelo efectivo pero mucho más pequeño que es más práctico para aplicaciones reales.

Teniendo esto en cuenta ahora se va a pasar a describir brevemente en qué consiste su arquitectura. En primer lugar, actúa la red CNN, que extrae una secuencia de características de las imágenes de entrada. Hay que tener en cuenta que estas imágenes de partida son el texto ya recortado que se ha sido detectado previamente con cualquier otro método. Esta primera parte es lo que en la figura aparece como *Convolutional Layers*.

Tras esto, pasamos a una segunda etapa, *Recurrent Layers*, donde se utiliza la red RNN, con una arquitectura LSTM³. En esta parte, a partir de las características extraídas antes se predice una distribución de etiquetas para cada frame. Finalmente, tiene lugar la capa de transcripción o *transcription layer*, en la cual se transcriben las predicciones de cada uno de los frame en la secuencia de texto final. [6] [7]

El resumen de cada una de las fases que tienen lugar se puede ver de forma más esquemática en la figura 4:

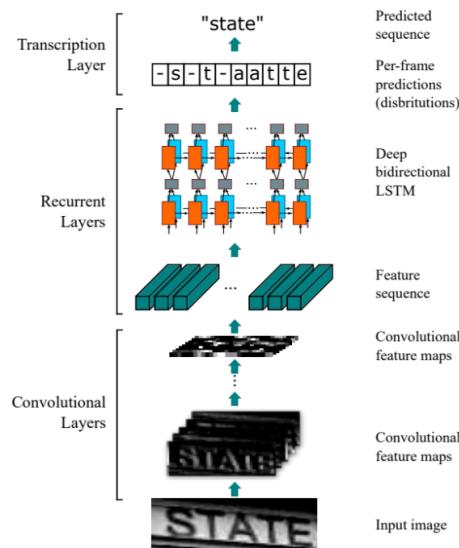


Figura 4. Arquitectura CRNN

³ **LSTM** (long short-term memory): arquitectura de una red neuronal recurrente, que se diferencia de las redes recurrentes estándares, las LSTM tienen una memoria a más largo plazo.

2.3. DETECCIÓN Y RECONOCIMIENTO

Además, también existen métodos que realizan ambas tareas, tanto la de detectar como la de reconocer el texto en las imágenes.

EasyOCR

Creado por la empresa Jaide AI, este paquete para implementar un OCR está construido con python y la biblioteca de deep learning Pytorch, se trata por tanto de un modelo ya pre entrenado. Hay que destacar que es capaz de reconocer texto en más de 70 idiomas, como inglés, español, hindi, coreano, entre otros muchos, y que si se posee una GPU , el proceso de detección se realizará bastante más rápido, estás dos indicaciones tanto el idioma como si se tiene GPU, son dos parámetros que se le pasan al algoritmo, entre otros. [10]

En cuanto a su arquitectura, podemos distinguir las siguientes etapas. En primer lugar, hay una fase de pre-procesamiento, donde tiene lugar la detección del texto. En esta parte se usa como método CRAFT (*Cascade Region-proposal-network And Fast CRNN*), que se trata de un método de segmentación que no realiza una clasificación a nivel de píxel de toda la imagen, sino que lo que hace es ubicar de precisa cada carácter y luego conectar todos los caracteres detectados en un texto, para realizar así la detección. Este método es adecuado para curvas, deformadas o texto muy largo, ya que solo presta atención a los caracteres y la distancia entre ellos. [11]

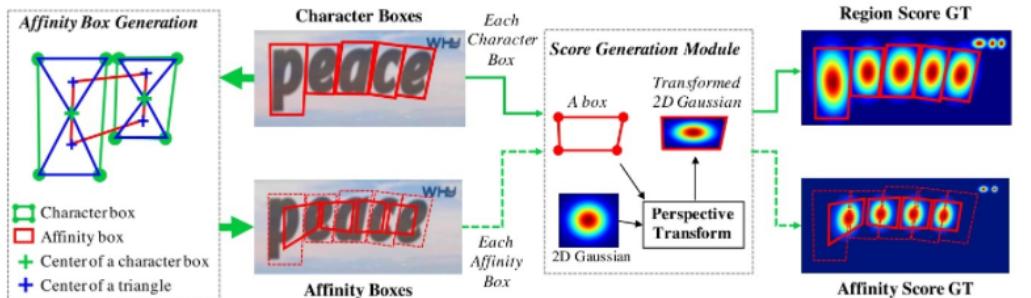


Figura 5. Método CRAFT

La siguiente parte del método es la fase de reconocimiento que se trata de una CRNN, cuyo funcionamiento ya se ha explicado más arriba.

En la figura siguiente aparece un esquema de la arquitectura de EasyOCR:

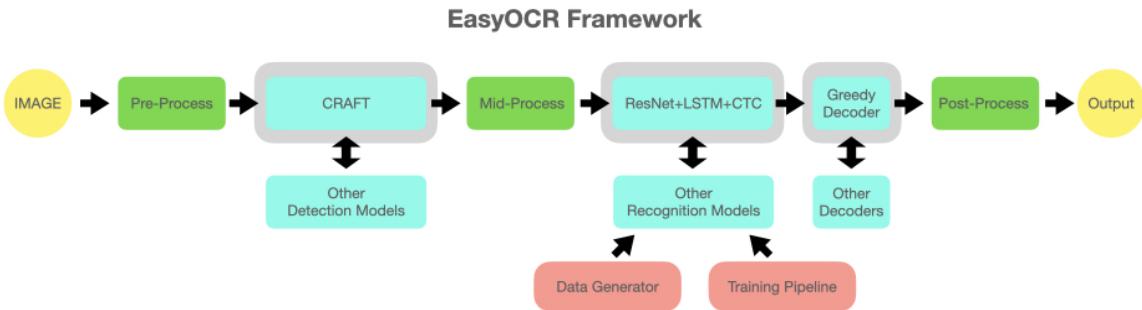


Figura 6. Arquitectura de EasyOCR

Tesseract OCR

Se trata de un motor de OCR de código abierto que fue desarrollado por HP entre 1984 y 1994. En 1995 fue considerado como uno de los OCR disponibles del mercado que aportan mayor precisión. Y ya desde 2006 su desarrollo ha sido financiado por Google.

Las primeras versiones de Tesseract, solo podían aceptar como imágenes de entradas aquellas que tuvieran una sola columna en formato TIFF⁴. Ya a partir de la versión 3, gracias a la incorporación de la biblioteca Leptonica se consigue compatibilidad con nuevos formatos de imagen. Y ya a partir de la versión, se añade deep learning con la incorporación de un OCR basado en la red LSTM, que se centra en el reconocimiento de líneas, además de que permite el reconocimiento de texto en 116 idiomas. [6][13]

En cuanto a su arquitectura, Tesseract necesita que las imágenes que le pasemos se conviertan a escala de grises. Una vez hecho esto, las regiones (*blobs*) se organizan en líneas de texto que se dividen en palabras según el tipo de espacio que hay entre los caracteres, y las líneas y regiones se analizan.

Una vez hecha la fase de detección, tiene lugar el reconocimiento, donde entra en juego la red LSTM. En esta fase distinguimos dos etapas (o pasadas). En la primera, se intenta reconocer cada palabra, cada palabra correcta se pasa a un clasificador adaptativo que sirve como datos de entrenamiento. En la segunda pasada, se intenta reconocer aquellas palabras que no se detectaron bien en la primera. [12]

En la figura 7 se resume lo que se acaba de explicar:

⁴ **Formato TIFF:** Es un formato de archivo de imagen empleado normalmente para trabajar con datos de imagen sin procesar.

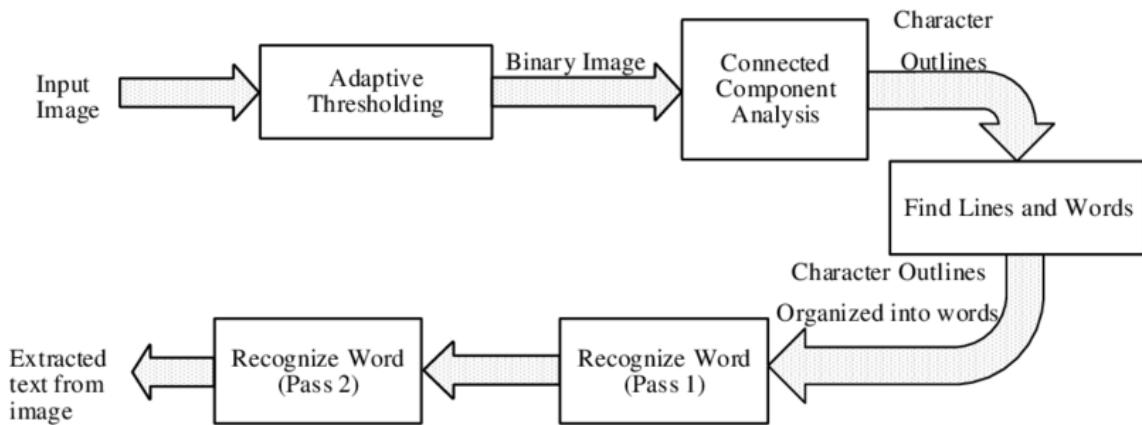


Figura 7. Arquitectura de Tesseract OCR

3.DATOS UTILIZADOS

En cuanto a los datasheet que se han utilizado debemos hacer distinción según el método. De esta manera :

- Para el caso de **MSER**, al no tratarse de un método que requiera deep learning no necesita ser pre-entrenado. Y para **EAST**, **EssyOCR** y **Tesseract OCR**, estos tres métodos ya pre-entrenados no se han encontrado con que datasheet habían sido entrenados. En el apartado *4.Experimentos y resultados*, las imágenes que se han usado para testear han sido sacadas directamente de internet.
- Para **CRNN**, el datasheet que se ha usado para el método pre-entrenado que hemos encontrado para el caso de la red RNN es Synth90k, que se trata de un datasheet con 9 millones de imágenes que abarcan 90.000 palabras en inglés, e incluye imágenes de entrenamiento, validación y test. Por otro lado, para testear este método a partir del modelo ya implementado se ha usado un datasheet (ch4_test_word_images_gt) con imágenes donde el texto aparece ya recortado, y que consta de 2077 imágenes.

4.EXPERIMENTOS Y RESULTADOS

En este apartado se va a analizar cada uno de los métodos expuestos en el apartado *2.Metodologías*, y aportaremos algunos ejemplos más concretos de imágenes para comentar los resultados que se obtienen al usarlos y sus ventajas e inconvenientes, también haremos una valoración final de precisión general de cada uno de los métodos. Cabe añadir que para poder realizar de mejor forma la comparación se han usado las mismas imágenes en los

distintos métodos, en todos salvo en el CRNN, ya que éste requiere que las imágenes que se aporten estén ya con el texto recortado.

4.1 MÉTODOS DE DE DETECCIÓN

Maximal Stable Extremal Regions (MSER)

RESULTADOS:

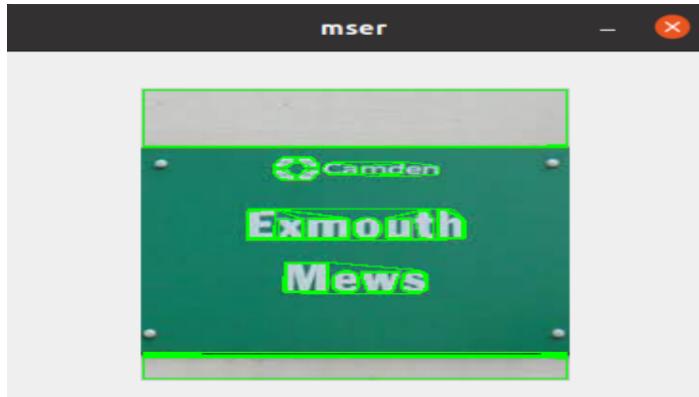


Imagen 1. MSER

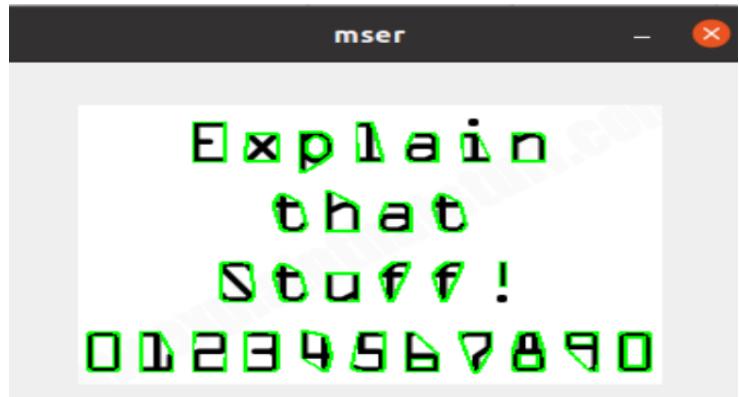


Imagen 2. MSER



Imagen 3. MSER



Imagen 4. MSER

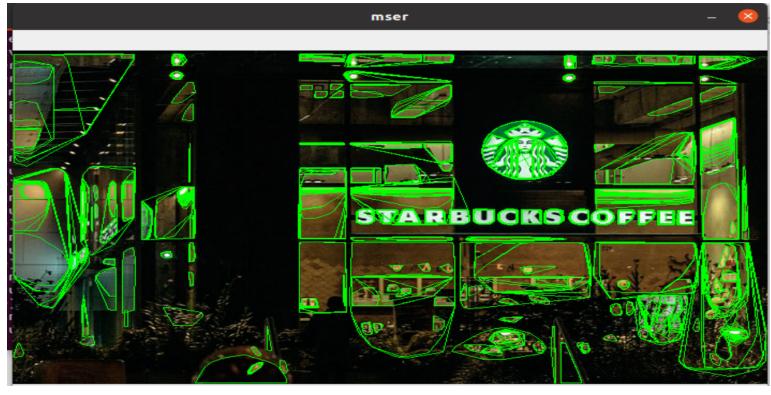


Imagen 5. MSER

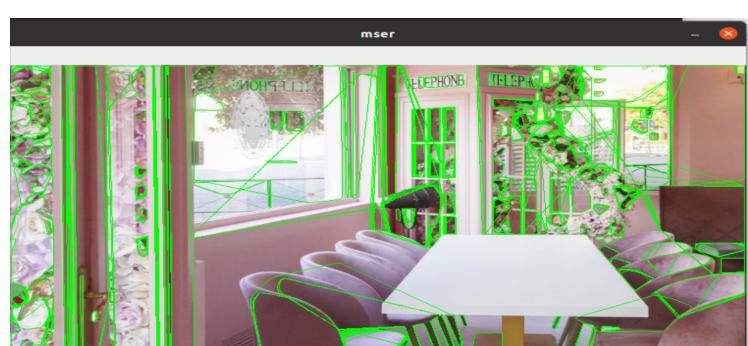


Imagen 6. MSER

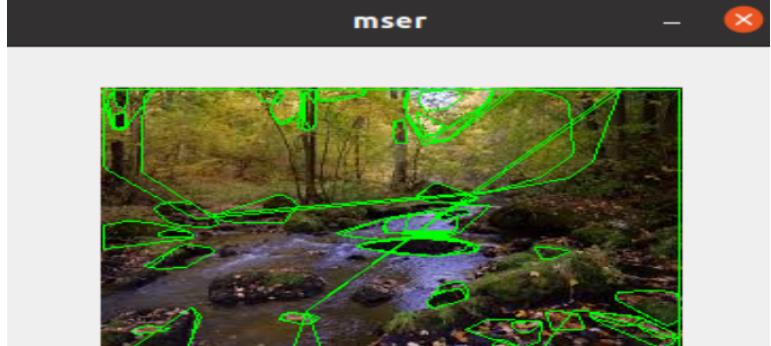


Imagen 7. MSER

ANÁLISIS DE RESULTADOS:

Se puede observar que este método funciona mejor cuando el texto es estructurado, como es el caso de las imágenes 2 y 3. Como se puede observar en estos dos casos cuando la distancia entre los caracteres es grande, MSER los detecta de forma individual, es decir, que no marca la palabra entera sino carácter a carácter.

Para el caso de la imagen 1, a pesar de ser un cartel, como la letra está a ordenador y en fondo monocromático, el resultado obtenido también es bueno, y en este caso a diferencia de la 2 y 3, al estar la letra más junta sí que lo detecta más o menos como si fuera una palabra. Aún así, marca algunas zonas a pesar de no tener texto en ellas.

En cuanto al resto de imágenes (4, 5, 6), cuando el texto es desestructurado y en escenas cotidianas (*in the wild*), vemos que MSER no da buenos resultados, ya que detecta muchos falsos positivos lo que hace que aparezcan muchas regiones marcadas a pesar de que no haya texto en ellas. E incluso en el caso de la imagen 7, en la que directamente ni hay texto marca regiones.

EAST

RESULTADOS:



Imagen 1. EAST

Explain
that
Stuff!
01234567890

Imagen 2. EAST



Imagen 3. EAST



Imagen 4. EAST



Imagen 5. EAST

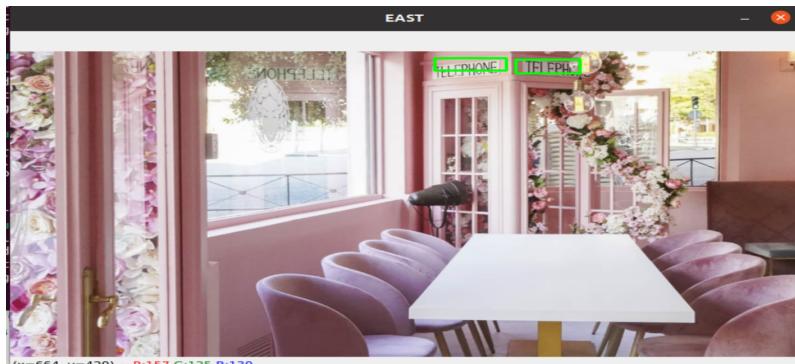


Imagen 6. EAST

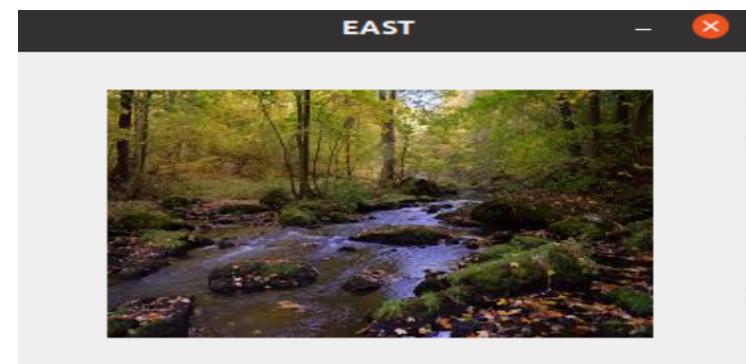


Imagen 7. EAST

ANÁLISIS RESULTADOS:

En este caso, se nota gran diferencia a la hora de detectar si lo comparamos con los resultados de MSER, y esto es gracias al uso del deep learning.

Como vemos, ahora en la imagen 2, donde el texto es estructurado, lo detecta correctamente, tanto letras como números, y en este caso a diferencia del MSER, detecta la palabra entera, marcándola con una bounding box, no como en MSER que detectaba carácter a carácter.

Para el caso de la imagen 3, la detección de a pesar de ser algo mejor que en MSER, sigue sin ser del todo correcta del todo, ya que marca dos palabras con una sola bounding box.

También, se nota mejoras en la imagen 1, donde ahora además de obtenerse la palabra perfectamente marcada, además marca zonas donde no hay texto, como hacia MSER.

En cuanto a las imágenes con texto no estructurado que están en escenas *in the wild*, tanto para el caso de la imagen 5 y 6, la detección se realiza perfectamente. No obstante, en la 4, a pesar de obtener mejores resultados que MSER, en este caso, marca zonas donde no hay texto.

Por último, vemos que para imágenes donde no hay nada de texto como la 7, en este caso, no aparece ninguna zona marcada.

4.2. MÉTODOS DE RECONOCIMIENTO

CRNN

Algunos ejemplos de reconocimiento de texto con CRNN cuyo resultado es favorecedor son los de las figuras de la 8 a la 13.



Figura 8: added



Figura 9: singapore



Figura 10: toffifee



Figura 11: joint



Figura 12: 154



Figura 13: boos

Por otro lado, se pueden observar otros resultados no tan bien conseguidos. Y se puede obtener algunas semejanzas entre ellas.

1. Si están borrosas e inclinadas (figura 14 y 15).



Figura 14: et



Figura 15: inonn

2. Si la palabra está incompleta, se ve cortada. En este caso, podría suponer que letra es la siguiente por la forma que se ve pero, no se asegura que sea el resultado correcto. Un caso demostrativo es el de la figura 16.



Figura 16: wati

3. Caracteres especiales. No los reconoce, por tanto, en algunos casos, imprime la letra que más se parece a este (figura 17 y 18).

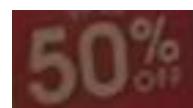


Figura 17: 50x



Figura 18: mec

4. Las palabras que están giradas 90° o más, como en las figuras 19 y 20.

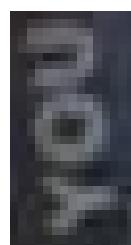


Figura 19: 3



Figura 20: r

5. En otras ocasiones no reconoce los números tan bien y se confunden unos con otros, como se observa en las figuras 21 y 22.



Figura 21: 191



Figura 22: fi

4.3. MÉTODOS DE DETECCIÓN + RECONOCIMIENTO

EasyOCR

RESULTADOS:



Imagen 1. EasyOCR

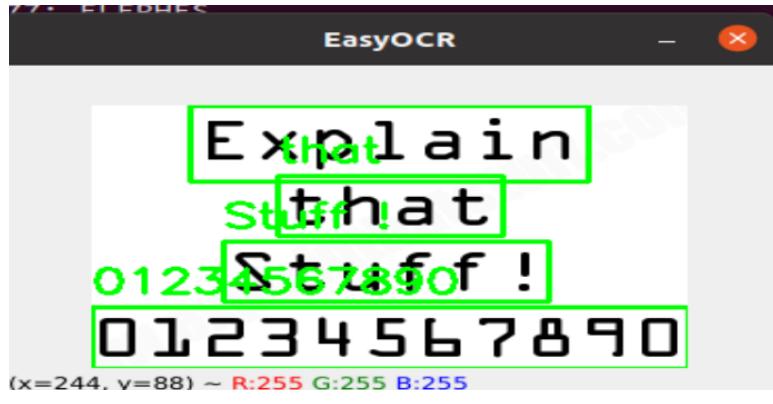


Imagen 2. EasyOCR



Imagen 3. EasyOCR

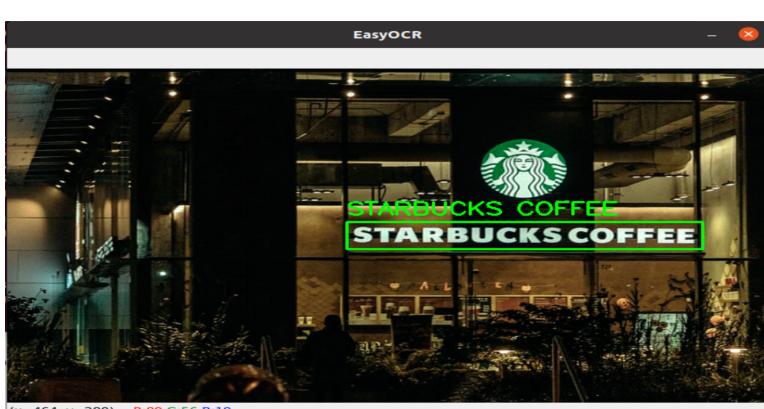


Imagen 4. EasyOCR



Imagen 5. EasyOCR

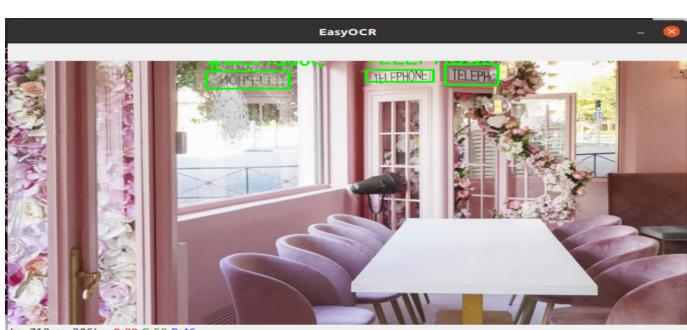


Imagen 6. EasyOCR

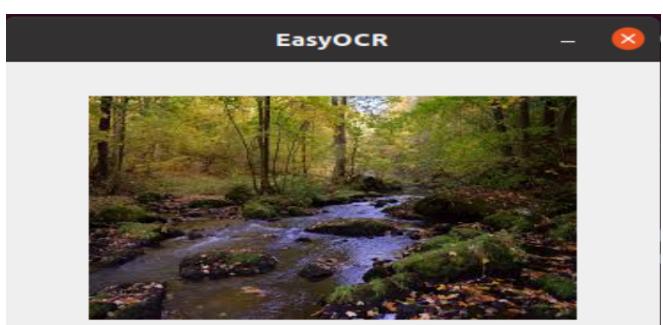


Imagen 7. EasyOCR

ANÁLISIS RESULTADOS:

Pasando ahora a la parte de reconocimiento, en líneas generales se puede decir que EasyOCR aporta unos resultados muy buenos.

Por un lado, para las imágenes con texto estructurado (imágenes 2 y 3), vemos que tanto la detección como reconocimiento se realizan de forma excepcional, llegando a reconocer caracteres, números e incluso otros caracteres especiales como símbolos de exclamación. En la imagen 1, la extracción del texto también se realiza favorablemente.

Para el caso de las imágenes con texto no estructurado (imágenes 4, 5, 6), también obtiene resultados exitosos. Algunos aspectos a destacar son:

- En la imagen 4, detecta y reconoce a la perfección tanto palabras extremadamente largas, como los carteles que aparecen más pequeños, y sin marcar ninguna zona incorrecta.
- En la imagen 6, llega a detectar hasta el reflejo del texto que aparece en el cristal, que es la palabra TELEPHONE al revés. No obstante, no la llega a reconocer bien, pero aún así es un aspecto a destacar, ya que es una palabra que no aparece muy clara, y ninguno de los detectores de texto nombrados anteriormente habían sido capaces de detectar hasta ese nivel. El resto del texto que aparece, a pesar de que no presenta una orientación completamente horizontal, no lo reconoce del todo bien, a pesar de detectarlo.

Por último, en la imagen sin ningún texto, la 7, EasyOCR, también actúa bien, ya que no marca ninguna región.

Tesseract

RESULTADOS:

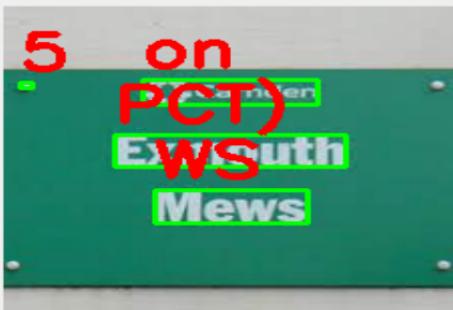


Imagen 1. Tesseract OCR



Imagen 2. Tesseract OCR



Imagen 3. Tesseract OCR

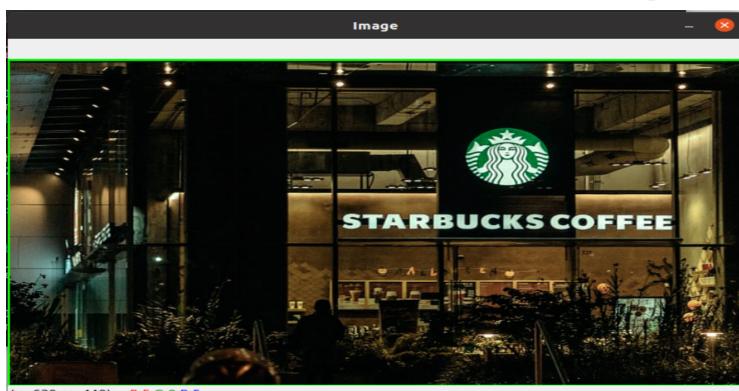


Imagen 4. Tesseract OCR



Imagen 5. Tesseract OCR

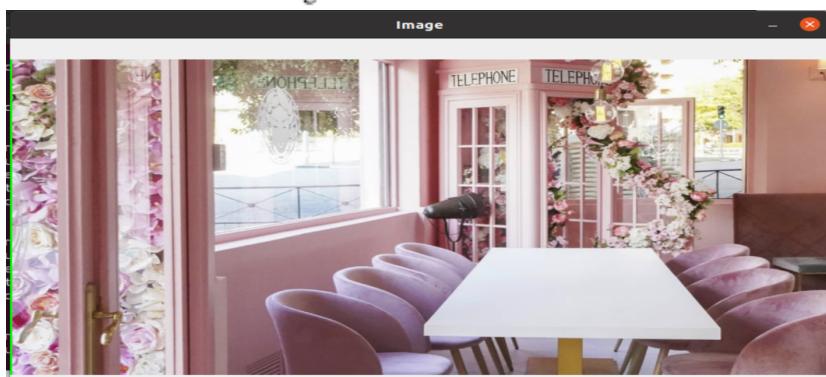


Imagen 6. Tesseract OCR

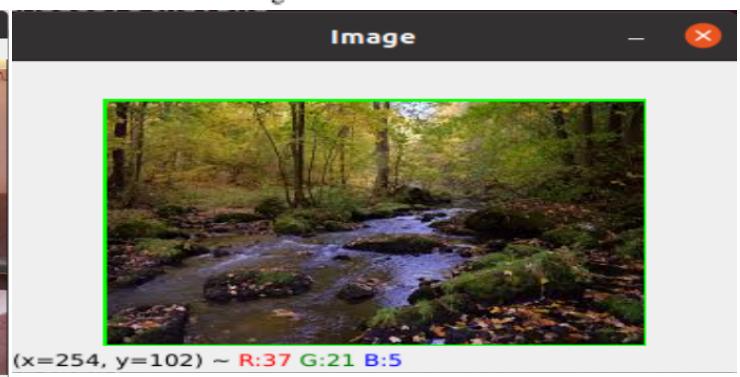


Imagen 7. Tesseract OCR

ANÁLISIS DE RESULTADOS:

Para este método, hemos podido observar que hay grandes diferencias entre la detección de texto estructurado y no estructurado.

Por un lado, para la imágenes 2 y 3, donde el texto es estructurado, Tesseract produce buenos resultados, sin embargo, la detección que realiza EasyOCR, es mejor, ya que por ejemplo para el caso de la imagen 2, Tesseract marca algunas regiones como que hay texto cuando en verdad no es así, a pesar de marcar estas regiones, luego la probabilidad que da de que ahí ponga algo, es muy baja. Por otro lado, también destacar, que Tesseract en texto estructurado es capaz de reconocer caracteres, números y caracteres especiales.

Por otra parte, cuando se trata de imágenes con texto no estructurado, Tesseract ya no da tan buenos resultados, ni en el caso de la imagen 1, donde el texto es mecanografiado y con fondo plano, ya que se observa que a pesar de realizar bien la detección, luego a la hora de reconocer no da buenos resultados. Además, también marca alguna zona incorrecta.

Y para el caso del reconocimiento en escenas, como en las imágenes 4, 5, 6 y 7, directamente no detecta ni reconoce, en estos casos que no detecta nada marca el borde de la imagen pero eso si no extrae el texto. Esto se debe a que Tesseract en imágenes donde el fondo no es tan claro, no da buenos resultados.

COMPARACIÓN DE LA PRECISIÓN: EasyOCR vs Tesseract OCR

A continuación, se van a comparar los resultados de ambos métodos de detección y reconocimiento, en cuanto la semejanza que presentan con el texto original, para cada uno de los ejemplos particulares vistos:

Texto imagen 1	EasyOCR Texto reconocido	EasyOCR accuracy (%)	Tesseract OCR Texto reconocido	Tesseract OCR accuracy (%)
Camden	Camden	0.5370	on	0.20
Exmouth	Exmouth	0.9999	pct)	0.10
Mews	Mews	1.0000	ws	0.29

COMENTARIOS IMAGEN 1:

- Tesseract confunde un tornillo de la esquina superior izquierda con un 5, con una fiabilidad del 0.07 %.
- EasyOCR, realiza hasta la distinción entre letras mayúsculas y minúsculas.

Texto imagen 2	EasyOCR Texto reconocido	EasyOCR accuracy (%)	Tesseract OCR Texto reconocido	Tesseract OCR accuracy (%)
Explain	Explain	1.0000	Explain	0.96
that	that	1.0000	that	0.96
Stuff!	Stuff!	0.8936	Stuff!	0.93
01234567890	01234567890	0.8977	01234567890	0.68

Texto imagen 3	Tesseract OCR Texto reconocido	Tesseract OCR accuracy (%)
Download	Download	0.96
on	on	0.95
the	the	0.96
App	App	0.96
Strore	Store	0.95

Texto imagen 3	EasyOCR Texto reconocido	EasyOCR accuracy (%)
Download on the	Download on the	0.9123
App Store	App Store	0.7743

COMENTARIOS IMAGEN 2:

- En este caso, hay que destacar que Tesseract realiza la detección palabra por palabra mientras que EasyOCR, lo hace línea a línea.
- En el caso de Tesseract, además marca alguna zona en la que realmente no hay texto pero con un porcentaje bajo.

Texto imagen 4	EasyOCR Texto reconocido	EasyOCR accuracy (%)	Tesseract OCR Texto reconocido	Tesseract OCR accuracy (%)
STARBUCKS COFFEE	STARBUCKS COFFEE	0.9889	<i>no detecta nada</i>	

Texto imagen 5	EasyOCR Texto reconocido	EasyOCR accuracy (%)	Tesseract OCR Texto reconocido	Tesseract OCR accuracy (%)
LLANFAIROWLLG WYNGYLLGOGER YCHWYRNDROB WLLLANTYSULI OGOGOGGOCH	LLANFAIROWLLGW YNGYLLGOGERYCH WYRNDROBWLLLL ANTYSULIOGOGOG GOCH	0.9718	<i>no detecta nada</i>	
Llan-vire-pooll-guin-gill-go-ger-u-queern-drob-ooll-llandus-ilio-gogo-goch	Llan-vire-pooll-guin-gil-l-go-ger-u-queern-drob-ooll-llandus-ilio-gogo-goch	0.8919	<i>no detecta nada</i>	
Way Out	Way Out	0.9956	<i>no detecta nada</i>	
Holyhead	Holyhead	0.7837	<i>no detecta nada</i>	

Texto imagen 6	EasyOCR Texto reconocido	EasyOCR accuracy (%)	Tesseract OCR Texto reconocido	Tesseract OCR accuracy (%)
TELEPHONE	TELEPHONE^	0.4519	<i>no detecta nada</i>	
TELEPHONE	ELEPHES	0.1577	<i>no detecta nada</i>	
ENOHPELET	#MOHQJJI;	0.4519	<i>no detecta nada</i>	

En la imagen 7, ambos métodos no detectan nada, y es correcto, ya que la imagen no tiene texto.

Tras el análisis de estos casos particulares, podemos hacer una valoración global de los porcentajes de precisión que presentan cada uno de los métodos.

Para los métodos de detección, el método EAST aporta unos resultados de acierto de entre el 80-90 % , en imágenes con texto estructurado, mientras que con imágenes con texto no estructurado dependiendo de la complicación de la imagen su precisión oscila entre 70 - 80%. Datos mucho mejores que los que se pueden obtener con MSER, el cual sobre todo para texto no estructurado presenta una capacidad de acierto casi nula, ya que como ya se ha mencionado genera muchos falsos positivos.

En cuanto al porcentaje obtenido aplicando el método CRNN, su valor oscila entre el 75 y 80%. Se ha comprobado que la efectividad del método se reduce considerablemente cuando el texto se encuentra totalmente girado o borroso, sin embargo, cabe destacar su eficiencia en texto estático y bien cortado.

Por último, para métodos de detección + reconocimiento, ya se ha visto que ante los mismos ejemplos de imágenes puntuales EasyOCR presenta bastante mejores resultados que Tesseract OCR, sobre todo para los casos de imágenes con texto no estructurado. El nivel de acierto de EasyOCR, oscila entre 80-90 %. Mientras que para el caso de Tesseract OCR cuando el texto es no estructurado, su porcentaje de acierto es prácticamente nulo.

5.MODELO RESNET

Por último, se ha decidido entrenar una pequeña red neuronal haciendo uso de la arquitectura ResNet para detección de caracteres escritos a mano. Cabe mencionar que esta tarea tiene una dificultad muy añadida ya que los caracteres escritos a mano son irregulares debido a que, no es común escribir manualmente las letras con la misma tipografía.

Como base de datos se han tenido que combinar 2 datasets. El primero consiste en el dataset MNIST 0-9 [17]. Este dataset consta de 60.000 imágenes de entrenamiento y 10.000 de testeo. Las imágenes poseen un tamaño de 28x28 píxeles las cuales se encuentran codificadas en un archivo de texto plano. Este dataset se puede utilizar directamente desde Tensorflow, sin embargo, para este entrenamiento, se han descargado los archivos comprimidos y leídos directamente desde el código, para así poder entrenarse sin necesidad de conexión a internet. La información de las imágenes se encuentran concatenadas en un vector almacenado en el fichero.

Por otro lado, se ha utilizado la base de datos Kaggle A-Z dataset [18]. Es una base de datos que cuenta con imágenes de todos los caracteres de la A a la Z (al ser en inglés no se presenta el carácter “Ñ”), escritas a mano. La base de datos se centra en caracteres en mayúsculas. Las imágenes se presentan en un archivo .csv cuyas filas y columnas representan la imagen de tamaño 28x28 (al igual que MNIST). De modo similar, se debe desarrollar una función que sea capaz de leer y formar las imágenes a raíz del archivo con formato csv.

Una vez se tienen ambos datasets leídos se deben unir en uno solo para aportarlas a la red para su posterior entrenamiento

En cuanto al modelo utilizado, primeramente se probó con un modelo clásico secuencial, haciendo uso de tensorflow y keras. Este modelo se entrenó únicamente para la detección de los dígitos y posteriormente junto con los caracteres. El modelo entrenado para dígitos tenía un porcentaje de acierto considerablemente óptimo (dependiente de la forma de escribir el número), sin embargo, al añadir el segundo dataset, poseía mayor error. De esta parte no se tomaron datos de la evolución del entrenamiento de la red. Al ver este resultado, se decidió emplear un modelo más complejo con la esperanza de mejorar el porcentaje de acierto de la red. Por este motivo, se decidió utilizar el modelo ResNet

Este modelo se conoce como red neuronal residual. Este modelo de red neuronal artificial se caracteriza por ser capaz de realizar saltos de dos o incluso tres capas ya que éstas se encuentran normalizadas por lotes y no linealizadas. Además, la red crea una matriz de pesos adicionales para estos saltos. La necesidad de dichos saltos para un modelo de una red neuronal reside en la pérdida de gradiente a la hora de realizar la retropropagación.

Para entrenar una red, se hace uso del algoritmo de retropropagación. En una primera instancia, los pesos de las neuronas se establecen aleatoriamente, se introduce una entrada y se coteja la salida esperada con la dada por la red. Muy probablemente, el resultado no coincidirá, es en este momento que se hace uso de la retropropagación. Este algoritmo recorre el camino inverso de la red para encontrar qué peso ha aportado mayor error y cambiarlo. Para hacerlo busca los mínimos locales, realizando el descenso del gradiente. Se basa en los pesos y el error y, habiendo calculado la derivada de la función (gradiente) se encuentra el mínimo local.

En ciertas ocasiones, este gradiente se va reduciendo en gran cantidad, resultando la finalización del entrenamiento de la red, ya que con gradientes cercanos al 0, no encuentra pesos para cambiar y no se modifica nada. Para solucionar este problema se crearon las redes neuronales residuales como ResNet.

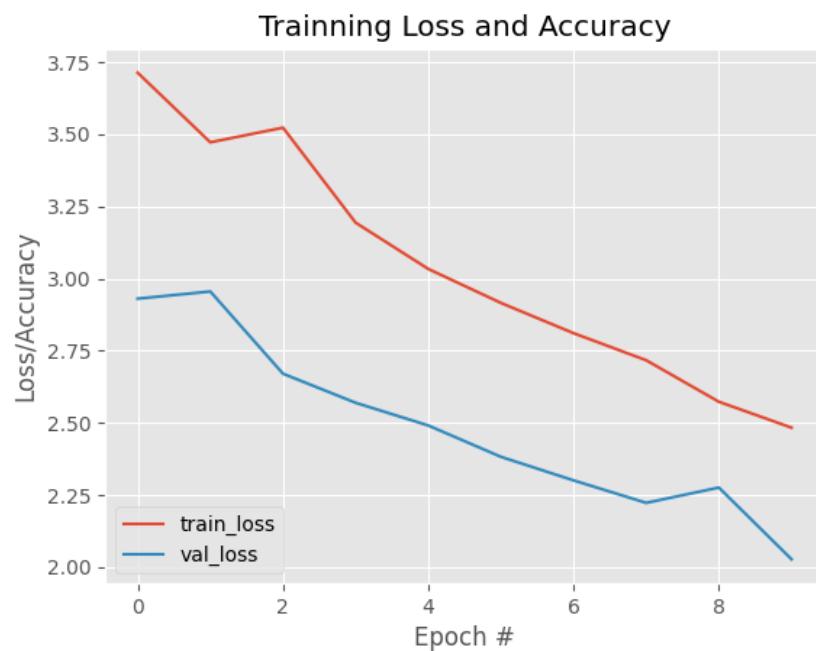
ResNet evita este problema ya que, con la posibilidad de realizar saltos entre capas, además de acelerar el proceso, se altera la forma de calcular el gradiente, lo que evita que se reduzca tanto. Además, la red aprende a diferenciar las capas que aportan y las que no, lo que aumenta la velocidad de procesamiento y entrenamiento.

Para realizar el entrenamiento se ha hecho uso del toolkit CUDA para ubuntu 21.18, con el objetivo de usar la GPU. En un primer lugar se trató de utilizar dicha herramienta en windows 11, sin embargo, fue imposible debido a diferentes problemas con las versiones soportadas por Tensorflow, CUDA y el soporte para windows 11. Se trató de realizar el entrenamiento sin uso de GPU, pero su tiempo fue demasiado alto, por este motivo se realizó finalmente en ubuntu.

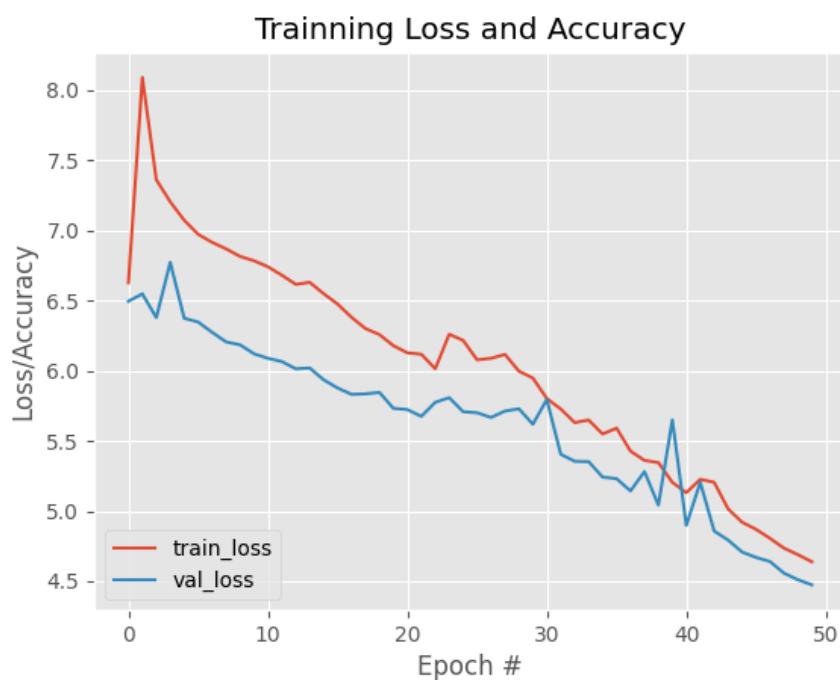
Debido a que el trabajo se centra principalmente en la explicación y comparación de métodos ya existentes, el tiempo para el entrenamiento de la red fue limitado. Por este motivo, el número de épocas es bajo. Primeramente se entrenó un modelo con 10 épocas y posteriormente uno con 50. Cada época, haciendo uso de una gráfica NVIDIA GeForce 940MX, tardaba en realizarse alrededor de 5 minutos y medio, adicionales a unos 20 segundos entre época y época.

Una vez entrenados ambos modelos, se imprime una gráfica representativa de la evolución del error calculado en el entrenamiento y en la validación. La gráfica del modelo con 10 épocas se representa en la gráfica 1 y en la gráfica 2 se muestra el modelo con 50 épocas.

Cabe destacar que, como la asignación de pesos iniciales es aleatoria, dependiendo de estos valores iniciales se requerirá mayor o menor tiempo para su entrenamiento. Esto se refleja en que el valor inicial de cada gráfica es muy distinto, incluso aumenta en las primeras épocas del segundo modelo.



Gráfica 1. Modelo con 10 épocas



Gráfica 2. Modelo con 50 épocas

Por desgracia, el modelo entrenado no presenta resultados satisfactorios. Esto refleja la complejidad de ésta tarea. Como mejora, se podría alterar el número de capas utilizado y aumentar el número de épocas en busca de una precisión mayor.

Para realizar la predicción de la red, se hace uso de funciones en las librerías cv2 e imutils, para captar los contornos presentes en la imagen, y extraer los caracteres por separado, creando una bounding box para cada carácter. Estas zonas representadas por las bounding boxes, se redimensionan y normalizan para entregar una lista de imágenes 28x28 las cuales son analizadas por la red individualmente. Para cada imagen devuelve el carácter leído y la certeza de que ha acertado la predicción.

Para probar las predicciones se ha hecho uso de una imagen de un folio en la que se ha escrito “HELLO WORLD” a mano. Al ser texto escrito manualmente, el resto de imágenes utilizadas anteriormente, no son válidas.

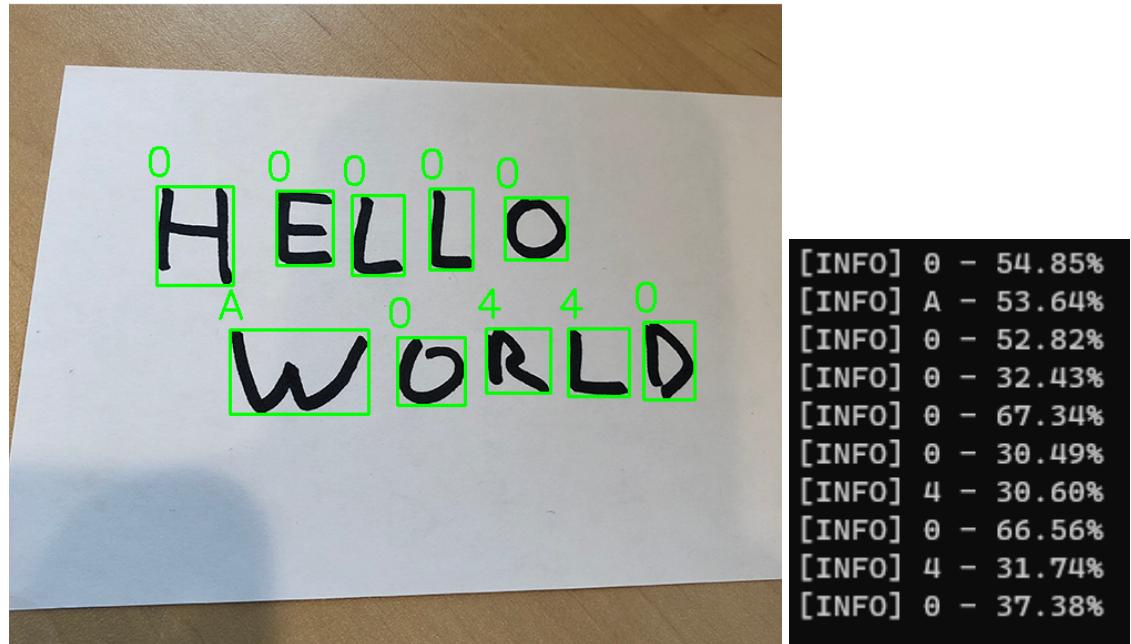


Imagen 8. Salida de imagen y terminal

Como se puede observar y como se ha explicado previamente, el resultado de la predicción de la red no es correcto. Además, cabe destacar que los porcentajes de certeza en las predicciones son demasiado bajos, ni llegando al 70%, lo cual, aunque las letras fueran correctas no sería un valor adecuado para “confiar” en la predicción.

6.CONCLUSIONES

Por tanto, se puede concluir que la extracción de texto de imágenes es una tarea ardua, sobre todo cuando el texto no es estructurado. Sin embargo, existen una serie de métodos que cada vez están haciendo más posible llevar a cabo esta labor, y es sobre todo gracias a la incorporación del deep learning a estas técnicas, lo que ha permitido obtener grandes avances en este sector, esto se ha podido comprobar al comparar el método MSER que no usa deep learning con el EAST que si lo usa. Los resultados, en el caso del EAST son mucho más óptimos, tanto para texto estructurado como no estructurado, como ya se ha explicado en la memoria.

Además, se ha podido comprobar que incluso entre métodos de detección + reconocimiento que incorporan deep learning también hay diferencia entre ellos. En nuestro caso se analizaron y compararon los resultados obtenidos por EasyOCR y Tesseract OCR, siendo los resultados del primero bastante mejores que los del segundo. Y destaca sobre todo la eficacia de EasyOCR para la extracción de texto en escenas, ya que pese a lo tediosa que es la tarea sus resultados se acercan a la perfección, aunque en determinadas situaciones donde el texto no es del todo claro, como en el caso de las letras reflejadas en el cristal, el resultado no es correcto.

Con lo cual, podemos decir que poco a poco se están obteniendo grandes mejoras ante este problema, aunque aún presentan margen de mejora las técnicas existentes.

Además, también se ha realizado el entrenamiento de una pequeña red neuronal, donde hemos podido comprobar y corroborar lo difícil que es esta tarea, sobre todo a la hora reconocer texto manuscrito.

6.BIBLIOGRAFÍA

[1]. EAST: An Efficient and Accurate Scene Text Detector

<https://arxiv.org/pdf/1704.03155v2.pdf>

[2]. ¿Qué es un NMS- Network Management System?

<https://www.cic.es/que-es-un-nms-network-management-system/>

[3]. FCN de red totalmente convolucional para segmentación de imágenes

<https://programmerclick.com/article/23651927681/>

[4]. Chen, H., Tsai, S., Schroth, G., Chen, D., Grzeszczuk, R. and Girod, B. (2011). Robust text detection in natural images with edge-enhanced maximally stable extremal regions. In: IEEE International Conference on Image Processing, pp. 2609–2612.

[5]. Non Maximum Suppression: Theory and Implementation in PyTorch

<https://learnopencv.com/non-maximum-suppression-theory-and-implementation-in-pytorch/>

[6]. Deep Learning Based OCR for Text in the Wild

<https://nanonets.com/blog/deep-learning-ocr/#east-efficient-accurate-scene-text-detector->

[7]. Text Recognition with CRNN-CTC Network

<https://wandb.ai/authors/text-recognition-crnn-ctc/reports/Text-Recognition-With-CRNN-CTC-Network--VmlldzoxNTI5NDI>

[8]. An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition

<https://arxiv.org/abs/1507.05717>

[9]. Scene Text Detection and recognition using EAST and Tesseract

<https://towardsdatascience.com/scene-text-detection-and-recognition-using-east-and-tesseract-6f07c249f5de>

[10]. Hands-on tutorial on EasyOCR for scene text detection in images

<https://analyticsindiamag.com/hands-on-tutorial-on-easyocr-for-scene-text-detection-in-images/>

[11]. Detección OCR: CRAFT

<https://www.codetd.com/es/article/12277690>

[12]. An overview of Tesseract OCR engine

<https://static.googleusercontent.com/media/research.google.com/es//pubs/archive/33418.pdf>

[13]. Tesseract OCR

https://es.wikipedia.org/wiki/Tesseract_OCR

[14]. Apuntes de teoría de la asignatura de Visión por computador, tema 6 características de imagen

[15]. Maximally stable extremal regions

https://en.wikipedia.org/wiki/Maximally_stable_extremal_regions

[16]. A gentle introduction to OCR

<https://towardsdatascience.com/a-gentle-introduction-to-ocr-ee1469a201aa>

[17]. MNIST 0-9 digit dataset

<http://yann.lecun.com/exdb/mnist/>

[18]. Kaggle dataset A-Z

<https://www.kaggle.com/sachinpatel21/az-handwritten-alphabets-in-csv-format/metadata>

[19] Redes neuronales residuales – Lo que necesitas saber (ResNet)

<https://datascience.eu/es/aprendizaje-automatico/una-vision-general-de-resnet-y-sus-variantes/>

[20] Comprensión de los fundamentos del descenso gradual

<https://datascience.eu/es/aprendizaje-automatico/descenso-gradual/>

[21] Repositorio donde se encuentran los archivos

https://github.com/Inigo183/scene_text_recognition