ULPGC Universidad de Las Palmas de Gran Canaria | ESCUELA DE INGENIERÍA INFORMÁTICA

Trabajo Fin de Grado

Reconocimiento e identificación de embarcaciones en entornos marítimo-portuarios

TITULACIÓN: Grado en Ingeniería Informática

AUTOR: Tinizara María Rodríguez Delgado

TUTORIZADO POR:
Nelson Monzón López, Jonay Suárez Ramírez y Leopoldo López Reverón

Fecha 2024

Agradecimientos

Me gustaría agradecer en primer lugar a mi familia, los cuales me han sujetado cuando más lo necesitaba, han sido mi pañuelo para las lágrimas, mis carcajadas en las tristezas y mi luz en la oscuridad: *Mamá, hermano, lo hemos conseguido*. Papá, no puedo olvidarme de ti, te fuiste sin poder ver cumplido nuestro sueño, pero hoy miro al cielo y te veo sonreír, *lo logramos*, tal y como me prometiste.

Abuelo, aunque acompañas a papá en el cielo, siempre seguirás siendo el capitán de mi barco y el timón de mi rumbo. Me enseñaste, junto a abuela, todo lo que sabían y más sobre el mar. Esto también es por ustedes, por el *Cathami*, y por el *Melonina*.

A mi compañera de piso y mi hermana de distinta sangre, Esperanza, por ser mi calma en momentos de angustia, por estar en las noches de dudas y miedos, y por sobre todo, nunca dejarme sola.

A la primera persona que conocí en esta carrera y a día de hoy, si miro al lado derecho de la mesa sigue ahí, Fernando, mi compañero de batalla.

Finalmente, quiero agradecer a mis tutores Nelson Monzón López, Jonay Suárez Ramírez y Leopoldo López Reverón por toda la ayuda y apoyo que me han brindado durante el desarrollo de este trabajo, al igual que a la empresa *Qualitas Artificial Intelligence And Science* S.A, y a todos y cada uno de los compañeros, los cuales siempre han estado dispuestos a tenderme una mano para resolver cualquier problema.

Resumen

La detección e identificación de embarcaciones desempeñan un papel crucial para garantizar la seguridad en entornos de playa, costa y litoral, y especialmente en zonas portuarias. Esta tarea es fundamental para proteger tanto los intereses económicos como la integridad de las operaciones en estos espacios estratégicos.

En el ámbito marítimo, particularmente en puertos con alta actividad turística y pesquera, la capacidad de identificar de manera precisa embarcaciones a través de información clave, como matrículas, nombres o nacionalidades, es de vital importancia. Sin embargo, las condiciones desfavorables, como la distancia entre las cámaras de videovigilancia y las embarcaciones, representan un desafío técnico significativo.

El desarrollo de tecnologías que permitan superar estas limitaciones no solo incrementará la seguridad en los entornos marítimos, sino que también optimizará procesos actualmente manuales, aportando mayor eficiencia al sector. Automatizar la recopilación de datos relevantes permitirá agilizar las operaciones y reforzar la capacidad de respuesta ante posibles riesgos o incidentes, consolidando así un entorno más seguro y controlado.

En este Trabajo Fin de Título (TFT), queremos desarrollar un prototipo funcional de un sistema capaz de identificar y reconocer caracteres automáticamente en entornos portuarios, utilizando cámaras PTZ (Pan-Tilt-Zoom) instaladas en dichos espacios. El objetivo principal es abordar los desafíos asociados a la identificación precisa en condiciones reales, como la distancia y las condiciones lumínicas variables.

Para ello, se llevó a cabo un análisis exhaustivo de bibliotecas y modelos especializados en aprendizaje profundo, así como en técnicas avanzadas de detección y reconocimiento óptico de caracteres (OCR). Esta fase incluyó una evaluación comparativa de diversos modelos, mediante pruebas específicas que permitieron analizar su rendimiento en términos de precisión, velocidad de procesamiento y consumo de recursos computacionales. Dichos resultados facilitaron la selección de los modelos más adecuados para el sistema. A continuación, se procedió al entrenamiento y desarrollo del modelo seleccionado, utilizando un conjunto de datos representativo del entorno portuario. Finalmente, se elaboraron las conclusiones basadas en los resultados obtenidos, que no solo validan el desempeño del sistema, sino que también sientan las bases para futuras mejoras. Estas mejoras incluirán la integración de nuevos conocimientos adquiridos durante el desarrollo, con el fin de optimizar el prototipo y garantizar su escalabilidad y robustez en aplicaciones reales.

Abstract

The detection and identification of vessels play a crucial role in ensuring safety in beach, coastal, and shoreline environments, particularly in port areas. This task is fundamental for protecting both the economic interests and operational integrity of these strategic spaces.

In the maritime domain, especially in ports with high levels of tourism and fishing activity, the ability to accurately identify vessels through key information such as registration numbers, names, or nationalities is of paramount importance. However, unfavorable conditions, such as the distance between surveillance cameras and the vessels, present significant technical challenges.

The development of technologies capable of overcoming these limitations will not only enhance safety in maritime environments but also optimize processes currently performed manually. Automating the collection of relevant data will streamline operations and strengthen the capacity to respond to potential risks or incidents, thereby fostering a more secure and controlled environment.

In this Bachelor's Thesis (TFT), our purpose is to develop a functional prototype system capable of automatically identifying and recognizing characters in port environments, utilizing PTZ (Pan-Tilt-Zoom) cameras installed in these areas. The main objective is to address the challenges associated with accurate identification under real-world conditions, such as distance and variable lighting.

To achieve this, an exhaustive analysis was conducted on libraries and models specialized in deep learning and advanced optical character recognition (OCR) techniques. This phase included a comparative evaluation of various models through specific tests designed to assess their performance in terms of accuracy, processing speed, and computational resource consumption. These results facilitated the selection of the most suitable models for the system.

Next, the selected model was trained using a dataset representative of port environments. Finally, conclusions were drawn based on the obtained results, which not only validate the system's performance but also lay the groundwork for future enhancements. These improvements will include the integration of new knowledge acquired during development to optimize the prototype and ensure its scalability and robustness in real-world applications

Índice General

Índice de figuras

Capítulo 1

Introducción

1.1. Motivación inicial y justificación

La seguridad marítima se refiere a la protección de la tripulación y/o pasajeros de las distintas embarcaciones que transitan en entornos marítimo portuarios o de playa, costa o litoral, así como de aquellas personas que residan, trabajen o simplemente puedan verse afectadas por cualquier peligro que surja cerca de estos entornos. Además, abarca la protección del medio ambiente, la actividad pesquera, el comercio y el transporte marítimo, y la defensa contra actividades ilegales como la piratería y el tráfico de personas.

La seguridad marítima adquiere una importancia crucial en entornos de playa, costa y litoral debido a la vastedad del mar en comparación con el suelo firme. En el mar, las infracciones normalmente son más complicadas de detectar ya que no existe una infraestructura completamente eficaz que sea capaz de identificar automáticamente y sin intervención humana qué embarcación está involucrada. Aunque existen tecnologías como el Sistema de Identificación Automática (AIS), que proporciona información valiosa para posicionar e identificar embarcaciones, este sistema tiene limitaciones importantes. Por un lado, el AIS depende de que las embarcaciones mantengan el dispositivo activado, lo que deja espacio para actividades ilícitas o infracciones por parte de aquellas que deliberadamente lo apagan. Por otro lado, el alcance del AIS puede ser limitado en zonas costeras con alta densidad de embarcaciones pequeñas, como las usadas para actividades recreativas, ya que muchas de ellas no están obligadas a llevar este sistema. Estas restricciones subrayan la necesidad de soluciones complementarias basadas en tecnologías avanzadas como la inteligencia artificial y la visión artificial para mejorar la vigilancia marítima en estos entornos.

Por este motivo, y en colaboración con la empresa *Qualitas Artificial Intelligence and Science*, estamos desarrollando un proyecto que busca suplir estas deficiencias en entornos marítimo-portuarios a distancia que puedan ser registradas por cámaras, mediante la aplicación de técnicas de inteligencia artificial. Utilizando la detección de objetos, el reconocimiento de caracteres y el procesamiento de imágenes, estamos creando un sistema automático capaz

de recibir imágenes de una o varias embarcaciones, discernir entre textos de interés como el nombre, la nacionalidad o algún tipo de identificación, y diferenciar estos datos de otros visibles como el modelo del motor o de la propia embarcación. Este sistema mejorará significativamente la capacidad de identificar embarcaciones implicadas en actividades sospechosas o ilegales, aumentando así la seguridad en el entorno portuario.

1.2. Objetivos

La realización de este trabajo posee una serie de pequeños objetivos, los cuales empiezan con un análisis de las librerías y estrategias relacionadas con el aprendizaje profundo y el campo de la detección y el reconocimiento de caracteres, además de entender el procedimiento que se lleva a cabo de manera general. A su vez, se continúa contrastando entre sí aquellos modelos que, de manera teórica, aportan unos resultados en términos de precisión y rendimiento mayores que la media. Tras esta comparación, se procederá a realizar la implementación de una primera versión del sistema en la que, haciendo uso de las estrategias seleccionadas anteriormente, se pongan a prueba, en un entorno controlado y con un *dataset* especifico de la temática del proyecto, comparando resultados en términos de robustez, calidad y uso de recursos, consiguiendo de esta manera una subselección de los modelos que sobresalgan, y dando pie a la posibilidad de unificar las fortalezas de cada uno de ellos, generando de esta manera un sistema que *a priori* obtendrá resultados superiores a si se ejecutaran de manera independientes.

Finalmente, y para cumplir con el objeto de estudio de este TFT, se procede a efectuar el código que nos permita llevar a cabo la lectura de las matrículas de las embarcaciones en un entorno marítimo-portuario.

- Familiarización con estrategias de Inteligencia Artificial y de detección y reconocimiento de caracteres
- Familiarización con el uso de librerías para el procesamiento de imágenes como *OpenCV*
- Familiarización con el entorno de captura y almacenamiento de imágenes desarrollado por *QAISC* con relación a las embarcaciones
- Detección consistente de las matriculas de las embarcaciones
- Reconocimiento de los caracteres de las matriculas de manera precisa

1.3. Competencias desarrolladas.

En este Trabajo de Fin de Título (TFT) se ha procedido a la investigación, comparación y mejora de un modelo encargado de aplicar la detección y reconocimiento de caracteres al mundo marítimo, aplicando técnicas de aprendizaje profundo y automático. Teniendo en

cuenta este enfoque del proyecto, la competencia especifica planteada en la Memoria del Plan de Estudios del Grado de Ingeniería Informática que responde a esto, esta definida en la ***CI15***: "*Conocimiento y aplicación de los principios fundamentales y técnicas básicas de los sistemas inteligentes y su aplicación práctica*"

Adicionalmente a lo comentado anteriormente, en este desarrollo se ha necesitado la aplicación de ciertos procedimientos algorítmico complejos que nos permitían diseñar una solución a los desafíos que nos enfrentábamos de manera eficiente, lo que se vincularía directamente con la competencia ***CI6***: "*Conocimiento y aplicación de los procedimientos algorítmicos básicos de las tecnologías informáticas para diseñar soluciones a problemas, analizando la idoneidad y complejidad de los algoritmos propuestos.*"

Por último, también es importante tener en consideración el ciclo de vida del *software* tanto el desarrollo de este, como en todos los proyectos de desarrollo. Desde la revisión inicial de los objetivos de lectura, pasando por la lectura de la documentación que ofrecía la base desde la que parte los modelos, hasta la implementación y posterior mejora, se aplica la metodología propia del desarrollo de *software*, cumpliendo de esta manera con la competencia ***CI16***: "*Conocimiento y aplicación de los principios, metodologías y ciclos de vida de la ingeniería del software.*", asegurando un proceso estructurado y sostenible.

1.4. Alineamiento con los objetivos de desarrollo sostenible

Dado que este proyecto se enfoca en la detección y reconocimiento de embarcaciones, ademas de sus respectivos medios de identificación en el entorno marítimo, el objetivo principal del desarrollo ha consistido en mejorar la seguridad tanto de los puertos, como de aquellas áreas donde existan fauna marina vulnerable.

En la tabla siguiente se muestra las relaciones que existe entre los objetivos de desarrollo sostenible (ODS) y los objetivos que ha perseguido el desarrollo de este Trabajo Fin de Título.

- **ODS 8: Trabajo decente y crecimiento económico**. Al automatizar la identificación de las embarcaciones, incrementamos la eficiencia del puerto, y liberamos de carga a los trabajadores, optimizando de esta manera el uso de los recursos, y facilitando un acceso controlado y seguro del medio, fomentando la inversión en este sector.
- **ODS 9: Industria, innovación e infraestructura**. La incorporación de esta tecnología presenta una innovación significativa en el sector al no existir una solución similar. Esto, de manera consecuente, impulsa el desarrollo de nuevas infraestructuras inteligentes mejorando la competitividad de los puertos con el resto de sectores.
- **ODS 14: Vida submarina**. Al reforzar el control de las zonas restringidas, este

ODS	Grado de relación			
	0 No procede	1 Bajo	2 Medio	3 Alto
1 Fin de la Pobreza	X			
2 Hambre cero	X			
3 Salud y Bienestar	X			
4 Educación de calidad	X			
5 Igualdad de género	X			
6 Agua limpia y saneamiento	X			
7 Energía Asequible y no contaminante	X			
8 Trabajo decente y crecimiento económico		X		
9 Industria, Innovación e Infraestructuras				X
10 Reducción de las desigualdades	X			
11 Ciudades y comunidades sostenibles	X			
12 Producción y consumo sostenibles	X			
13 Acción por el clima	X			
14 Vida submarina				X
15 Vida de ecosistemas terrestres	X			
16 Paz, justicia e instituciones sólidas	X			
17 Alianzas para lograr objetivos	X			

Cuadro 1.1: Grado de relación del TFT con los objetivos de desarrollo sostenible.

sistema contribuye a la protección de las especies y ecosistemas vulnerables.

1.5. Presupuesto

Un punto importante de todo proyecto de investigación, consiste en ofrecer una visión clara sobre el coste de los recursos tanto *hardware* como *software* y humano necesarios para poder llevarlo a cabo de manera adecuada. A continuación, se procederá a realizar un desglose detallado donde en primer lugar, se detallará los costes asociados al personal que incluirá el valor del tiempo invertido, tanto del alumno resto del personal involucrado, como son tutor y co-tutores, cual experiencia y conocimientos son necesarios para una correcta ejecución.

En segundo lugar, se tratará sobre los recursos *hardware* y *software*, que incluirán el uso de las diferentes tarjetas gráficas del servidor, propiedad de la empresa colaboradora, necesarias para poder ejecutar los modelos en una infraestructura correcta, además de otros recursos relevantes para el desarrollo del proyecto.

1.5.1. Coste humano

Cuando tratamos sobre el coste del personal en este proyecto, principalmente se centra en el tiempo que el alumno ha dedicado a la investigación del tema, así como el tiempo invertido en el desarrollo, verificación, y ejecución de las diferentes pruebas que permiten obtener conclusiones que, posteriormente pueden establecer una linea de trabajo a futuro. Este proceso ha sido costeado por la *Fundación Universitaria de las Palmas de Gran canaria* (FULP) a través de una beca para la realización de prácticas extracurriculares.

Además del tiempo invertido por parte del alumno, también es importante tener en cuenta el coste asociado al trabajo del tutor y co-tutores, que poseían funciones de acompañamiento y supervisión continuo, durante el desarrollo del trabajo asegurando de esta manera, la calidad del trabajo.

Personal	**Horas**	**Coste/hora**	**Coste total**
Alumno	300	8,21€	2465€
Tutor	50	50€	2500€
Co-tutor 1	60	30€	1800€
Co-tutor 2	60	30€	1800€
Total	470	118,21€	**8500€**

Cuadro 1.2: Tabla de costes humanos

1.5.2. Coste tecnológico

En cuanto al coste de recursos *hardware* y *software*, es fundamental considerar todas aquellas librerías, herramientas y dispositivos necesarias y que han sido utilizado para el

desarrollo de código, redacción de la documentación pertinente, y obtención de las imágenes necesarias para la creación del conjunto de datos, que se empleara como *dataset* para las pruebas y comparaciones.

En relación a las librerías utilizadas, todas ellas eran de uso gratuito, y en algunos casos, de código abierto, por lo que no ha incurrido en costes adicionales. Sin embargo, en lo que respecta a los dispositivos físicos, se tendrá en cuenta el ordenador del estudiante, así como las cuatro tarjetas gráficas *NVIDIA GeForce RTX 3060 Ti* de 12GB de *VRAM* que forman parte del servidor de la empresa *QAISC*. Estas fueron brindadas al estudiante para que pueda realizar las tareas de desarrollo, así como de ejecución de los diferentes modelos sin que esto suponga agun problema computacional por falta de potencia en el portátil personal.

Personal	**Coste total**
Ordenador portátil	1250€
Tarjeta gráficas	650€/gráfica
Total	**3850€**

Cuadro 1.3: Tabla de costes tecnológicos

Capítulo 2

Estado del arte.

2.1. Identificación de buques.

De manera general, todas las embarcaciones deben poder ser identificadas por las autoridades marítimas con el objetivo de poder velar por la seguridad no solo de los trabajadores y pasajeros de los diferentes barcos, sino también con la finalidad de conocer que mercancías se transportan y que ruta se sigue, evitando de esta manera posibles ataques por parte de piratas. En definitiva, una buena identificación del medio de transporte marítimo, aporta seguridad no solo en alta mar, sino en los puertos de amarre y de destino.

Para ello, a nivel internacional existe la **Organización Marítima Internacional** [54], la cual, tal y como se definen ellos mismos: " *Organización Marítima Internacional es un organismo especializado de las Naciones Unidas responsable de las medidas para mejorar la seguridad y la protección del transporte marítimo internacional y prevenir la contaminación procedente de los buques.*" [55]

De igual manera, esta organización le brinda a aquellos buques de transporte de mercancías cuyo volumen interior sea igual o superior a 100 toneladas de manera obligatoria, un número formado por las letras IMO, seguido de 7 dígitos que se les asigna por parte de la **IHS Maritime** [26]. Igualmente, desde 2013, y de forma voluntaria, a los buques pesqueros que también cumplan con poseer más de 12 metros de eslora total o [1] máxima, se les conferirá este número aunque no cumplan el criterio anteriormente comentado. A partir de 2016, también se ha ofrecido la posibilidad de que barcos pequeños, barcos refrigeradores u otros tipos, puedan solicitarlo. Es importante destacar que este número permanece inalterable durante el tiempo de vida del buque, es decir, siempre será el mismo aunque se modifique algún aspecto material, se venda a otra empresa o persona, o posea otra función durante un largo tiempo, además de tener la obligatoriedad de estar visible junto al nombre del mismo. [67, 53]

De igual manera, una de las principales maneras de identificar una embarcación, sobre

(a) Vista general de una embarcación (b) Vista específica del IMO

Ilustración 2.1: Ejemplo de identificación de buques: IMO. Fuente: ShipSpotting

todo las de recreo y los barcos de pasajeros, es por su **nombre** que se debe encontrar de manera visible en la popa [2] del barco. Éste tiene la utilidad de en caso de accidente o necesidad de solicitar ayuda poder determinar, junto con el resto de identificadores, de que navío nos referimos.

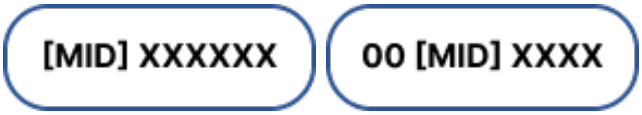
En cuanto al nombre, la legislación permite cualquier tipo de combinación que no supere las tres palabras, y su composición puede estar formada por anagramas y dígitos, siempre y cuando no preste a confusión bien con la matricula o algún otro texto que nos podríamos encontrar en el casco [3].

(a) Vista general de una embarcación (b) Vista específica del nombre.

Ilustración 2.2: Ejemplo de identificación de buques: nombre. Fuente propia.

Asimismo existe otro número, denominado **Identidad de Servicio Marítimo Móvil** (MMSI), que también es utilizado para la identificación de buques, pero, por contra, éste sí puede modificarse en el caso de que, por ejemplo, se cambie el abanderamiento [4] o el distrito de éste.

De manera general, el MMSI está compuesto por 3 dígitos, que representan el identificador del país denominado ***Maritime Identification Digits*** (MID). Seguido de este número, se encuentran 6 dígitos más que se conceden de manera aleatoria, y se le asigna al equipo de radio, en todas las bandas que éste emita. De este modo, existen diferentes formatos de MMSI dependiendo de si identificamos a una embarcación, o a una estación costera, En el caso de España, existen dos números MID: 224 para los barcos y 225, para las estaciones costeras. [44, 46, 13]



(a) Ejemplo MMSI de un barco.

(b) Ejemplo MMSI de una estación costera.

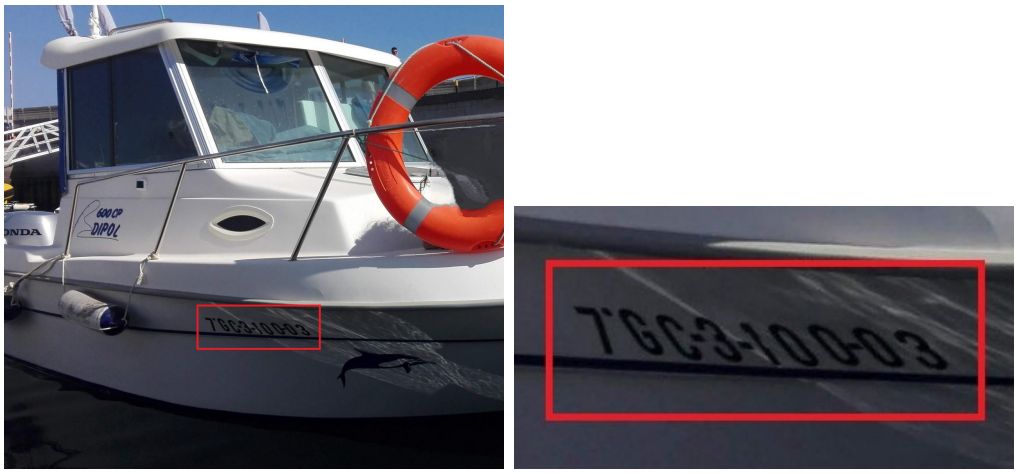
Ilustración 2.3: Ejemplo de identificación de buques: MMSI. Elaboración propia.

Como podemos observar en la en la figura 2.3b, al identificador MID de las estaciones costeras se les añade dos ceros, con el objetivo de poder diferenciar las llamadas o los avisos que puedan provenir de las embarcaciones.

La tercera opción mediante la cual se puede identificar a un buque es a través de su **matrícula**,la cual debe estar o pintada o fijada en las amuras [5] de manera que siempre se pueda visualizar independientemente del estado estético en el que se encuentre la embarcación.

En España, ésta posee un patrón único cuyo formato es el siguiente:

- Un dígito en formato ordinal que representa la lista a la que pertenece. En España existen ocho listas fijas que permiten la clasificación de las embarcaciones dependiendo del tipo de actividad que realice, ademas de una novena y última, que se utiliza de manera provisional para aquellos barcos que se encuentren en construcción, siempre y cuando no sean fabricaciones en serie.
- Provincia marítima.
- Distrito marítimo [6] del puerto al que pertenece.
- Folio en el que se encuentra registrado.
- Dos últimas cifras del año en el que se haya registrado.



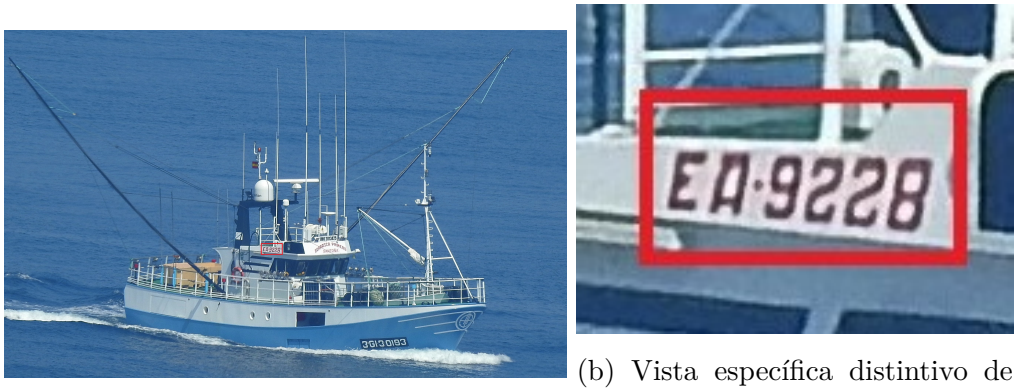
(a) Vista general de una embarcación (b) Vista específica de una matricula

Ilustración 2.4: Ejemplo de identificación de buques: matrícula. Elaboración propia.

Por último, aunque no es muy común encontrarlo debido a que no posee obligatoriedad en aquellas que posean un sistema de comunicación digital como una radio satélite o radio baliza, existen embarcaciones que si cuentan con el denominado **distintivo de llamada** *(call sign)*.

La **Unión Internacional de Telecomunicaciones** [70], indica cuales son las letras que identificará al país de abanderamiento de la embarcación. En el caso de buques cuyo peso sea inferior a 100 toneladas, el formato se corresponde con las siglas del país, seguidos de cuatro dígitos que representen la estación del barco. En el caso, de que los buques superen el peso anteriormente comentados, su patrón estará compuesto por tres letras en lugar de dos.

Centrándonos en el caso de España, sus *call signs* comenzaran con dos letras que se encuentran contenidas entre el intervalo AM-AO y EA-EH.[45, 14]



(a) Vista general de una embarcación (b) Vista específica distintivo de llamada

Ilustración 2.5: Ejemplo de identificación de buques: distintivo de llamada. Fuente: ShipSpotting

2.2. Redes Neuronales. Tipos.

Las redes neuronales poseen este nombre debido a la semejanza que presentan en su estructura con las neuronas presentes en el cerebro humano. Del mismo modo que las personas aprendemos y reconocemos patrones para razonar, las redes pueden ser entrenadas para que actúen de manera similar.[40, 21]

A modo de historia, en 1943, McCulloch y Pitts [41] desarrollaron el primer modelo simplificado de neurona en la que a partir de una serie de datos de entrada, se obtenía una única salida que representaba un valor entre 0 y 1. No fue hasta 1950, cuando Rosenblatt desarrolló el concepto de *Perceptrón*, el cual definía una red neuronal formada por varias entradas y pesos, se les realizaba una serie de operaciones: en primer lugar, el producto de la entrada por su peso. Posteriormente, se realizaba el sumatorio, al cual se le aplicaba la función de activación que determinaba si ésta, estaba activada o no, produciendo así la salida. Aunque era un modelo bastante sencillo y muy limitado, sentó las bases para las futuras investigaciones y desarrollos de este campo.

No fue hasta la década de los 80 y en adelante, cuando se produjeron avances destacados, tales como el desarrollo de nuevas arquitecturas, entre las que destacan las redes neuronales convolucionales y recurrentes, en 1989 y 1997 respectivamente, así como el algoritmo de *backpropagation*.[6, 30]

En los apartados siguientes, se explicarán los subtipos de redes neuronales comúnmente utilizados por los modelos dedicados al *OCR*.

2.2.1. Redes Neuronales Convolucionales (CNNs)

Este tipo de redes esta formada por una capa de entrada, una de salida y una o varias capas ocultas, las cuales, en su conjunto, son utilizadas para el procesamiento de datos de entrada que posean una estructura cuadricular, como pueden ser las imágenes o vídeos. En las capas ocultas, se aplican filtros convolucionales denominados K que nos permiten obtener patrones y características relevantes de las regiones. Este tipo de redes tienen un papel importante en las tareas de clasificación o detección, debido a esta capacidad de aprender y reconocer patrones. [47, 61]

2.2.2. Redes Neuronales Recurrentes (RNNs)

Las *RNNs* se caracterizan por permitir la transmisión de los datos a través de la red gracias a la presencia de "nexos" entre las capas, que simulan la existencia de una memoria entre los nodos. Esta "memoria temporal" permite, no solo propagar hacia delante la información, sino también, volver atrás, y realizar conexiones con otros nodos. Dada esta facilidad de vinculación de resultado, las *RNNs* son ampliamente utilizadas para aquellas tareas de procesamiento del lenguaje natural, como puede ser la traducción o el reconocimiento de palabras o sonidos.

Asimismo, una de las principales debilidades que poseen este tipo de redes, es la falta de capacidad de mantener el conocimiento dentro de la red, ocasionado por el denominado *desvanecimiento del gradiente*. Éste ocurre cuando una red neuronal esta formada por un gran número de capas, de manera que se propaga de forma sistemática el error, ocasionado que el entrenamiento acabe siendo muy lento e inestable, especialmente en las primeras neuronas.

Es por ello, que existe una variante denominada *Long Short-Term Memory* (LSTM) [22], en la cual se agregan unas *celdas de memoria*, que están compuestas por puertas de entrada, salida y de olvido, por lo que dictaminan en cada momento que información debe almacenarse, transmitirse, o eliminarse, siendo especialmente útiles cuando la secuencia es larga y compleja. [20, 68]

2.2.3. Redes Neuronales Recurrentes Convolucionales (CRNN)

Esta arquitectura combina las dos arquitecturas explicadas anteriormente con el objetivo de ser capaces de procesar aquellos datos de entrada secuenciales, así como datos con estructura espacial.

Su estructura esta formada por capas convolucionales intercaladas con capas recurrentes las cuales, primero realizaran la extracción de las características de las regiones de los datos de entrada, para, posteriormente, procesarlas de manera secuencial obteniendo los patrones que puedan estar presentes. [19, 7]

2.2.4. Redes Neuronales Profundas (DNNs)

Este tipo de redes son una técnica de aprendizaje profundo automático que están formadas por múltiples capas conectadas entre sí mediante conexiones ponderadas, sentando las bases de lo que conocemos actualmente como Aprendizaje Profundo.

Las DNNs presenta similitudes con el funcionamiento del cerebro humano, debido al gran número de conexiones y capas que la componen. De igual manera, su proceso de aprendizaje se asemeja al de otras redes neuronales, calculando primero las salidas de cada neurona que compongan la capa. A continuación, se compara ese valor con el valor esperado haciendo uso de una función de perdida. En el caso de que no sea el valor esperado, se aplica una función similar al descenso de gradiente, para ajustar los pesos de entrada y continuar de manera iterativa el proceso hasta que obtengamos la salida correcta. [24, 43]

2.2.5. Red Neuronal de Grafos (GNN)

Las Redes Neuronales de Grafos (Graph Neural Networks, GNNs), a diferencia de las redes neuronales tradicionales, utilizan grafos como datos de entrada en lugar de matrices o vectores, lo que les permite aprender relaciones más complejas y estructuradas. De este modo, la red trabaja directamente sobre las aristas y los nodos que definen el grafo.

En cuanto a su funcionamiento, se propagará la información de cada nodo haciendo uso de una función de agregación que recopila los datos de los nodos vecinos. Posteriormente, se aplica una función de transformación que actualiza el estado de cada nodo en función de la información agregada. Este proceso se repetirá de manera iterativa durante todas las capas que formen la *GNN*, y se obtendrá, en la capa de salida la representación de la tarea solicitada, bien sea una clasificación, predicción, etc. [77, 11]

2.2.6. Red Transformer

Las redes *transformers* nacen en 2017 en un artículo desarrollado por *Google* denominado *Attention is all you need* en el que se buscaba tratar la problemática de las traducciones de idiomas. Este tipo de arquitectura, se caracteriza por su capacidad para aprender del contexto a través de las relaciones entre datos secuenciales.

A diferencia de las redes convolucionales o las recurrentes, esta red destaca por la posibilidad de ejecutar en paralelo, gracias a la introducción de las denominadas *capas de atención*, cuya función es brindar mayor atención a ciertos aspectos de las entradas acorde al peso que se le haya asignado.

2.3. Detección y reconocimiento óptico de caracteres.

En la industria del **Reconocimiento Óptico de Caracteres** (OCR), existen dos enfoques principales de ejecución, los cuales pueden ser abordados haciendo uso de diferentes algoritmos y modelos.

Mientras que por un lado, se encuentran los modelos de ejecución en *dos fases*, de igual manera existen los denominados *end-to-end*. Los primeros ejecutan las dos fases secuencialmente pudiendo hacer uso de dos algoritmos especializados y diferentes, aportando flexibilidad y capacidad de adaptación al contexto en el que se encuentre el texto. En cambio, en los segundos se combinan en un solo algoritmo los procesos de detección y reconocimiento. Este enfoque permite desarrollar modelos más compactos y rápidos en la obtención de resultados, aunque pueden requerir un mayor consumo de recursos.

El normal flujo de ejecución de los modelos de detección y reconocimiento,es el ilustrado en la imagen 2.6, donde se distinguen dos fases principales. En primer lugar, el proceso de detección,que, mediante segmentación, se divide las zonas en regiones mas pequeñas. El siguiente paso, consiste en el reconocimiento de las regiones anteriormente obtenidas aplicando modelos tradicionales, que realizan una comparación píxel a píxel mediante similitud de patrones conocidos (esto se explicará un poco más en profundidad en el apartado 2.5), o bien con modelos modernos aplicando aprendizaje profundo y automático.

Asimismo, existe la posibilidad de añadir a las etapas de detección y reconocimiento,

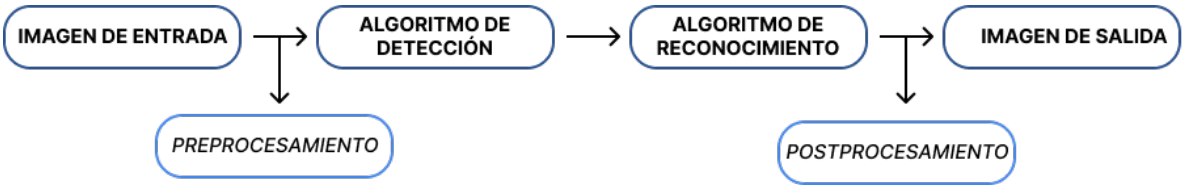


Ilustración 2.6: Flujo del proceso OCR en dos etapas. Elaboración propia.

fases intermedias que contribuyan a mejorar tanto la imagen de salida, como los resultados textuales. Estos procesos se podrán realizar tanto antes de ejecutar la detección, como después de haber obtenido la palabra o texto al completo. Si se añade tras capturar la imagen de entrada, generalmente se realiza una normalización y se aplicará una escala de grises con el objetivo de enfatizar los contornos, que derivarán, teóricamente, en un aumento de la precisión del módulo del reconocedor.

Con respecto a la introducción de un postprocesamiento tras la etapa de reconocimiento, el objetivo consiste en reconstruir el texto que hemos obtenido a nivel de carácter,es decir, aunque se introduzca una imagen en el que haya un párrafo, tras la detección, se traducirá de un mapa de bits a formato *ASCII* cada letra. A continuación, se llevará a cabo el proceso inverso: unir las letras obtenidas para formar las palabras, y finalmente, reconstruir la frase que conformará el texto al completo.

En cambio, existen las estrategias conocidas como *end-to-end*, las cuales haciendo uso de un único algoritmo, centralizan todos los pasos que se siguen. De igual modo que ocurre en las estrategias tradicionales de OCR, pueden añadirse etapas intermediarios que mejoren la calidad de la imagen. En la figura 2.7, podemos observar el esquema que representa el flujo general que se sigue al aplicar este planteamiento.

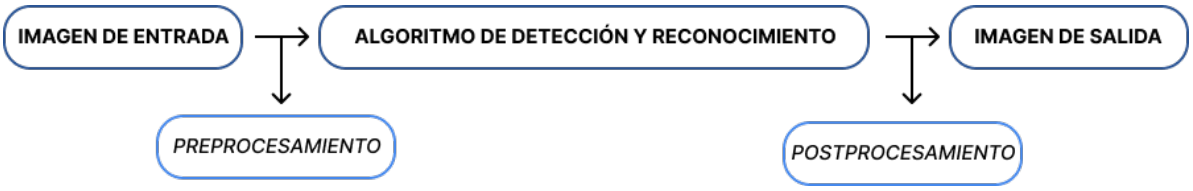


Ilustración 2.7: Flujo del proceso OCR aplicando la estrategia *end-to-end*. Fuente propia.

A la hora de elegir entre ambas estrategias, el usuario debe tener en cuenta las ventajas y desventajas que presentan, además de las diferencias técnicas que poseen. Si se opta por los modelos que dividen el proceso en múltiples etapas, se estaría optando por la modularidad, flexibilidad y especificación de cada acción que se realiza. El poder seleccionar un algoritmo para la detección y otro para el reconocimiento, podemos adaptarnos a muchas mas situaciones y formato de textos. Por su parte, los *end-to-end*, se inclinan por la simplicidad, fluidez y adaptabilidad a diferentes sistemas y dependencias, ya que eliminan la necesidad de realizar pruebas con cada uno de los algoritmos para comprobar cual posee mejores rendimiento en cada situación concreta.

En resumen, los modelos tradicionales suelen requerir más tiempo y recursos, lo que puede conllevar a su vez, la propagación un mayor número de errores. En contrapunto, los modelos de una sola etapa, poseen en su diseño mayor complejidad al tener que centralizar todo lo necesario para que se ejecute de forma correcta.

2.3.1. Modelos neuronales para la detección.

MixNet.

Red neuronal que ha sido entrenada para la detección de textos en situaciones complejas gracias a la utilización de diferentes capas, donde se fusionan las características contextuales extraídas en varios niveles de la red.

Se ha comprobado que los *backbones* basados en redes convolucionales pueden generar resultados erróneos en aquellos casos donde las entradas sea irregulares. Es por ello que *MixNet* opta por usar otra red no tan común denominada *FSNet*, formada por tres módulos que se entremezclan y se ejecutan repetidas veces, con el fin de obtener una imagen con un mayor numero de detalle. En cuanto a su arquitectura, nos encontramos con que, en el primer bloque se encuentran las capas convolucionales, que extraerán toda aquella información útil para el posterior reconocimiento del texto. El siguiente módulo es el encargado del recorte de las regiones independientemente de la posible calidad que éstas posean, para, finalmente, en el ultimo modulo nos encontraremos con las capas responsables de la combinación de los detalles extraídos anteriormente, tanto en los niveles superiores como las de nivel inferiores.

Finalmente se combinan los resultados de la ejecución de todos los módulos en un único mapa de calor con todas las regiones en las que se ha encontrado texto, tal y como podemos observar en la figura 2.8. [81, 65]

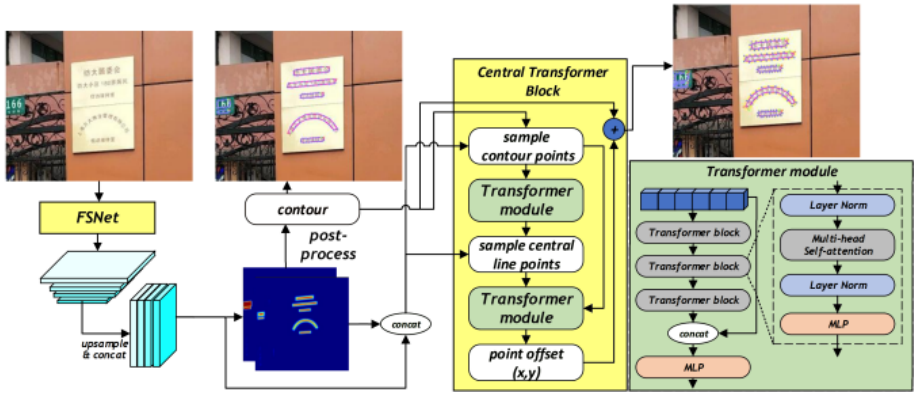


Ilustración 2.8: Esquema de MixNet. Fuente: *MixNet: Toward Accurate Detection of Challenging Scene Text in the Wild*

Differentiable Binarization (DBNet).

Este algoritmo hace uso de la denominada **binarización diferenciable**, donde el proceso de binarización se integra en el entrenamiento en lugar de ser un tratamiento separado, permitiendo así un mayor ajuste en sus umbrales, evitando utilizar los valores predeterminados.

Se entiendo como binarización al proceso mediante el cual una imagen solo puede poseer uno de dos valores posibles: 1, para el caso del blanco, ó, 0 para el caso del negro. En el contexto del OCR, se hace uso de este algoritmo en aquellos casos en los que los detalles de la imagen no son importantes, pero existe una gran diferencia entre el fondo y el texto, optando de esta manera por una imagen más simple. [34]

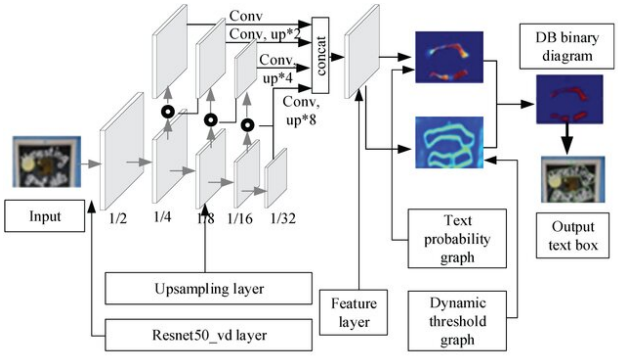


Ilustración 2.9: Esquema de DBNet. Fuente: *Data Recognition for Multi-Source Heterogeneous Experimental Detection in Cloud Edge Collaboratives*

Differentiable Binarization and Adaptive Scale Fusion (DBNet++).

DBNet++, es la mejora del algoritmo anteriormente comentado, que incorporan nuevas técnicas como la fusión de escala adaptativa para mejorar la escalabilidad de la binarización de las imágenes. En esta nueva versión, se combinan diferentes regiones de la imagen con el objetivo de conseguir una imagen resultante con un alto nivel de detalle.

Así, es la propia red la que durante el entrenamiento, ajusta de manera dinámica el nivel de binarización de cada una de las regiones segmentadas de la imagen de entrada, para su posterior combinación de manera eficiente. [35]

Deep Relational Reasoning Graph Network (DRRG).

Red basada en grafos formada por la unión de una red convolucional y una red convolucional gráfica, que se centra en resolver la problemática de la detección de textos en situaciones y formas aleatorias. Es por ello que su flujo de ejecución se basa en, primer lugar, realizar recortes de tamaños pequeños, para que posteriormente, a través de las diferentes capas convolucionales, se obtengan aquellos atributos que definan la forma de la región. Finalmente, se procederá a realizar conexiones como las que encontramos en los grafos, con el objetivo de determinar que región existe correlativa a otra, en función de los atributos obtenidos anteriormente. [83]

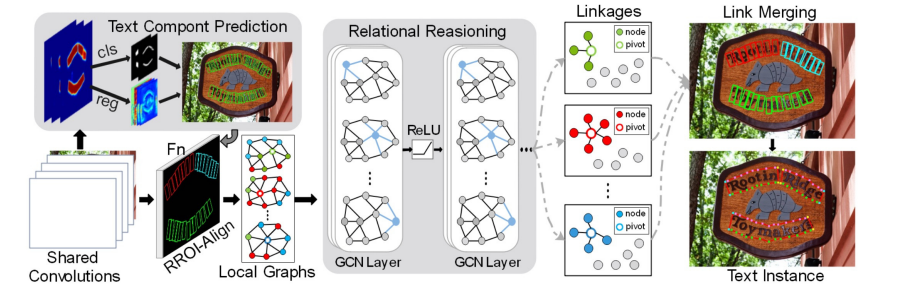


Figure 2. Overview of our overall architecture. Our network mainly consists of five components, *i.e.*, shared convolutions, text component prediction, local graphs, relational reasoning, and Link Merging.

Ilustración 2.10: Esquema de DRRG. Fuente: *Deep Relational Reasoning Graph Network for Arbitrary Shape Text Detection*

Efficient and Accurate Scene Text Detector (EAST).

Este algoritmo, también hace uso de redes convoluciones (CNN) para detectar posibles palabras en la imagen, siendo su principal objetivo la segmentación de las imágenes en regiones en una sola etapa. Es decir, tras introducir la imagen en la red, la CNN extrae los atributos como los contornos, bordes, etc, de todo aquello que resalte en la imagen. A continuación, se predicen los posibles puntos que delimitarán las regiones que encierren todo aquello que la red considere texto, junto con sus respectivos valores de confianza de las palabras reconocidas.

Como desventaja de este enfoque, se encuentra la pérdida de precisión del modelo al realizar la detección en una sola etapa, en comparación con otros que poseen un proceso más largo y complejo. Estos, se enfocan en detalles mas pequeños y en características mas especificas, lo que requiere un mayor uso de recursos. [85]

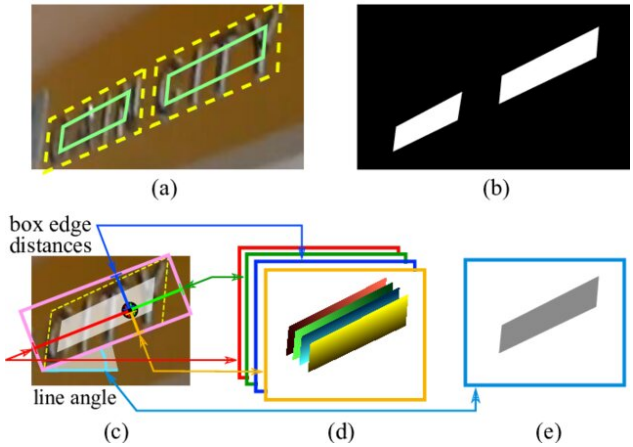


Ilustración 2.11: Esquema de EAST. Fuente: *EAST: An Efficient and Accurate Scene Text Detector*

Fourier Contour Embedding (FCENetwork).

Este modelo destaca por utilizar la *transformada de Fourier* para la representación de los bordes de las posibles palabras. La ventaja que nos ofrece emplear esta fórmula matemática consiste en la capacidad de poder hacer frente a aquellas regiones con formas complejas, filtrando además, aquellas zonas que presenten cierto ruido.

A diferencia de los modelos tradicionales, que se centran en la predicción de las cajas delimitadoras, este algoritmo predice los coeficientes de las series de Fourier que representan dichos contornos, ofreciendo de esta manera una mayor precisión. [86, 28]

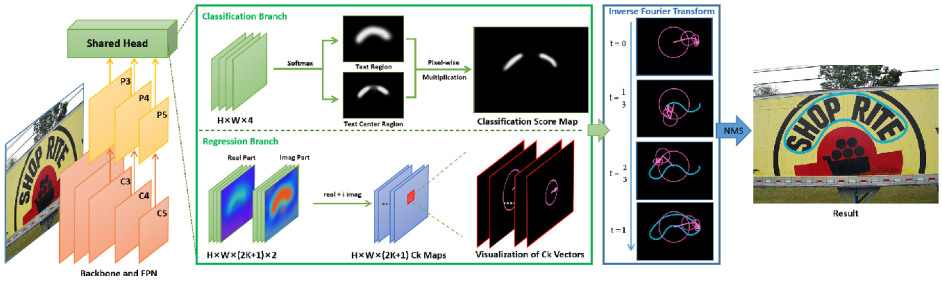


Ilustración 2.12: Esquema de FCENetwork. Fuente: *Fourier Contour Embedding for Arbitrary-Shaped Text Detection*

Progressive Scale Expansion Network (PSENet).

PSENet es un modelo que destaca por su capacidad de manejar textos que se encuentren muy juntos o que posean tamaños y formas dispares. A diferencia de los modelos tradicionales, esta red lo que realiza es, partiendo de un determinado punto, expande las regiones de manera iterativa a lo largo de todo el texto, es decir, por cada punto se expande una región que podrían llegar a solaparse con otras, lo que indicaría que todas esas regiones forman parte de la misma palabra. [74]

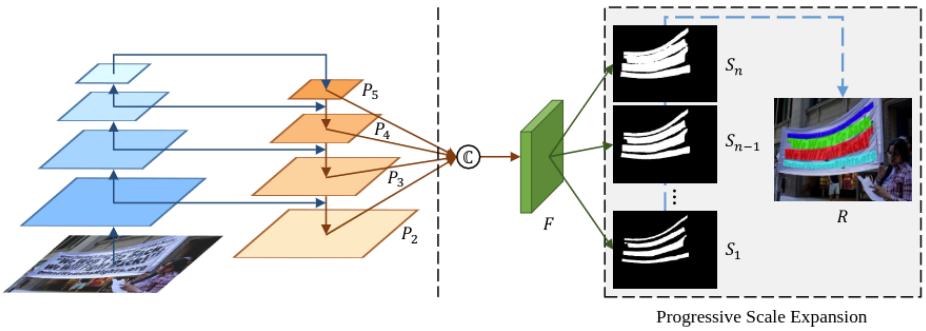


Ilustración 2.13: Esquema de PSENet. Fuente: *Shape Robust Text Detection with Progressive Scale Expansion Network*

Single-Shot Arbitrarily-Shaped Text Detector (SAST).

El siguiente modelo se caracteriza por la capacidad de poder detectar textos que se encuentren con cierta curvatura o incluso con deformidad, además de ofrecer tiempo de

ejecución cercanos al tiempo real, facilitando así su integración en aquellas aplicaciones en la que este detalle sea primordial.

Al igual que en el *EAST*, *SAST* se centra en la predicción del contorno de los textos mediante la utilización de un bloque denominado *Bloque de Atención al Contexto*, el cual almacena la mayor cantidad posible de información del contexto sobre cada píxel en el borde, mejorando de es manera la precisión en el momento de combinarlas. Asimismo, también cuentan con la integración de nuevas medidas geométricas las cuales, tras segmentar las regiones, y asignarles etiquetas con sus atributos, generan un vector que representa la inclinación que posee dicha región tanto desde el centro, como desde los vértices.

Finalmente y siguiendo la dirección generada, se reconstruye la cadena de texto, y el usuario acaba obteniendo la representación en la imagen de las regiones con la totalidad de caracteres que representa la palabra completa, junto con los puntos que indican el borde de unión de cada una de las regiones. [72]

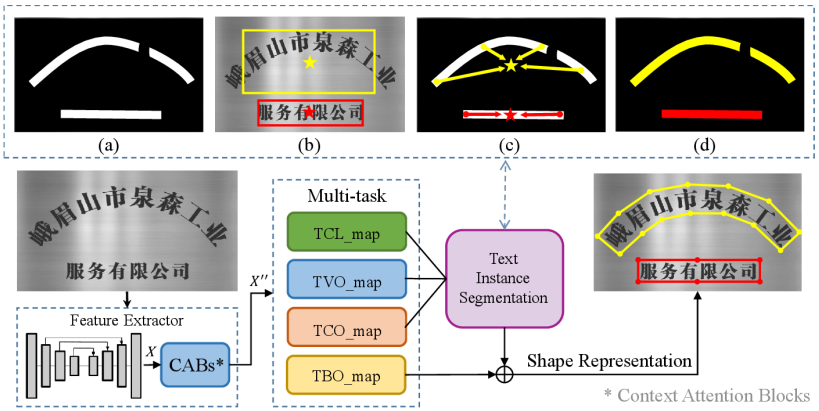


Ilustración 2.14: Esquema de SAST. Fuente: *A Single-Shot Arbitrarily-Shaped Text Detector based on Context Attended Multi-Task Learning*

Text Snake.

Siguiendo la analogía del movimiento de una serpiente este modelo, se caracteriza por la flexibilidad que tiene el algoritmo para la detección de texto en todas las direcciones, bien sea de manera horizontal, vertical o en curvas.

La representación de las regiones se lleva a cabo mediante la red *VGG16*, una red convolucional compuesta por 3 capas *fully-connected* y 13 capas convolucionales, que se encarga de la extracción de las características. Estas, generarán tres mapas: en primer lugar, cual es el punto central de la región a estudiar; un segundo mapa con la anchura posible que posee la serie de puntos que representa la palabra, y por último, un tercer mapa que almacenará la orientación en base al primer punto.[37]

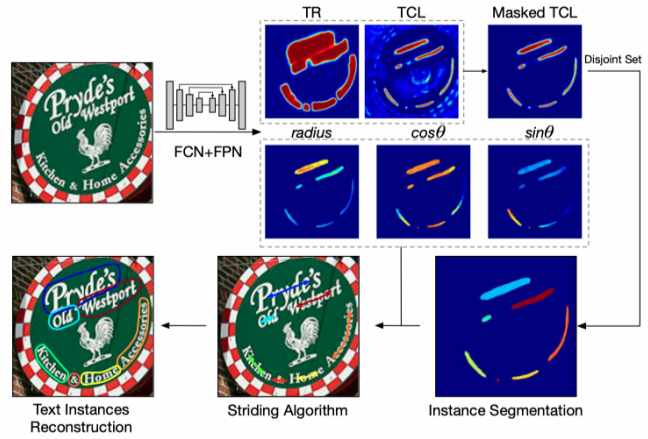


Ilustración 2.15: Esquema de Text Snake. Fuente: *TextSnake: A Flexible Representation for Detecting Text of Arbitrary Shapes*

2.3.2. Modelos neuronales para el reconocimiento.

Clip4STR.

Arquitectura diseñada específicamente para el proceso de reconocimiento de textos basada en la red neuronal **CLIP** *(Contrastive Language-Image Pretraining)*, la cual fue entrenada con imágenes que poseían textos en un formato mas común al humano en internet, es decir, imágenes cuyos textos representaban una escritura o expresiones coloquiales. Igualmente, también fue entrenado de la manera inversa, es decir, a partir de una serie de imágenes, la red debía seleccionar aquellas que se correspondían con la frase que el usuario dictaminaba.

Su arquitectura está dividida en dos partes importantes: por un lado, se encuentra la rama dedicada al codificador de la imagen, el cual hace uso de un transformador de vídeo (ViT) desarrollado en una red transfomer. Con esto, se segmenta la imagen en diferentes regiones mas pequeñas, almacenando la información sobre la posición original de cada una de estas regiones en la imagen original. Posteriormente, y a diferencia que el modelo *CLIP*, *CLIP4STR*, además de obtener el valor de la clase del texto, se extraen, a su vez, el resto de características de las regiones, como puede ser la inclinación del texto en referencia a un punto referencial o la distancia entre cada región. Como podemos ver en la imagen 2.16, la segunda rama hace referencia al codificador del texto. [84]

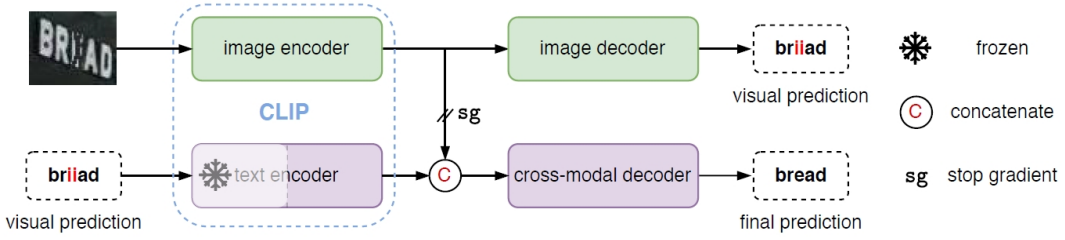


Ilustración 2.16: Esquema de CLIP4Str. Fuente: *CLIP4STR: A Simple Baseline for Scene Text Recognition with Pre-trained Vision Language Model*

StarNet.

Starnet combina una arquitectura de red neuronal convolucional junto con una red neuronal recurrente, aplicando a su vez la técnica de *Clasificación Temporal Conexista* (CTC). De esta manera, en un primer lugar, se extraen todas las características y atributos que permiten la posterior unificación de los segmentos que forman las palabras. Posteriormente, en las capas recurrentes, es donde se captura las dependencias gracias a su capacidad de almacenar las secuencias que ya han sido procesados. En la ultima capa, es donde nos encontramos con el módulo CTC, el cual se encarga interpretar y representar las salidas obtenidas de las capas recurrentes según el valor de confianza que se le ha asignado previamente.

Este modelo destaca por la flexibilidad que ofrece al trabajar con entradas que posean diferentes longitudes gracias a la unificación de las capas convolucionales con las recurrentes y el capa CTC. [36]

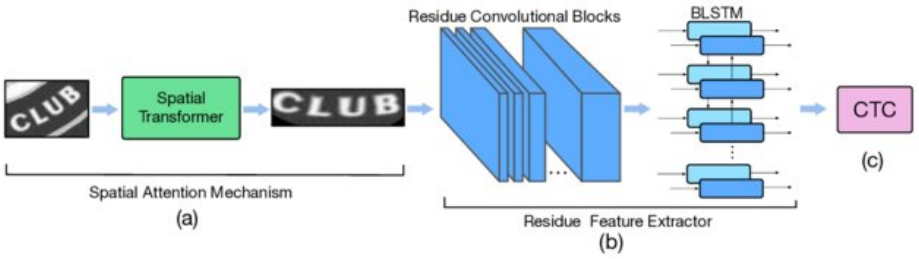


Ilustración 2.17: Esquema de Starnet. Fuente *STAR-Net: A SpaTial Attention Residue Network for Scene Text Recognition*

Robust Scene Text Recognition with Automatic Rectification (RARE).

Algoritmo diseñado para el reconocimiento de textos que posean formas irregulares o distorsionado que consta de una *red de transformación espacial* (STN), encargada de realizar un preprocesamiento a la imagen de manera que las diferencias entre el fondo y el texto sean más notable. Además, aplica todas aquellas transformaciones que sean necesarias en el caso de que por ejemplo, éste posea una curvatura, la *SRN* es la capa encargada de realizar el reconocimiento de la imagen, completando así el proceso de interpretación. [63]

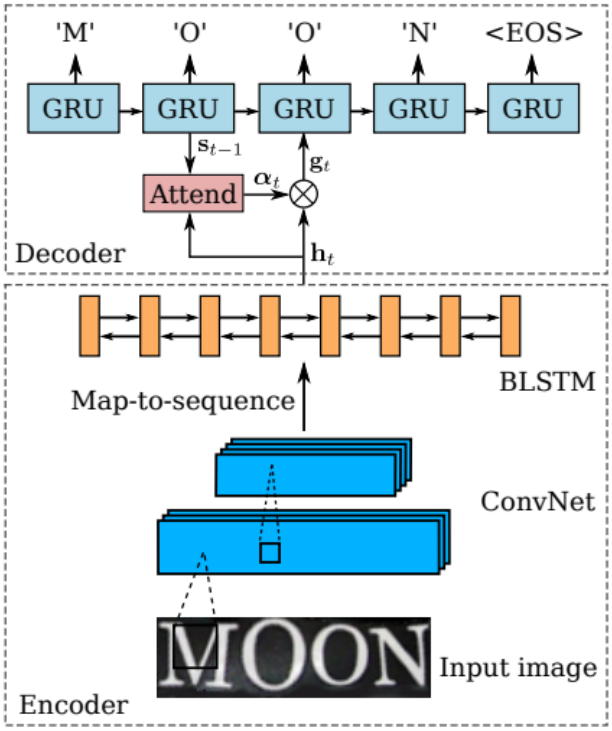


Ilustración 2.18: Esquema de Rare. Fuente *Robust Scene Text Recognition with Automatic Rectification*

Semantic Reasoning Networks (SRN).

Este modelo introduce las relaciones semánticas en el reconocimiento de texto, con el fin de no sólo basarse en las diversas características visuales presentes en las imágenes utilizadas como datos de entrada. Antes de realizar las operaciones semánticas, se ejecuta un preprocesamiento de la imagen en la que se *tokeniza* generando de esta manera un mapa de vectores que representa cada uno de los caracteres individuales.

En cuanto a su arquitectura está formada por cuatro módulos principales: el primero consiste en el *backbone*. El segundo es el denominado módulo de atención visual paralela (PVAM), a diferencia de los métodos tradicionales que dependen del estado oculto para almacenar toda la secuencia, utiliza el orden normal de lectura, lo que permite calcular de forma incremental las dependencias entre los términos a medida que se avanza por el mapa de vectores, mejorando significativamente la eficiencia en el tiempo de ejecución. A continuación se encuentra el módulo de razonamiento semántico global (GSRM), que analizará y relacionará los conceptos semánticos del texto. Por último, se encuentra el módulo del decodificador de fusión visual-semántica (VSFD) que combinará las características visuales con la informacion semántica para generar la salida. [78]

No-Recurrence sequence-to-sequence Text Recognizer (NRTR).

En este modelo, se introduce el concepto de arquitectura *codificador-decodificador*, eliminando la necesidad de empelar métodos tradicionales como las capas recurrentes. En el

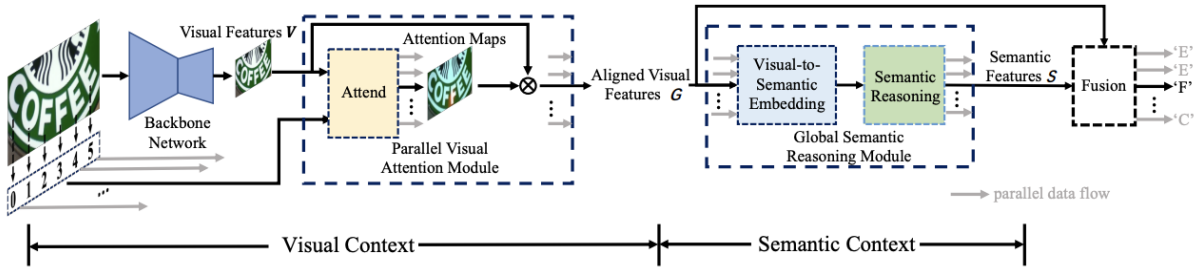


Ilustración 2.19: Esquema de SRN. Fuente *Towards Accurate Scene Text Recognition with Semantic Reasoning Networks*

codificador, se implementa el proceso de *autoatención apilada* junto con redes ”transformer”, que permite enfocar la atención en distintas partes de las entradas simultáneamente. De esta manera, se genera que algunas palabras posean mayor peso que otras, optimizando la representación de características relevantes.

Posteriormente, en el decodificador, la salida generada por el codificador es utilizada para producir un texto legible por parte del usuario. En esta etapa, se aplica el *mecanismo de atención cruzada*, característico de las redes ”transformer”, que establece conexiones entre la información procesada por el codificador y la información generada por la red hasta ese momento, mejorando la coherencia en la reconstrucción del texto.

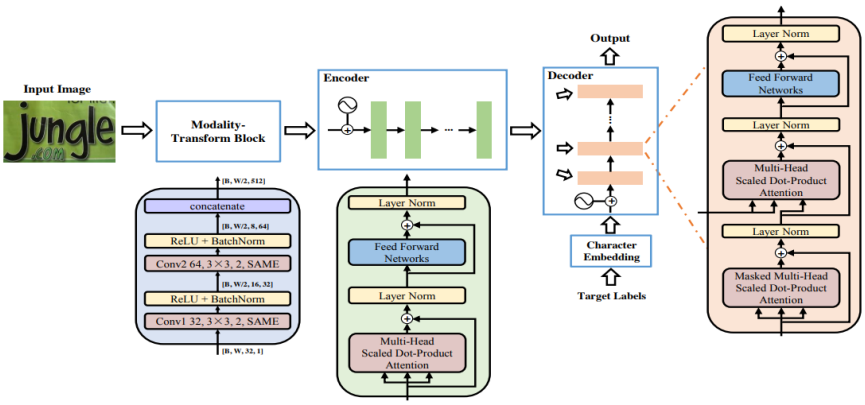


Ilustración 2.20: Esquema de NRTR. Fuente *NRTR: A No-Recurrence Sequence-to-Sequence Model For Scene Text Recognition*

Show, Attend and Read (SAR).

SAR combina redes neuronales profundas junto con mecanismos de atención para el reconocimiento de textos. Este modelo está compuesto por tres componentes principales. En primer lugar, se encuentra la fase de extracción de características *(show)*, que utiliza redes convolucionales. A continuación, nos encontramos la fase de aplicación del mecanismo de atención *(attend)*, diseñado para ayudar al modelo a centrarse únicamente en las regiones que contengan siluetas que puedan asemejarse a textos, ignorando de esta manera zonas irrelevantes como pueden ser los fondos donde se ubiquen edificios. La última fase es la referente a la decodificación del texto *(read)*, donde se transforman, las características visuales obtenidas de las fases anteriores, en una secuencia de caracteres, completando así el proceso. [33]

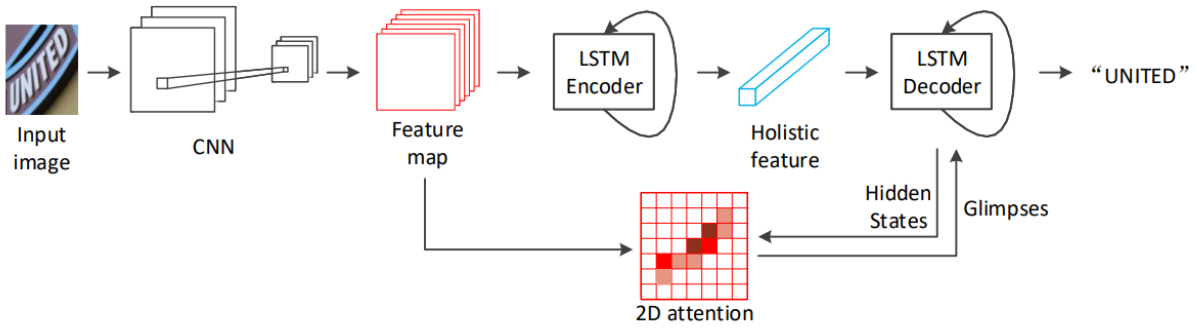


Ilustración 2.21: Esquema de SAR. Fuente *Show, Attend and Read: A Simple and Strong Baseline for Irregular Text Recognition*

Semantics Enhanced Encoder-Decoder Framework for Scene Text Recognition (SEED).

El *framework* aquí descrito introduce al actual enfoque de codificador-decodificador, una mejora semántica en el proceso de reconocimiento, permitiendo de esta manera una mayor comprensión del contexto.

Su arquitectura está compuesto por cuatro módulos principales: módulo de rectificación, encargado de ajustar las imágenes de entrada para facilitar el procesamiento; módulo codificador, que emplea múltiples *CNN* combinadas con un mecanismo de atención, como se mencionó en anteriores modelos; módulo semántico, que predice el resultado a partir de la salida obtenida del codificador, y que se usará como entrada para el último módulo; y el modulo decodificador, que emplea redes "transformer" y aplica un mecanismo de atención cruzada para generar el texto final. [59]

Scene Text Recognition with a Single Visual Model (SVTR)

Esta arquitectura adopta una modalidad híbrida que elimina la división tradicional, y genera dos bloques diferenciados: módulo visual que realizaba la extracción de las carac-

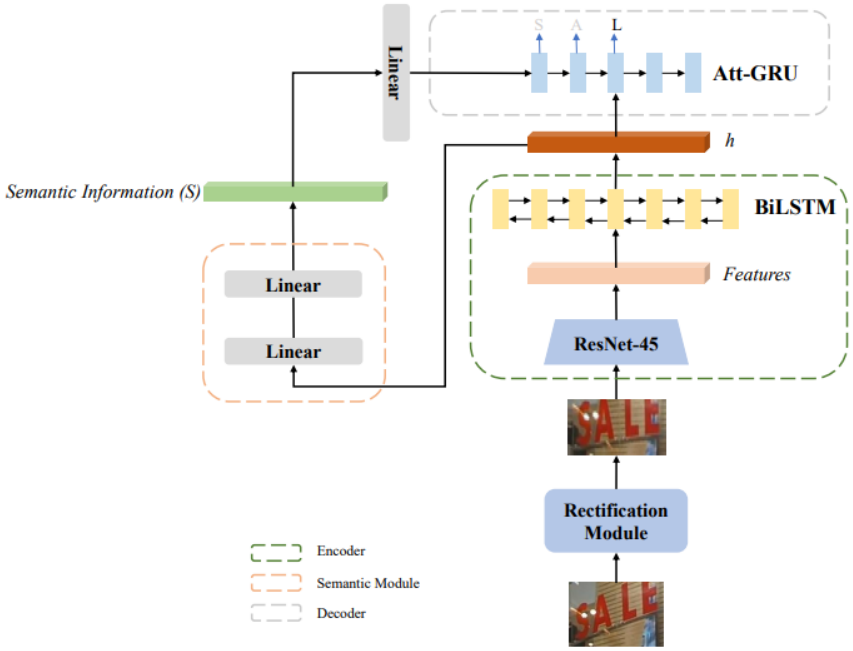


Ilustración 2.22: Esquema de SEED. Fuente *SEED: Semantics Enhanced Encoder-Decoder Framework for Scene Text Recognition*

terísticas, y el módulo secuencial encargado de la traducción a texto.

El flujo de trabajo, a rasgos generales, comienza con la *tokenización* las imágenes mediante la aplicación de parches. Inicialmente, se segmenta la imagen en pequeñas regiones que representen los posibles caracteres, un proceso denominado como *componentes de caracteres*. Seguidamente y de manera iterativa, se combinan y fusionan esas regiones formando patrones que representen la silueta del carácter al completo. Este procedimiento se realiza tanto a nivel global, integrando partes de todos los componentes, como a nivel interno, es decir, buscándolos dentro de la misma región. [12]

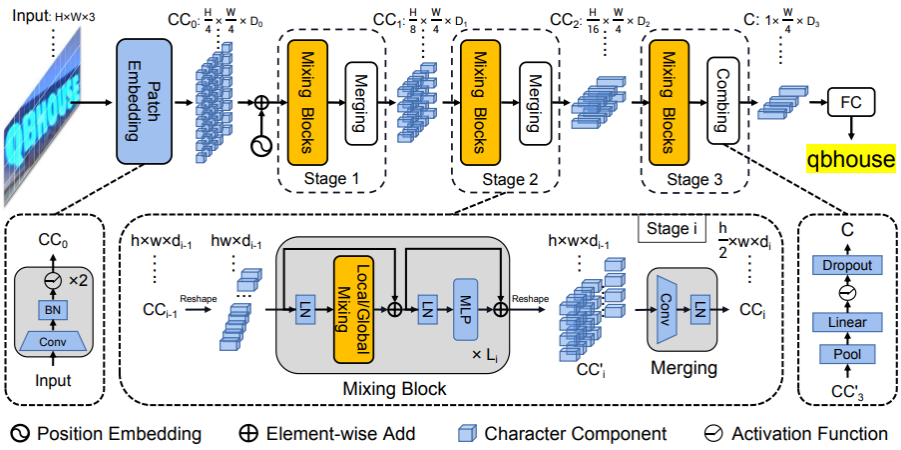


Ilustración 2.23: Esquema de SVTR. Fuente *SVTR: Scene Text Recognition with a Single Visual Model*

Vision Transformer for Fast and Efficient Scene Text Recognition (ViSTR)

Se introduce el *transformador por visión*, que procesa las imágenes como una secuencia de parches individuales que son procesados por las redes "transformer" que conforman su arquitectura, permitiendo identificar relaciones entre las partes de la imagen. Asimismo, también se incorpora la *técnica de atención visual* para enfocarse en regiones especificas bien sea para identificar textos completos como fragmentos de ellos.

Su utilidad principal radica en la capacidad de manejar texto que tenga orientaciones variadas, ya sea que se encuentren siguiendo una línea horizontal o presenten alguna curvatura. [4]

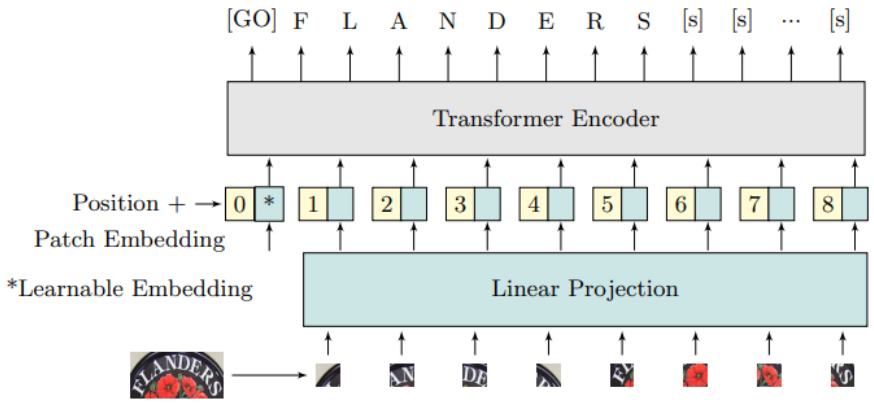


Ilustración 2.24: Esquema de ViSTR. Fuente *Vision Transformer for Fast and Efficient Scene Text Recognition*

Read Like Humans: Autonomous Bidirectional and Iterative Language Modeling for Scene Text Recognition.

Este modelo destaca por la capacidad de leer la imagen en dos direcciones, es decir, realiza la lectura tanto en el orden natural, de izquierda a derecha, como en el sentido contrario, de derecha a izquierda. Para ello, se hace uso de redes neuronales recurrentes de memoria a largo plazo (LSTM), que captura y almacena las dependencias que ocurran en las diferentes etapas e iteraciones.

Este proceso se realiza de manera iterativa, lo que significa que, sobre una misma imagen, se realizan múltiples repeticiones de la extracción de características, Durante estas iteraciones, el modelo ajusta y corrige los posibles errores derivados de las predicciones previas. Finalmente, en el decodificador, utilizando la salida de la fase anterior, generamos el texto resultado con una mayor precisión en comparación con los modelos tradicionales que procesa la imagen una única vez.[15]

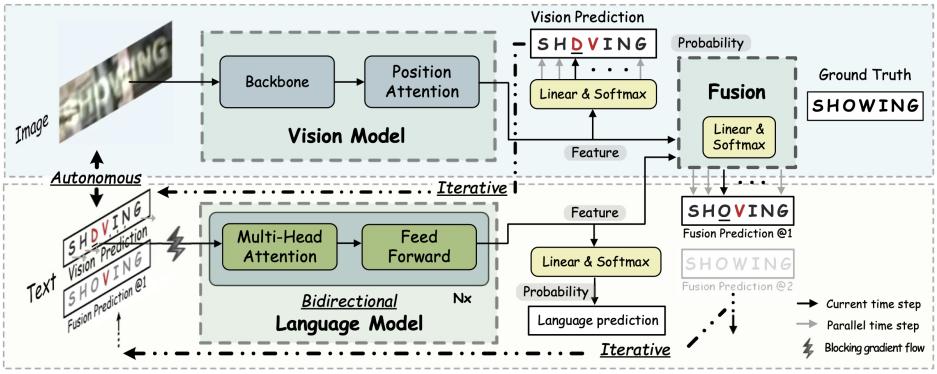


Ilustración 2.25: Esquema de Autonomous. Fuente *Read Like Humans: Autonomous, Bidirectional and Iterative Language Modeling for Scene Text Recognition*

Visual Language Modeling Network (VisionLAN).

VisionLan destaca por sus resultados en el reconocimiento de texto presentes en carteles urbanos y señales de tráfico. Esto se logra mediante la combinación de dos módulos: primero, un módulo visual que se enfoca en aspectos destacados de la imagen como son los colores o formas que éstas poseen; y un módulo lingüístico que aplica técnicas propias del procesamiento del lenguaje natural para establecer relaciones entre la información obtenida del módulo visual, con los posibles textos que puede estar presente en las formas detectadas. [75]

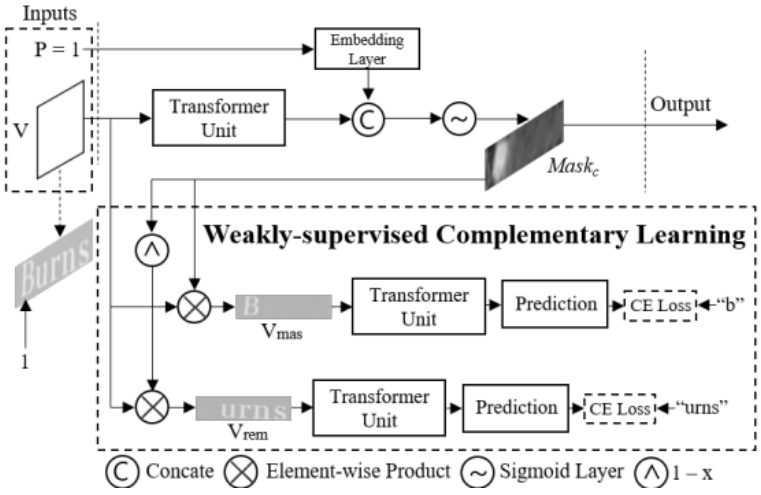


Ilustración 2.26: Esquema de VisionLan. Fuente *From Two to One: A New Scene Text Recognizer with Visual Language Modeling Network*

Structure-Preserving Inner Offset Network for Scene Text Recognition (SPIN).

Con esta arquitectura se busca abordar el problema de reconocimiento de texto considerando las variaciones cromáticas posibles que podrían afectar a la imagen, en lugar de centrar la atención en las posibles deformidades que ésta pueda prestar, lo que la diferencia del resto de modelos comentados.

Introduce un nuevo módulo, el cual puede integrarse, ejecutarse o incluso entrenarse de manera independiente a la arquitectura principal, teniendo en cuenta que éste, debe suceder antes del proceso de reconocimiento. [82]

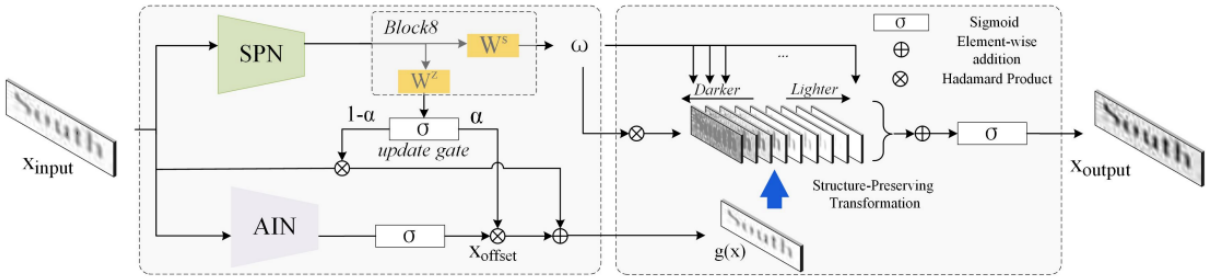


Ilustración 2.27: Esquema de SPIN. Fuente *SPIN: Structure-Preserving Inner Offset Network for Scene Text Recognition*

RobustScanner.

En este algoritmo se centra en mejorar específicamente la parte de dedicada a la decodificación para obtener texto legible. Para ello, se introducen dos módulos clave: el de posición, que almacena la ubicación exacta del carácter en la imagen, permitiendo, a la hora de unificarlos, determinar de que lugar original procedía cada letra que forma la cadena de salida. El segundo módulo es el de atención, cuya función es determinar en que momento se ha traducido el texto de la imagen, proporcionado así el contexto necesario para poder indicar si una palabra se había generado antes que otras. [79]

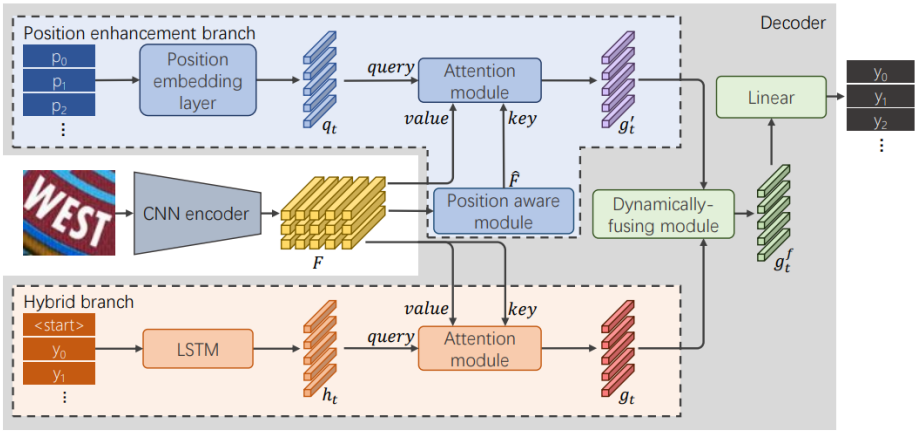


Ilustración 2.28: Esquema de RobustScanner. Fuente *RobustScanner: Dynamically Enhancing Positional Clues for Robust Text Recognition*

Reciprocal Feature Learning via Explicit and Implicit Tasks in Scene Text Recognition (RFL).

RFL es un modelo cuyo objetivo es mejorar el proceso de aprendizaje a partir de las características visuales de la imagen, mediante la inclusión de dos parámetros: tarea explicita y tareas implícitas. La principal diferencia entre ambos radica en que, mientras que en las explícitas se realiza el reconocimiento de los caracteres directamente gracias a que la imagen posee una buena calidad y el texto es fácilmente legible. En las implícitas, en cambio, no se realiza un reconocimiento directo, sino que se generan subtareas que permiten a la red aprender sobre ciertos aspectos como pueden ser si el texto tiene un estilo cursiva o negrita.

Una vez completada esta etapa, el resto del proceso de *OCR*, persigue el mismo flujo que los modelos tradicionales, con la principal diferencia de que, en el apartado de fusión de las características, no solo se combinan las característica visuales, sino que además, se incluye el aprendizaje obtenido de las tareas implícitas. [29]

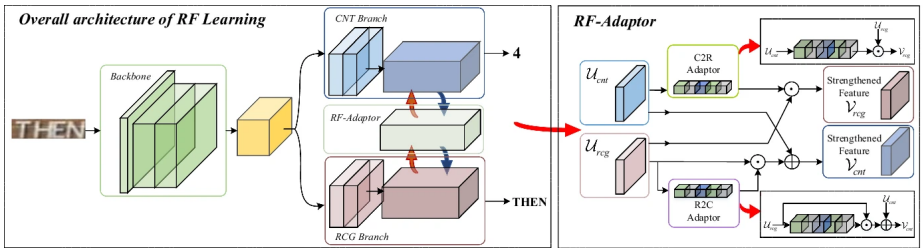


Ilustración 2.29: Esquema de RFL. Fuente *Reciprocal Feature Learning via Explicit and Implicit Tasks in Scene Text Recognition*

2.3.3. Modelos neuronales de dos etapas.

EasyOCR

Librería especializada en la detección y reconocimiento de caracteres, desarrollada y mantenida por Jaided AI [2]. Su principal ventaja radica en el soporte para más de 80 idiomas, además de ofrecer múltiples modelos preentrenados. De igual manera, su integración con Python es mas sencillo debido a que se basa en la librería de aprendizaje profundo, PyTorch.

En cuanto a su arquitectura, EasyOCR, utiliza redes neuronales convolucionales recurrentes (CRNN), que estan compuestas de dos tipos de redes neuronales: por un lado, una red neuronal convolucional profunda (DCN), y por otro, red neuronal recurrente (RNN).

Siguiendo con el proceso que sigue el modelo, éste es bastante sencillo, ya que de manera iterativa, inicializa el modelo, y utilizando la imagen o imágenes que se le haya introducido por parámetros, aplica la fase de detección con el objetivo de encontrar y obtener la caja delimitadora de aquellas regiones que pueden contener los posibles textos. A continuación, se procede al reconocimiento, obteniendo como salida, tanto el texto como sus coordenadas en la imagen junto con un valor de confianza del resultado. [27].

Aunque este modelo destaca por su facilidad de uso y configurabilidad, también presenta algunas desventajas, tales como la variabilidad de los resultados en situaciones imperfectas, como por ejemplo, en aquellos casos en los cuales las imágenes capturadas no se encuentran bien iluminadas o presentan borrosidad.

EasyOCR esta formado por tres componentes principales que permiten realizar el proceso de detección y reconocimiento de caracteres de manera automática: [38, 39]

1. **Extracción de características.** En este primer componente se procede a la conversión de los datos de entrada, es decir, se transforman las imágenes, en un conjunto de datos que pueda ser manipulado por el modelo. Asimismo, también se procede a identificar todas aquellas características representativas que sean referentes a los caracteres. Para ello, se hace uso de redes residuales (ResNet) y de los Grupo de Geometría Visual (VGG). Mientras que con ResNet se utilizan *conexiones residuales* que permiten la omisión de ciertas capas, facilitando el aprendizaje y permitiendo profundizar más en la red; con VGG, gracias a su arquitectura formada por filtros de convolución de tamaño 3x3, ayudan a la detección de bordes o patrones que representen caracteres o dígitos.

2. **Etiquetado de la secuencia**. En el segundo componente, se hace uso de redes neuronales recurrentes (RNN) de tipo Long Short-Term Memory (LSTM), que son capaces de evitar el problema del desvanecimiento del gradiente, comentado en el apartado 2.2.2. Cada nodo de estas RNN contiene una célula de memoria que almacenará la

información obtenida de las entradas anteriores. El propósito principal de este proceso, es identificar y clasificar las regiones detectadas como texto o no texto, para su posterior tratamiento en la etapa de decodificación.

3. **Decodificación**. En este último componente, se procede a "traducir" a texto todo lo que se encuentre contenido en la regiones clasificadas como texto en el componente previo. Para ello, si se hace uso de los modelos preentrenados, el sistema buscará y asociará *píxeles* que se asemejen a los utilizados en su entrenamiento, determinando de esta forma, a qué letra o numero corresponden. Este proceso se realiza de forma recursiva, recorriendo todas las regiones etiquetadas como texto, unificando aquellos caracteres que se encuentren dentro de una misma zona para generar las palabras o textos completos.

En resumen, cada uno de los tres componentes corresponde a cada una de las etapas clásicas del OCR, donde, la extracción de características, hace referencia al preprocesado de la imagen identificando aspectos relevantes que le podamos transferir al proceso de etiquetado de la secuencia, que consiste en delimitar las zonas anteriormente categorizadas con aspectos de interés. Por último, se produce la decodificación, con el objetivo de convertir a lenguaje natural la salida de la etapa anterior.

PaddleOCR

PaddleOCR es una librería de código abierto diseñada para realizar tareas de detección y reconocimiento de caracteres basada en la librería *PaddlePaddle (PArallel Distributed Deep LEarning)* y desarrollada por la empresa Baidu. Una de sus principales ventajas es la compatibilidad con más de 80 idiomas, ademas de contar con múltiples versiones que permiten adaptarse a diferentes contextos y necesidades. Las diferentes versiones de PaddleOCR aunque se basan en redes neuronales convolucionales, éstos se diferencian entre ellos en el tamaño y rendimiento: por un lado, los modelos ligeros ofrecen una velocidad de ejecución superior aunque poseen una menor precisión; mientras que, los modelos pesados, aseguran una mejor *performance*, pero consumen mayor cantidad de recursos. [57]

Para la etapa de detección, PaddleOCR cuenta con una amplia variedad de algoritmos que puedan seleccionarse según las necesidades y el contexto del que provenga las imágenes. Entre esos algoritmos se encuentran: *Differentiable Binarization*(DB) y su versión mejorada *Differentiable Binarization and Adaptive Scale Fusion*(DB++), *Deep Relational Reasoning Graph Network* (DRRG), *Efficient and Accurate Scene Text Detector* (EAST), *Fourier Contour Embedding* (FCENetwork), *Progressive Scale Expansion Network* (PSENet), y *Single-Shot Arbitrarily-Shaped Text Detector* (SAST). Todos estos modelos se encuentran explicados en profundidad en los apartados 2.3.1. [56]

Del mismo modo, para la etapa de reconocimiento, esta librería también ofrece una varie-

dad de algoritmos que se diferencia entre ellos en la arquitectura de la red neuronal en la que se basan. Los algoritmos son: *Conventional Recurrent Neuronal Network* (CRNN), *Rosetta*, *StarNet*, *Robust Scene Text Recognition with Automatic Rectification* (RARE), *Semantic Reasoning Networks* (SRN), *No-Recurrence sequence-to-sequence Text Recognizer* (NRTR), S*how, Attend and Read* (SAR), *Semantics Enhanced Encoder-Decoder Framework for Scene Text Recognition* (SEED), *Scene Text Recognition with a Single Visual Model* (SVTR), *Vision Transformer for Fast and Efficient Scene Text Recognition* (ViSTR), *Autonomous, Bidirectional and Iterative Language Modeling for Scene Text Recognition* (ABINet), *Visual Language Modeling Network* (VisionLAN), *Structure-Preserving Inner Offset Network for Scene Text Recognition* (SPIN), *RobustScanner*, y, *Reciprocal Feature Learning via Explicit and Implicit Tasks in Scene Text Recognition* (RFL). En el apartado 2.3.2, podemos encontrar una explicación mas extensa de cada uno de estos modelos. [56]

Por último, PaddleOCR, también ofrece la posibilidad de realizar la detección y reconocimiento de texto en un único proceso mediante el algoritmo *Point Gathering Network* (PGNet) [73]. Con este modelo se logra un procedimiento mas rápido y simple. El flujo de ejecución que sigue este modelo es el descrito a continuación: primero se identifican aquellos puntos de interés definidos como *puntos extremo a extremo*, los cuales representan los bordes delimitadores de la palabra que estamos analizando. A continuación, se procede a segmentar en caracteres individuales tras agrupar los pares de puntos anteriores formando subregiones, que, a su vez, forman cada una de las letras que componen la palabra. Posteriormente, y tras determinar el punto central de cada carácter, se procede a obtener cual es la orientación del texto (horizontal, vertical o invertida). Por ultimo, se realiza el reconocimiento obteniendo finalmente las palabras finales. Este enfoque es especialmente util en aquellos casos donde la velocidad es esencial.

KerasOCR

KerasOCR [32] es una biblioteca de código abierto desarrollada a partir de las librerías de aprendizaje profundo *Keras* [31] y *Tensorflow*, que permite obtener un alto rendimiento al poder hacer uso de GPU o CPU, dependiendo de las necesidades del entorno. Su arquitectura se basa en redes neuronales convolucionales recurrentes (CRNN), las cuales combinan redes neuronales convolucionales (CNN), para la extracción de las características relevantes de la imagen tras la aplicación de filtros, como pueden ser los bordes o patrones, junto con redes neuronales recurrentes (RNN), que se encargan de procesar esas características para generar como salida la cadena de texto correspondiente.

Uno de los componentes clave de su arquitectura es el modelo *Character-Region Awareness For Text detection* (CRAFT) [5, 1], diseñado para superar los límites de los métodos tradicionales que utilizan cajas delimitadoras para representar la región con texto. Este enfoque resulta problemático en el caso de que el texto estuviera presente en una curva. *CRAFT* aborda el problema analizando cada carácter de manera individual asignándole dos valores:

el *region score*, que representa la probabilidad de que una región contenga un carácter; y el *affinity score*, que evalúa la proximidad entre dos regiones. Esta combinación permite determinar si dos caracteres pertenecen a la misma cadena o, si por el contrario, son de diferentes textos o incluso letras individuales.

De igual manera, este modelo nos ofrece la posibilidad de hacer uso de modelos preentrenados que se diferencian entre ellos, por la arquitectura y complejidad que poseen. De igual manera, se puede entrenar nuestro propio modelo con un *dataset* previamente etiquetado, lo que resulta bastante útil para aplicaciones en un entorno especifico como puede ser el definido en el desarrollo de este trabajo.

PyTesseract

PyTesseract es el motor para la librería Tesseract-OCR, desarrollado por Hewlett Packard en 1980, siendo de código abierto en 2005 y mantenido desde el 2006 por Google, centrado específicamente para el proceso de detección y reconocimiento de caracteres en documentos. [50]

En sus primeras versiones, Tesseract mostraba un rendimiento sobresaliente en entornos controlados con imágenes de alta calidad, sin apenas ruido, y el texto se encontraba en un primer plano. En posteriores versiones, se fueron superado limitaciones, ademas de ampliar el soporte a datos de entrada que se encuentren en más de 100 idiomas. Asimismo, se añadió un modelo de aprendizaje profundo basado en una red de memoria a largo plazo (LSTM), junto con la utilización de redes neuronales recurrentes (RNN) para incrementar la precisión del modelo.[80]

Cuando se instala PyTesseract, también se incluye la librería OpenCV, dado que ambas herramientas trabajan conjuntamente con el objetivo de ofrecer un sistema sólido en todas las etapas de desarrollo. El flujo comienza con el preprocesamiento de la imagen haciendo uso de las herramientas que ofrece OpenCV, donde se aplica tanto escala de grises como la eliminación del ruido, mejorando así la calidad de imagen de entrada. A continuación, PyTesseract, realiza la segmentación de la imagen de manera que se genera un mapa de bits que identifica cada una de las regiones de interés con texto, para posteriormente subdividirlas a nivel de carácter. Seguidamente, a cada uno de estos caracteres, es procesado con el modelo basado en LSTM. Finalmente, los caracteres reconocidos se combinan para reconstruir la cadena referente al texto original. [51, 64]

Asimismo, cabe destacar el apartado de configuración personalizada que nos brinda este modelo. Por un lado, se ofrece la posibilidad de seleccionar el *modo de operación* en el que queremos que se ejecute el algoritmo, teniendo en cuenta que consta de dos tipos de motores: *Legacy Tesseract Enginer*, que hace uso de las técnicas clásicas de clasificación y reconocimiento; o bien, *LSTM engine*, que se basa en el aprendizaje profundo y en la

aplicación de las redes neuronales de memoria a corto y largo plazo. Los modos de operación son los siguientes: [60].

- *Modo 0*: utiliza solo el motor Legacy Tesseract
- *Modo 1*: utiliza solo el motor LSTM
- *Modo 2*: combina ambos motores con el objetivo de mejorar tanto en precisión como rendimiento.
- *Modo 3*: modo por defecto, y el cual seleccionará el motor de ejecución en función de los recursos disponibles del sistema

Por otro lado, también se permite la selección del *modo de segmentación de páginas*, el cual define la estructura de los datos de entrada, mejorando de esta manera el rendimiento y la precisión del modelo. Existen 10 modos de segmentación, entre los cuales, centrándonos en nuestro caso de estudio, destacan los siguientes: [60]

- *PSM=0*: analiza el grado de inclinación que tiene el texto en el contexto del idioma que le indiquemos
- *PSM=1*: modo por defecto, que analiza tanto la inclinación como la división del texto en caracteres individuales.
- *PSM=3*: modo predeterminado del sistema
- *PSM=7*: asume que solo hay una línea de texto (en nuestro caso, por ejemplo, esa linea de texto hace referencia a la matrícula)

MMOCR

Desarrollado por OpenMMLab y basado en PyTorch, MMOCR es un modelo de código abierto que ofrece dieciocho algoritmos tanto para la detección como para el reconocimiento de textos, así como para la extracción de características clave, como los puntos medios de cada caracteres.

Sus características principales es la capacidad de enfrentarse a diferentes entornos debido, al poder de elección por parte del usuario, a la variedad de algoritmos adaptables a diversas situaciones. Por un lado, se encuentra el *SingleStageTextDetector*, que concentra en una única etapa, la detección y localización del texto, mientras que por otro lado, algoritmos como *TextSnake* o *DBNet*, ofrecen soluciones a problemas más específicos. Mientras que el primero utiliza, para el proceso de detección, un enfoque basada en contornos continuos emulando el movimiento de una serpiente; el segundo, hace uso de un mapa de calor que se genera tras la segmentación y binarización de la zonas con textos, permitiendo diferenciar de una manera clara el fondo y las regiones relevantes.

En cuanto a la etapa de reconocimiento de texto, MMOCR también cuenta con diversos algoritmos adaptables a los diferentes entornos. Entre ellos destaca *ABINet*, un modelo capaz de reestructurar el texto reconocido al realizar lecturas bidireccionales, es decir, tanto de izquierda a derecha, como de arriba hacia abajo. Además incorpora el algoritmo *SAR* (Show, Attend and Read), que combina las redes neuronales convolucionales con mecanismos de atención, con el fin de poder mejorar el modelo al eludir ciertas irregularidades del fondo que puedan interferir en las muestras que realmente sean importantes.

Rosetta

Modelo desarrollado inicialmente por Meta con el fin de identificar y moderar las imágenes que se encuentran en la plataforma, evitando que pudieran ser ofensivas hacia los usuarios. Es por ello, que aprovechando la gran cantidad de contenido que se publica en la red social *Facebook*, Meta cuenta con un amplio *dataset* con el que entrenar la red. El sistema se ejecuta en dos etapas: detección y reconocimiento, considerando en ambas el contexto en el que el texto se encuentra.

Para la detección se hace uso de la red *Faster R-CNN*, que implementan y mejoran las redes convolucionales basadas en regiones, la cual aprende de manera simultánea a clasificar y localizar los objetos, acelerando y optimizando, el tiempo de entrenamiento de la red. Asimismo, este modelo integra una *Red de Propuesta de Regiones* (RPN), en la cual se generan un gran número de regiones basadas en las características obtenidas del bloque previo, y, las cuales, se clasificarán en base al nivel de confianza que presenten sobre si contienen o no texto. Posteriormente, se priorizará aquellas que posean un valor mayor y se aplicará el algoritmo de *supresión no máxima* (NMS), para evitar que existan regiones duplicadas o solapadas, consiguiendo de este modo una muestra más limpia para el proceso de reconocimiento.

En la fase de reconocimiento, este modelo aplica la técnica de *clasificación temporal conexionista* (CTC) ampliamente utilizada en sistemas de traducción de voz a texto. Esto permite que el algoritmo aprenda a reconocer las posibles secuencias de palabras con el desconocimiento de en que punto comienza o termina la palabra o frases.

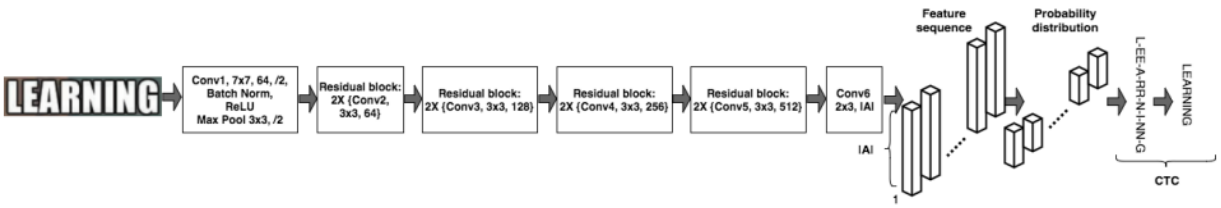


Ilustración 2.30: Esquema de Rosetta. Fuente: *Rosetta: Large scale system for text detection and recognition in images*

2.4. Métricas de evaluación

Dado que los resultados obtenidos tras la ejecución de los modelos comentados previamente, cuando se aplican sobre un entorno tan específico como el que nos acontece, pueden contener errores bien sea a nivel de carácter individual como a nivel de palabra completa, resulta fundamental tener un control adecuado sobre ellos, con el objetivo de garantizar la calidad de las detecciones, obteniendo de esta manera, un modelo eficaz y seguro.

Es por ello, que existen una serie de métricas que permiten analizar y cuantificar el desempeño del sistema. Cada una de ellas ofrece información sobre distintos aspectos como pueden ser la tasa de error bien por palabra, bien por carácter, la tasa de acierto propiamente del programa, o el número de modificaciones y transformaciones que ha efectuado el sistema para obtener una palabra incorrecta en base a la forma correcta que debería poseer.

2.4.1. WER

Word Error Rate o Tasa de Error por Palabra, se basa en el concepto de la distancia de Levensthein, y nos indica cual es el número total de palabras erróneas que se han transcrito tras la ejecución del programa. Cuanto menor sea este valor, mayor precisión tendrá el sistema y los resultados serán mas fiables en base al entorno real.[25, 17, 48]

$$\mathrm{WER} = \frac{\mathrm{i}_w + \mathrm{s}_w + \mathrm{d}_w}{\mathrm{n}} \tag{2.1}$$

En la fórmula anterior, $\mathrm{i_w}$ representa las inserciones de palabras realizadas por el reconocimiento de caracteres, $\mathrm{s_w}$, se refiere a las sustituciones de palabras, y $\mathrm{d_w}$, corresponde con las palabras eliminadas, es decir, aquellas que si se encuentran en el fichero *ground truth*, pero no en el fichero de salida tras la ejecución. Por último, en cuanto a n, representa el total de palabras del documento original.

2.4.2. CER

Character Error Rate o Tasa de Error por Carácter, al igual que el *WER*, se basa en la distancia de Levensthein, pero en este caso, obtenemos cual es el porcentaje de caracteres erróneos en función del total de caracteres del documento. [25, 48].

$$CER = \frac{i + s + d}{n} \tag{2.2}$$

En esta fórmula, i, son las inserciones de caracteres, es decir, aquellos que no se encuentran en la palabra original pero han sido añadidos en la de salida; s, son las sustituciones (caracteres que han sido reemplazados por otros); y d, son los caracteres eliminados en la palabra final. Por último, n, representa el total de caracteres originalmente.

2.4.3. WIP

Word Information Preserved o denominada en español como Tasa de Información por Palabra Preservada, es una métrica que mide cuánta información del texto original se conserva de manera íntegra en el texto resultado tras ejecutar el modelo. Un valor más alto, indica un mejor rendimiento del sistema.

Asimismo, WIP es una métrica complementaria al WER ya que aporta una mayor precisión a los resultados, debido a que ésta evalúa la proporción de palabras necesarias para preservar el significado y el contexto del texto original, mientras que el WER, se enfoca exclusivamente en el numero de errores por palabra en el texto. [48]

$$WIP = \frac{H}{N_1} \cdot \frac{H}{N_2} \approx \frac{I(X,Y)}{H(Y)} \tag{2.3}$$

En la formula superior, H representa la cantidad de información presente en el texto de referencia. N_1 es el número de palabras del texto original, y N_2 es el número de palabras del texto evaluado. En cuanto a $I(X,Y)$, representa la cantidad de información que se encuentra en el original y que el modelo ha logrado conservar tras aplicar el OCR; por último, $H(Y)$ es la cantidad total de información generada por el modelo.

2.4.4. WIL

La tasa de Información por Palabra Perdida (*Word Information Lost*), se entiende como la complementaria a WIP. Se define como la cantidad de información que se ha perdido durante el proceso, obteniéndose directamente a partir del WIP. [48]

$$WIL = 1 - WIP \tag{2.4}$$

2.4.5. MER

Math Error Rate, Tasa de Error de Coincidencias, corresponde a la última de las métricas utilizadas para evaluar el rendimiento de los modelos comentados anteriormente. Representa el porcentaje de palabras incorrectas con respecto al número total de palabras originales. Dicho de otra manera, es la proporción de palabras incorrectas calculadas a partir de las palabras reconocidas correctamente. [48, 17]

$$MER = \frac{s + d + i}{N = h + s + d + i} = 1 - \frac{h}{N} \tag{2.5}$$

En este caso, mientras que s, es el número de sustituciones, d, el de eliminaciones e i, el de inserciones, con h, indicamos el número de palabras que fueron reconocidas de manera correcta, y N, el número total de palabras que deberían reconocerse.

2.4.6. Distancia de Levensthein

Previo a comentar que es la distancia de Distancia de Levensthein, y sus fundamentos, debemos entender el termino de distancia. Según la Real Academia Española de la Lengua (RAE), la distancia se define como: "*Diferencia, desemejanza notable entre unas cosas y otras.*" [9]

En el caso que nos ocupa, la distancia hace referencia al número mínimo de transformaciones que se realizan para convertir la palabra de referencia en la palabra obtenida. Estas transformaciones incluyen inserciones, eliminaciones o sustituciones de caracteres. Basándonos en la definición de la RAE, la distancia de Levensthein, nos indica mediante un valor numérico, cuán de similar son dos palabras. Cuanto mas cercano a cero sea el valor, estaremos frente a dos palabras semejantes; en cambio, si ese valor es elevado, concluiremos que difieren en gran parte de sus caracteres.

Por ejemplo, consideremos la palabra de referencia "hola" y, tras ejecutar el modelo, obtenemos "oca". La distancia de Levensthein resultado entre esas palabras es de 2, ya que han ocurrido dos transformaciones: 1. Eliminación de la "h", obteniendo de esta manera "ola"

$$\text{hola} \rightarrow \text{ola}$$

2. Sustitución de "l" por una "c"

$$\text{ola} \rightarrow \text{oca}$$

2.5. Evaluación de la similitud de textos.

En el ámbito del procesamiento del lenguaje natural, un aspecto fundamental es realizar un análisis sobre las diferencias que existen entre las frases reconocidas por el modelo, y las que realmente se encuentran en él. Con el análisis, lo que se persigue es identificar similitudes y discrepancias entre los textos de referencia y los de salida del programa, contando diversas aplicaciones como la detección de plagio o la clasificación de documentos.

2.5.1. Similitud del coseno o *Cosine similarity*

La similitud del coseno mide el angulo que generan dos vectores en un espacio vectorial ignorando sus magnitudes, obteniendo de esta manera un intervalo entre -1 y 1, que indica cuanto de similar son dos textos. Si el resultado es -1 o cercano él, denota que los textos son totalmente distintos; mientras que si obtenemos un valor de 1, indica que son idénticos.

En este caso, las palabras se representarán de forma vectorial, donde cada dimensión corresponde a un termino, y su valor representa la frecuencia de dicho término en el documento. El conjunto de documentos definirá un espacio vectorial, y la similitud se valúa en función del ángulo que forman los vectores en el espacio. Si son cercanos a cero, los vectores son casi

paralelos, lo que sugiere una alta similitud; por el contrario, si los ángulos son cercanos a 90 grados, estos serán ortogonales, lo que determinaría que son diferentes. [76]

$$\text{similitud}(a, b) = \cos(\theta) = \frac{a \cdot b}{||a||||b||} = \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2}\sqrt{\sum_{i=1}^{n} b_i^2}} \tag{2.6}$$

1. *El valor de similitud es igual a -1.* Este primer caso, es el peor debido a que nos indica que los ángulos son totalmente diferentes, por lo que estaríamos analizando dos textos que no tiene correlación entre ambos.

2. *El valor de similitud es igual a 0.* Estaríamos en el caso general, el cual, significa que ambos poseen cierta similitud. Si lo trasladamos a nuestro caso de estudio, este resultado sugiere que los textos comparte algunos caracteres en las mismas posiciones de las palabras, aunque presentan ciertas diferencias significativas.

3. *El valor de similitud es igual a 1.* Estaríamos ante el caso ideal, donde ambos vectores dibujan el mismo ángulo, y por ende, estaríamos ante la misma representación del texto.

CountVectorizer

La clase que CountVectorizer es una herramienta que forma parte de la librería de *Scikit-learn* de *Python*, cuya función principal consiste en transformar un texto en una matriz que represente las palabras y la frecuencia con la que aparecen en ese texto. [23, 42]

En este contexto, el término de "*documento*" no se refiere exclusivamente a un archivo de la forma que lo conocemos propiamente como: *"Diploma, carta, relación u otro escrito que ilustra acerca de algún hecho, principalmente de los históricos."* [10], sino que, mas bien, hace referencia a un conjunto de frases o palabras que forman una unidad dentro del corpus y que son independientes unas de otras.

Lo primero que se realiza es un proceso de *tokenización* mediante el cual se dividen el texto en unidades mínimas que denominamos como *tokens*. Es decir, en el caso de las matrículas de las embarcaciones, cada numero y cada carácter se consideraran un *token*, aunque para nosotros esos elementos posean un significado propio, en la matriz, adoptarán la forma de un elemento abstracto. En este proceso, y para evitar que se manejen excepciones por el formato de las palabras, se recomienda, normalizarlas convirtiéndolas a todas al mismo formato, por ejemplo, minúsculas.

El siguiente paso, consiste en la creación de una matriz dispersa donde las columnas representan las palabras únicas (*tokens*) del corpus, y las filas los documentos que lo conforman. Los valores de las diferentes celdas se corresponden a la frecuencia con la que un *token* aparece en cada documento especifico.

TF-IDF Vectorizer

Al igual que la herramienta anteriormente comentada, TF-IDF Vectorizer *(Term Frequency-Inverse Document Frequency Vectorizer*), o en español, *Frecuencia de Términos - Inversa de la Frecuencia del Documento* es una clase de la librería *Scikit-learn* de *Python*, la cual nos indica lo relevancia de una palabra en función de su frecuencia de repetición en un documento.

El valor de TF-IDF irá aumentado acorde se aumente la frecuencia en la que aparecen los término, aunque, de igual manera, disminuirá a medida que esa palabra se repita de forma asidua en el resto de los documentos que forman parte del corpus o colección. La fórmula que define el TF-IDF es la siguiente:

$$W_{ij} = t_{fij} * \log\left(\frac{N}{d_{fi}}\right) \tag{2.7}$$

En ella, definimos t_{fij} como la frecuencia de aparición de i en el documento j, dividido entre el número total de palabras en el documento j. De igual manera, N, es el número total de documentos en el corpus. Finalmente, d_{fi} es el número de documentos del corpus o colección que contienen la palabra i.

El proceso de trabajo seguido por esta clase, se diferencia del *CountVectorizer* en que no solo cuenta la frecuencia de las palabras, sino que ademas, les asigna un peso en función de lo relevante que ésta sea. Por ejemplo, en el caso de que existan términos técnicos, éstos tendrán mayor peso en comparación con palabras comunes, puesto que estos últimos, suelen repetirse con frecuencia.

Tras calcular t_{fij}, se aplica la fórmula mencionada para obtener el peso final de cada palabra. Este resultado indica la importancia de ese término en ese documento especifico en comparación con otros términos y documentos del corpus.

Capítulo 3

Metodología de Trabajo

3.1. Scrum

SCRUM [62] es un marco de gestión de proyectos de metodología ágil caracterizado por la entrega de forma continua, iterativa y en cortos periodos de tiempo con una duración máxima de 4 semanas denominados *sprints*. Al final de cada uno de ellos, se realizará una entrega de una versión preliminar del proyecto para su posterior valoración, con el objetivo de obtener una idea del punto en el que se encuentra el proyecto y poder reevaluar y reordenar las prioridades y los objetivos definidos al comienzo del desarrollo, aunque, también puede darse la situación que aparezcan tareas nuevas, que deben ser analizadas e introducidas en la pila de producto. Este proceso se realiza de manera repetitiva hasta que se procede a la entrega del producto final.

La metodología SCRUM también destaca por hacer uso de tableros para una organización mas visual, denominados KANBAN [3], en los cuales cada una de las columnas que lo formen definen, por un lado, el estado en el que se encuentran las tareas , siendo el mas básico el formado por las columnas: Pendiente, En progreso y Terminado. Por otro lado, también se organizan en función de la prioridad, es decir, se debe comenzar por aquel trabajo que permita entregar al finalizar el *sprint* un Producto Mínimo Viable (MVP) [71]. Dicho de otra manera, se comienza a trabajar para conseguir una primera versión del producto que satisfaga las necesidades básicas del cliente.

Esta metodología además de ser ampliamente utilizada en el sector gracias a los numerosos beneficios y ventajas que ofrece la compatibilidad entre ambas metodologías de trabajo, también es utilizada por la empresa QAISC, por lo que se ha optado por aplicarla en el desarrollo de este Trabajo Fin de Título (TFT). Durante el desarrollo del mismo, se realizaron reuniones semanales, al igual que, se dividió el trabajo en tareas asignándoles una prioridad y tiempo de ejecución aproximado, las cuales se revisaban y reevaluaban a medida que se alcanzaban los objetivos a corto plazo definidos al comienzo. Todo este proceso se realizaba haciendo uso de la herramienta de gestión Trello, tal y como podemos observar en la figura

3.1 [69]

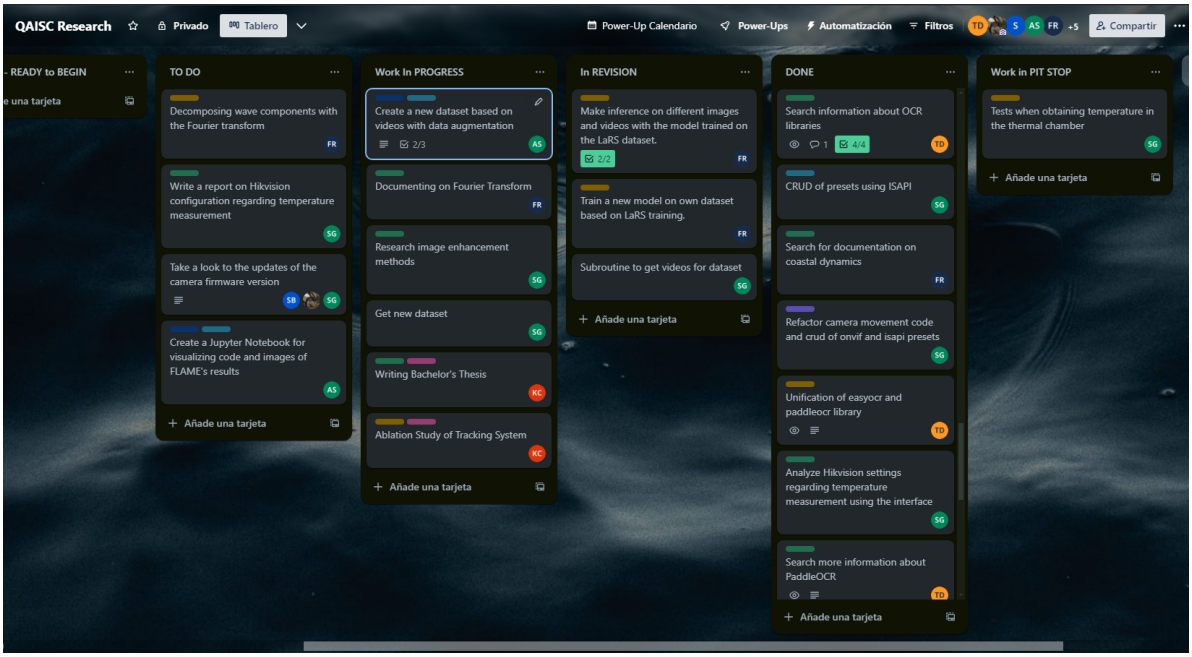


Ilustración 3.1: Ejemplo de tablón de Trello. Fuente propia.

3.2. Tecnologías empleadas

En esta sección se procederá a realizar un corto análisis sobre el lenguaje de programación por excelencia utilizado en este trabajo, así como todas aquellas herramientas y modelos, utilizados y/o estudiados para la realización, tanto de pruebas que permitan la obtención de resultados tras una comparación de diferentes estrategias, como para el desarrollo del propio trabajo.

3.2.1. Python

Python es un lenguaje de programación de alto nivel y orientado a objetos ampliamente utilizado en la industria de la ciencia de datos y la inteligencia artificial, aunque también lo podemos encontrar en el desarrollo de aplicaciones y web. [18].

Centrándonos en el ámbito del proyecto, *python* esta formado por una serie de librerías como *TensorFlow* o *PyTorch*, las cuales ofrecen funciones y herramientas especificas para el tratamiento de imágenes, y desarrollo de algoritmos para el aprendizaje profundo y la visión por computador. A continuación, se listarán las librerías principales utilizadas en este trabajo [16].

OpenCV.

OpenCV (Open Computer Vision Library) es una librería la cual nos proporciona facilidades para el procesamiento de imágenes. Ofrece funcionalidades tales como, detección de

objetos, detección de bordes, edición de parámetros varios como el color, tamaño...[52].

Pillow.

Pillow, al igual que *OpenCV* es una biblioteca especializada en el tratamiento y almacenamiento de imágenes. Sus herramientas nos permiten realizar modificaciones como manipulación de píxeles, conversiones de color u operaciones mas básicas como recorte y reescalados. [58].

NumPy.

Numpy es una librería especializada en el campo del procesamiento de la programación científica, es decir, es ampliamente utilizada para la realización de cálculos matemáticos y análisis de datos. Asimismo, otras librerías como las citadas anteriormente (*TensorFlow* o *PyTorch*), comúnmente utilizadas en el campo de la IA y el *deep learning* hacen uso de esta para realizar algunos de sus cálculos para la manipulación de los datos. [49].

Tensorflow.

Plataforma de código abierto para *Machine learning* , teniendo un papel fundamental en el desarrollo de métodos y modelos que se apliquen en el aprendizaje profundo y la inteligencia artificial. Su uso se encuentra bastante extendido entre la comunidad científica debido a su facilidad de uso y las múltiples configuraciones que le permiten a los usuarios, a elección entre ejecución con Unidad de Procesamiento de Gráficos (GPU), Unidad Central de Procesamiento (CPU) o, incluso con Unidades de procesamiento de Tensores (TPU) [66].

PyTorch.

Biblioteca de aprendizaje profundo de código abierto, que al igual que *Tensorflow*, su uso esta bastante extendido debido a su capacidad de realizar cálculos utilizando tensores, además de ofrecer bibliotecas con modelos preconfigurados y preentrenados. También soporta el uso de GPU maximizando el rendimiento en el entrenamiento de la redes neuronales utilizadas en el desarrollo de este trabajo.

Selenium.

Conjunto de herramientas diseñadas para automatizar los diversos procesos de las diferentes paginas webs. En este caso el uso que se le brindó a esta utilidad fue la utilización de Selenium Webdriver para la realización de automatizar el proceso de descarga de imágenes e información de las paginas web especializadas en barcos, como si lo estuviera realizando un humano, para la creación del *dataset*. A este proceso se le conoce como *web scrapping*.

3.2.2. CVAT

Computer Vision Annotation Tool (CVAT) es una herramienta cuyo objetivo principal consiste en facilitar el proceso de etiquetado de imágenes y vídeos que serán posteriormente, utilizados para el entrenamiento de modelos de detección de objetos o segmentación semántica. Debido a su interfaz de usuario intuitiva, permite al usuario manejar grandes cantidades de imágenes de manera cómoda, destacando la posibilidad de preetiquetar varias de estas de forma automática, así como la anotación en grupo, permitiendo la colaboración entre diversos miembros del proyecto. [8]

3.2.3. API para la detección de las embarcaciones

Para poder realizar de forma correcta la detección de los textos en las embarcaciones, en primer lugar se debe de haber realizado una detección de embarcaciones lo suficientemente precisa para cuando se aplique el proceso de *OCR*, este no lo realice en objetos o entornos que no son relevantes para el estudio realizado.

Es por ese motivo, por el cual, se ha hecho uso de una *API* propia de la empresa *Qaisc S.A*, cuya función consiste en detectar todas aquellas embarcaciones que se encuentren en la imagen que se la introducido como entrada. Como salida de la ejecución de ésta, tendremos un archivo *json* formado por un conjunto clave/valor, con el tipo de embarcación, confianza de ese resultado, el tamaño original de la imagen, y las coordenadas de la ubicación de ese barco en la imagen, es decir, nos devuelve cuatro puntos, que representan las cuatros esquinas del rectángulo delimitador contenedor del objeto. Un ejemplo de un fichero de salida que obtenemos tras ejecutar esta *API*, lo podemos observar en la figura 3.2.


```
{"model_name": "yolov7-boats",
    "data": [
        {"class": "boat", "model_class": 0, "top_left_x": 1278, "top_left_y": 342, "bottom_right_x": 1341, "bottom_right_y": 357, "confidence": 60.7421875},
        {"class": "boat", "model_class": 0, "top_left_x": 238, "top_left_y": 517, "bottom_right_x": 2094, "bottom_right_y": 1250, "confidence": 75.48828125},
        {"class": "boat", "model_class": 0, "top_left_x": 1148, "top_left_y": 312, "bottom_right_x": 1227, "bottom_right_y": 332, "confidence": 84.814453125},
        {"class": "boat", "model_class": 0, "top_left_x": 2087, "top_left_y": 395, "bottom_right_x": 2196, "bottom_right_y": 450, "confidence": 86.5234375},
        {"class": "boat", "model_class": 0, "top_left_x": 1960, "top_left_y": 290, "bottom_right_x": 2079, "bottom_right_y": 321, "confidence": 87.060546875},
        {"class": "boat", "model_class": 0, "top_left_x": 1744, "top_left_y": 332, "bottom_right_x": 1862, "bottom_right_y": 358, "confidence": 87.060546875},
        {"class": "boat", "model_class": 0, "top_left_x": 2077, "top_left_y": 250, "bottom_right_x": 2150, "bottom_right_y": 274, "confidence": 87.255859375},
        {"class": "boat", "model_class": 0, "top_left_x": 1564, "top_left_y": 302, "bottom_right_x": 1700, "bottom_right_y": 328, "confidence": 88.0859375},
        {"class": "boat", "model_class": 0, "top_left_x": 1979, "top_left_y": 245, "bottom_right_x": 2084, "bottom_right_y": 280, "confidence": 90.380859375}],
    "original_height": 1522, "original_width": 2234
}
```


Ilustración 3.2: Ejemplo de un fichero de salida tras ejecutar la *API*. Fuente propia.

Capítulo 4

Entorno de experimentación y desarrollo.

En esta sección se procederá a desarrollar y explicar el procedimiento seguido de manera detallada de cada una de las partes que conforman el trabajo final, mediante el uso de las métricas que evalúen su rendimiento que se explicaron en el apartado 2.4.

4.1. Creación del *dataset*.

En primer lugar y con el objetivo de poder obtener un entorno experimental que simule de manera realista las condiciones a las que nos enfrentaremos al poner en producción este trabajo, se debía construir una colección representativa de escenarios. Este conjunto de datos, debían abundar los elementos claves a analizar, tales como eran las matriculas, nombres y demás medios de identificación citados en el apartado 2.1. Por ello, se creó un primer *dataset* que evolucionaba a medida que se avanzaba en las pruebas y se iba identificando y comprendiendo los errores que se obtenían, además de las mejoras que se le aplicaba al propio código con el fin de subsanar aquellos pequeños detalles que influían de alguna manera en el resultado.

La versión inicial estaba compuesta por 10 imágenes que, posteriormente, se aumentó a 21. Estas fueron capturadas mediante las cámaras PTZ que se encuentran situadas tanto en Anfi del Mar, como en el puerto de Taliarte, ambos puntos localizados en la isla de Gran Canaria.

En la imagen superior podemos observar esta primera muestra, la cual esta formada por imágenes que enfocan al mismo punto, pero con diferentes niveles de *zooms* de cámara y condiciones lumínicas, aportando de esta manera mayor valor y calidad al conjunto. Esto es así ya que uno de los factores externos al que se le debe hacer frente son los posibles reflejos de los rayos del sol incidiendo sobre el casco del barco, que pueden acabar generando posibles sombras, o incluso llegando a imposibilitar completamente la detección de caracteres. Otro

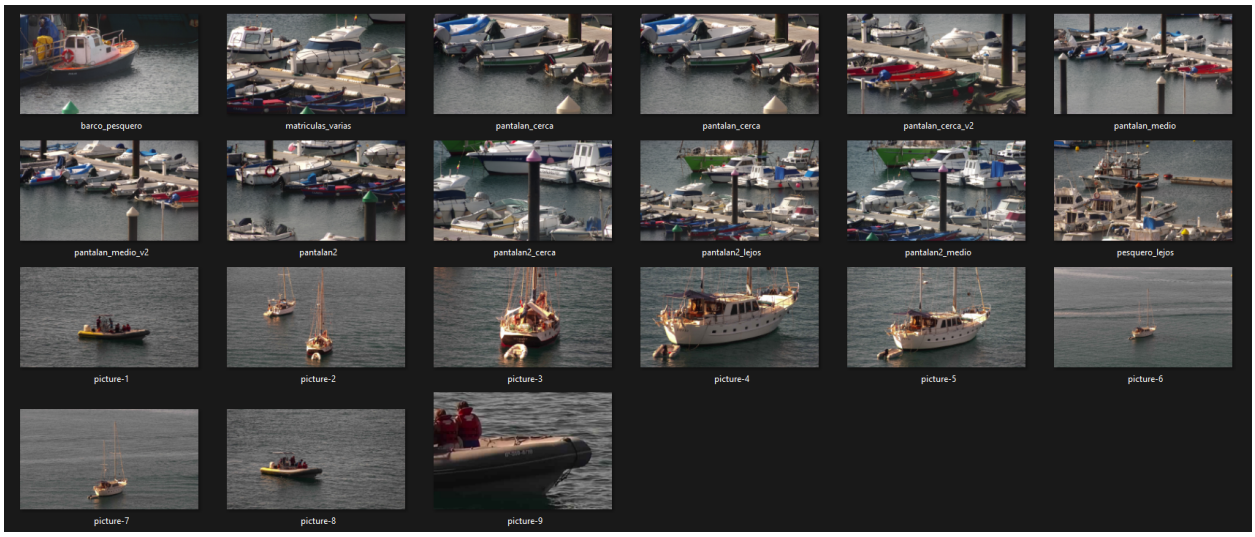


Ilustración 4.1: Primer *dataset* con imágenes de Taliarte y Anfi del Mar. Fuente propia.

de los factores importantes es la distancia que existe entre la cámara que captura la imagen y la embarcación, aunque ésta puede ser ajustada mediante la aplicación de *zoom* digital. Sin embargo, esta correccional implica un compromiso mayor: cuanto más ajuste se aplique, mayor número de pixeles visibles, pudiendo reducir de esta manera la calidad de la imagen.

Es por ello, que se decidió ampliar el número de muestras, y de igual manera, diversificar los casos para incluir imágenes obtenidas en diferentes momentos del día, como puede ser por la noche, amanecer o atardecer, además de fotos con embarcaciones ubicadas a distintas distancias y de diversos tamaños. Con este aumento se buscaba obtener una visión más realista de los puntos fuertes y las áreas de mejora.

Para poder realizar esta expansión del *dataset*, se desarrolla un *scrapper* [7], haciendo uso de la utilidad ***Selenium***. Esto permitió automatizar la descarga de imágenes desde una pagina web especializada en imágenes de embarcaciones, como es ***ShipSpotting***, que ofrece una amplía cantidad de filtros de búsqueda, ademas de poder obtener información sobre cada una de éstas, como la calidad de la imagen y el punto en el que fue capturada. Tras finalizar la descarga, éstas pasan por una segunda etapa de filtrado en las que se eliminan aquellas en las que no existía ningún texto visible, o en cuanto a la información, todo dato que no sea estrictamente necesario para la creación del fichero *ground truth* [8], que se utilizará en las posteriores fases.

En la figura 4.2, podemos ver una pequeña muestra correspondiente a las 300 imágenes que fueron finalmente seleccionadas como conjunto de datos. éste se utilizó para realizar diferentes pruebas que incluyeron técnicas de tratamiento de imágenes, tales como binarización, énfasis de contornos, o reescalado aplicando diversos métodos de interpolación. Todas las imagenes seleccionadas cubre, como mínimo, uno de los casos necesarios definidos previamente.

Asimismo, se llevo a cabo una clasificación exhaustiva de la totalidad de las imágenes

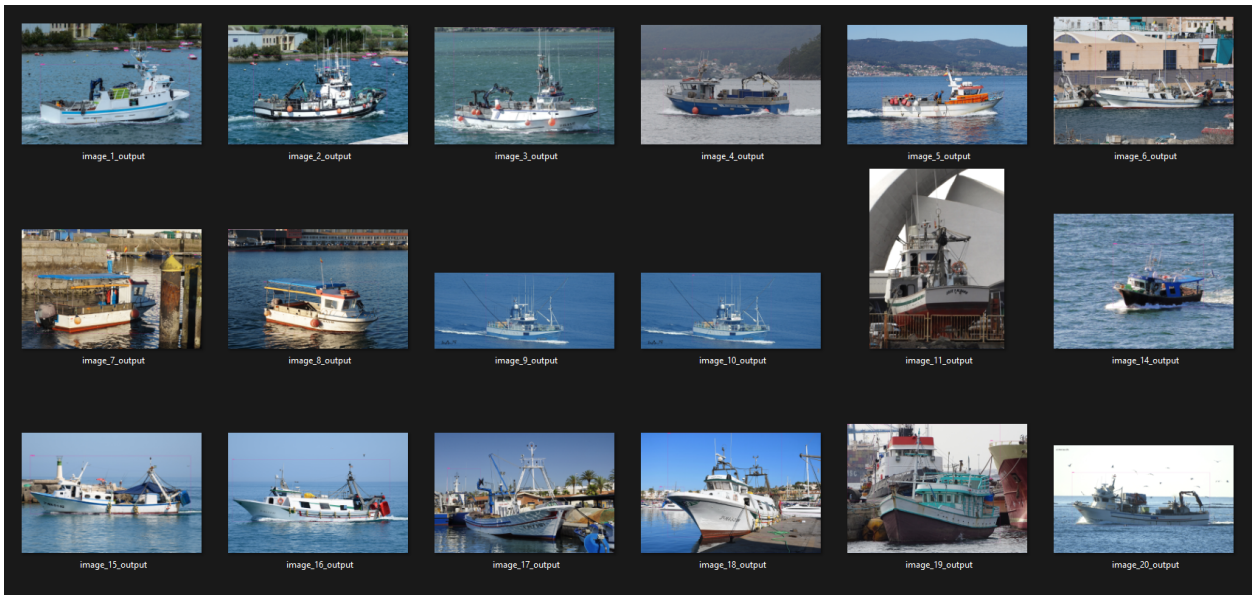


Ilustración 4.2: Pequeña muestra del segundo *dataset*. Fuente: ShipSpotting.

con el objetivo de poder conformar una muestra lo suficientemente precisa como para poder determinar con exactitud cuales son las situaciones en las que los modelos obtienen mejores resultado, así como aquellas situaciones más complejas, como puede ser poca visibilidad o baja calidad de las imágenes impactan de manera negativa en su rendimiento.

En esta clasificación se optó por dividir el *dataset* en estos seis tipos:

- **Cargueros**. Embarcaciones dedicadas al transporte de mercancías mediante el uso de contenedores. Este tipo nos aportan medios de identificación diferentes al resto de tipos de embarcaciones de la clasificación, como son las nacionalidades o el *IMO* de los buques.

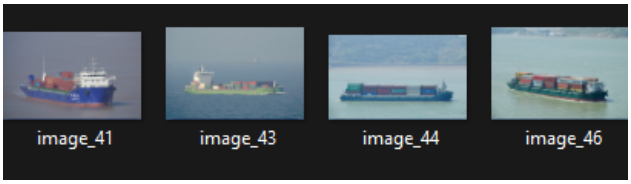


Ilustración 4.3: Muestra de ejemplo para el tipo cargueros

- **Condiciones adversas**. Imágenes que posean una baja calidad a nivel de píxeles. También podemos encontrar muestras en las que la pintura del barco se encuentre caída imposibilitando la correcta lectura de la matrícula. De la misma manera, hay imágenes en las que las condiciones meteorológicas y de oleajeque aumentan la complejidad del proceso de OCR.

- **Distancia larga**. Ejemplos de aquellos barcos que están lo suficientemente lejos como para requerir realizar algún tipo de ampliación sobre la imagen debido a que no se observa ningún escrito sobre ella si no se realiza esta acción.

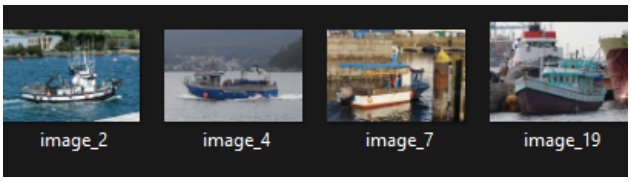


Ilustración 4.4: Muestra de ejemplo para el tipo condiciones adversas

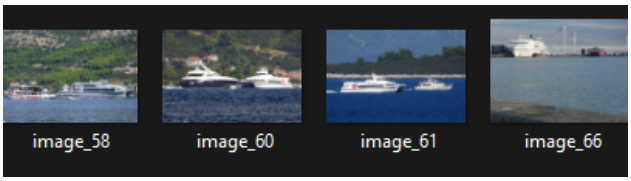


Ilustración 4.5: Muestra de ejemplo para el tipo distancia larga

- **Distancia corta**. Muestras en las que las embarcaciones se encuentran o bien amarrado en el pantalán [9] , o lo suficientemente cerca de las cámaras como para no tener que realizar ningún tipo de *zoom*. El texto, en este caso, es fácilmente legible sin necesidad de ajustes.

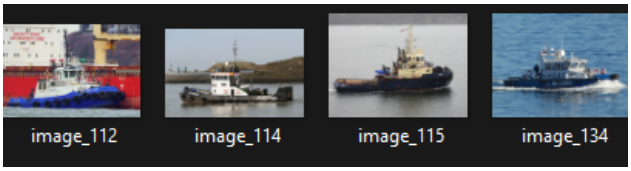


Ilustración 4.6: Muestra de ejemplo para el tipo distancia corta

- **Barcos grandes**. Hace referencia a aquellos barcos que superen los 21 metros de eslora, pudiendo observarse embarcaciones de pesca profesional, de recreo, así como embarcaciones de salvamento marítimo.
- **Barcos pequeños**. En este caso, son imágenes de las embarcaciones cuya eslora es inferior a los 21 metros. Aquí nos encontraremos en su mayoría barcos de recreo personal, lanchas hinchables o barcos de vela ligera.
- **Noche**. Imágenes extraídas después del atardecer, en la condiciones de oscuridad, donde la iluminacion proviene principalmente de los propios focos de los barcos.

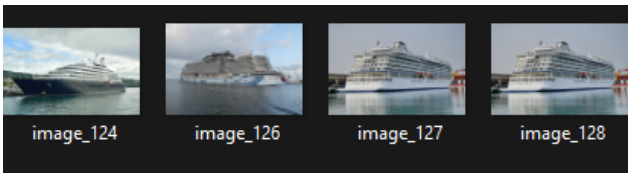


Ilustración 4.7: Muestra de ejemplo para el tipo barcos grandes

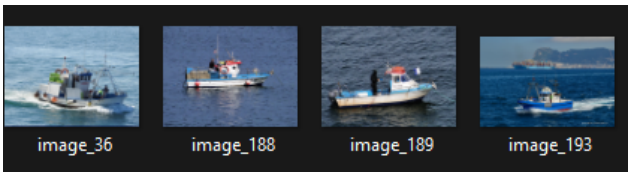


Ilustración 4.8: Muestra de ejemplo para el tipo barcos pequeños

A la totalidad de las imágenes se les debe aplicar el proceso de etiquetado haciendo uso de la herramienta CVAT. Este nos genera un fichero en formato *json*, que incluye las coordenadas de las regiones que hemos marcado como de interés durante el proceso de etiquetado, y que se añadirán al fichero *ground truth* anteriormente comentado.

El proceso que se sigue con esta herramienta es bastante sencillo, debido a que la utilidad de anotación de imágenes nos permite dividir la totalidad del conjunto en lo que ellos denominan *tasks*, que a su vez, en el caso de que poseyéramos un volumen inmanejable de ilustraciones, se podría separar en pequeños subconjuntos, tal y como podemos ver en la imagen inferior.

4.2. Fase de detección.

Una vez creado un *dataset* amplio que permite evaluar el modelo en diversas situaciones como las comentadas en el punto anterior, se procede a generar la primera parte de las fases que componen un sistema de detección y reconocimiento de textos. En ella, debemos tener en cuenta ciertos desafíos y/o limitaciones que se nos presentan, que son de índole externa, y que, por ende, podrían acarrear una mayor tasa de errores.

Uno de los principales es la distancia a la que se encuentra la embarcación de la cámara, generando de esta manera unas imágenes donde el texto posee un tamaño minúsculo que debe pasar previamente por un proceso de reescalado con el fin de aumentar su tamaño sin perder el formato de forma, ya que ésto generaría la distorsión del texto. Asimismo,

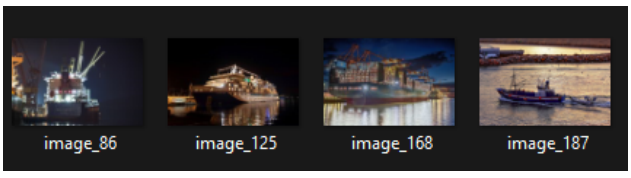


Ilustración 4.9: Muestra de ejemplo para el tipo noche

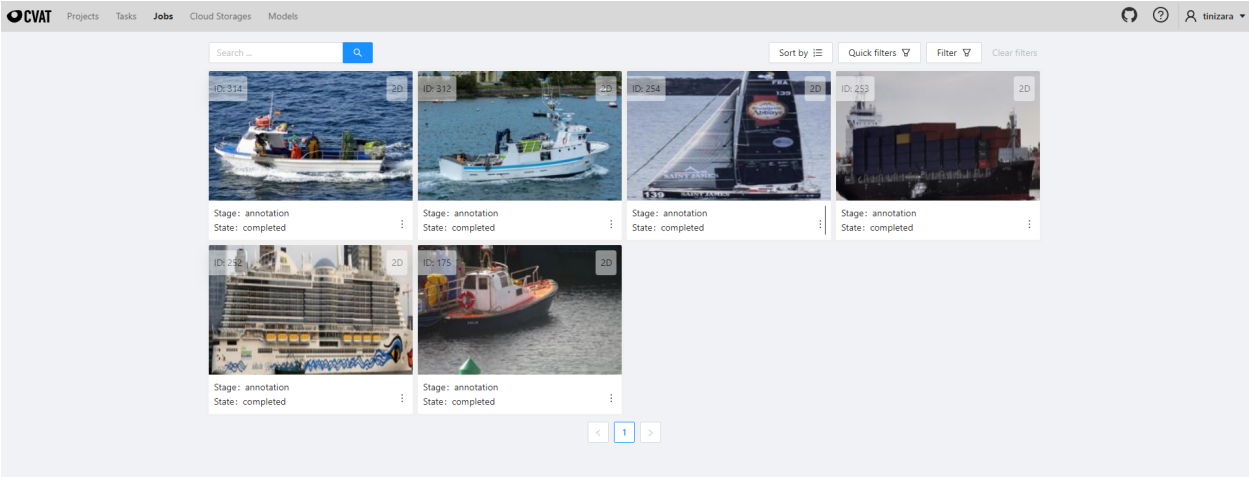


Ilustración 4.10: Ejemplo del apartado *jobs* del CVAT.

para suplir este procesamiento de la imagen, se puede optar por realizar cierto *zoom* a la embarcación, con el contrapunto de perder calidad de la imagen si este acercamiento digital es significativo, ya que podría generar la aparición de ruido y el aumento del tamaño de los *pixeles* visibles.

De igual manera, otro de los puntos importantes y que causan problemas en la detección es el solapamiento que existe entre las embarcaciones en los diferentes amarres de los puertos, al momento de obtener la imagen ya que de manera general, en los pantalanes, estas se encuentran a poca distancia la una de la otra tapándose así, unos barcos a otros, sin tener en cuenta así vez, los propios elementos del puerto como focos, o entradas de agua que puedan ocasionarlo de igual manera.

El estado de la pintura de la embarcación, y la fuente y materiales utilizados para escribir sobre el casco, el nombre, matricula y cualquier otro texto identificativo, también posee uno de los papeles fundamentales en esta fase. Esto es así debido a que, en cuanto peor estado se encuentre estéticamente (bien sea pintura desgastada o caída, como caligrafía con exceso de florituras), mayor dificultad a su lectura.

Por último, unas condiciones meteorologías adversas como pueden ser, en Canarias, los días de mucha calima, o aquellos con mareas y fuertes oleajes, como las conocidas popularmente como ”mareas vivas”, también afectan de manera significativa debido a que el choque de las olas con las embarcaciones generan la espuma que salta y puede ocultar de alguna manera los textos. Asimismo, la propia superficie del mar y la reflexión de los rayos del sol sobre el casco de las embarcaciones pueden generar brillos y reflejos que dificultan la visibilidad de los caracteres, o incluso, distorsione ante la cámara las formas de las letras haciendo que algunas partes de los textos sean prácticamente invisibles al sistema.

Tras entender el contexto al que hay que hacer frente, se debe diferenciar la manera de ejecución de la fase de detección que realizan los modelos de una sola fase (modelo *end-to-end*, y los modelos en dos fases.

Los modelos de una sola fase, funcionan de manera que reciben como entrada la imagen que captura la cámara directamente, y tras ello, la imagen se transforma a un mapa de píxeles, donde cada *pixel* posee un valor que representa el color de la región en un intervalo entre 0 y 255 para las imágenes de color, o si es en blanco y negro, solo poseerán dos valores, 0 para representa el negro y 255, para el blanco. Posteriormente, este mapa se transformará a un tensor que tiene tres dimensiones (altura, anchura y canal del color), y será la entrada de la red neuronal convolucional encargada de extraer las características relevantes para los procesos posteriores. A continuación, y basándose en estas características, tanto *kerasOCR* como *easyOCR*, además del resto de modelos que unifican la detección y el reconocimiento de caracteres, se generarán directamente las posibles *bounding boxes* o cajas delimitadoras, que encerraran en su interior todo aquello que su forma represente de manera similar la forma de las letras del abecedario con las que ha sido entrenado el modelo respectivo.

En cambio, en los modelos de dos etapas como *MixNet* en su configuración más avanzada, inician la ejecución de la misma manera, es decir, la imagen que se obtiene de las cámaras es procesada como un mapa de píxeles que a su vez se transforma en el tensor que funcionara como conjunto de entrada para la red. En cambio, donde reside la principal diferencia es en el segundo paso, incluso entre los propios modelos de dos fases. Mientras *paddleOCR* en esta fase hace uso de *DBNet* que funciona como un mapa de calor con una red convolucional para detectar los bordes y contornos que puedan representar caracteres. Tras obtener esta primera salida, se procesan de manera que se recortan las regiones de interés que se poseen hasta el momento, para, a continuación, utilizarlos como entrada para un segundo bloque de redes *DB* en conjunto con las *CNNs*, donde se repetirá este proceso, refinando de esta manera los resultados.

Este proceso lo podemos observar en la imagen que encontramos a continuación, que resume de manera visual lo comentado anteriormente.

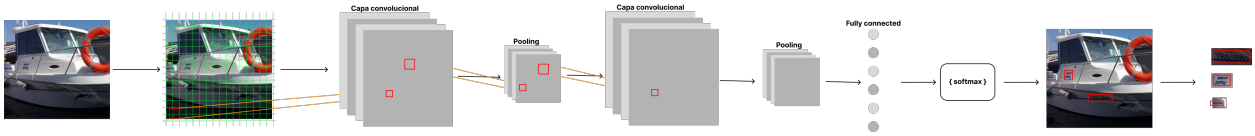


Ilustración 4.11: Diagrama de flujo para la fase de detección de *PaddleOCR*. Elaboración propia.

Por otro lado, *MixNet*, esta formado por dos módulos, donde en primer lugar, hace uso de una red *FSNet* para la extracción de las características de la imagen original a diferentes escalas, es decir, de textos con un tamaño de fuente tanto grande y pequeño. De esta manera, obtenemos diferentes perspectivas, que posteriormente se unifican en una sola imagen, para, finalmente, transformarlo en el tensor correspondiente. Seguidamente, la salida de este proceso, pasa como entrada a un nuevo módulo denominado *CTBlock*, cuyas función consiste, aplicando transformadores, realizar el refinamiento de los contornos del texto, que en el caso de *paddleOCR*, se realizaba pasando la salida por diferentes bloques de redes *CNN*, decrementando la velocidad de ejecución del sistema.

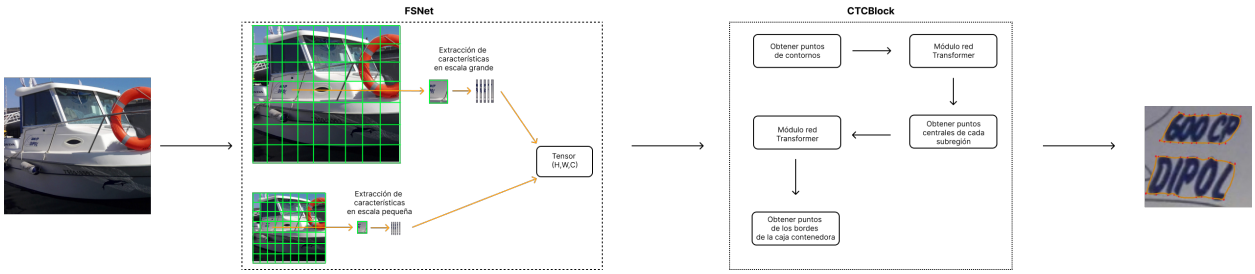


Ilustración 4.12: Diagrama de flujo para la fase de detección de *Mixnet*. Elaboración propia.

Finalmente en ambos tipos de modelos, si el recorte que vamos a utilizar como entrada para la etapa de reconocimiento, no se encuentra en escala de grises, es recomendable aplicar esta transformación. Esto ayudará a la red a detectar de una forma más clara los contornos que representan las formas de los caracteres, lo que de manera teórica se verá reflejado en una mayor tasa de acierto, aunque, como bien hemos comentado, es importante tener en cuenta el contexto en el que se ha obtenido la imagen original.

4.3. Fase de reconocimiento.

Tras la ejecución de la fase anteriormente comentada, se procede a ejecutar la fase de reconocimiento, donde, teniendo en cuenta el entorno en el que el sistema debe funcionar, también existe una serie de detalles que de alguna manera generan cierta problemática.

La principal a la que se enfrenta el modelo a la hora de reconocer los caracteres, y más concretamente cuando tratamos con textos en castellano, es el uso de ciertos caracteres especiales, como son, por un lado, la existencia de la letra *ñ*, y por otro el indicador de ordinal ^{a}. Ambos caracteres poseen especial importancia en nuestro trabajo debido a que, mientras que el primero lo podemos encontrar como parte del nombre de algún barco, el segundo lo encontramos de manera general en la totalidad de las matriculas nacionales, tal y como se comentó en el apartado de identificación de los embarcaciones (apartado 4).

Esta dificultad que se nos presenta específicamente con el indicativo de ordinal, viene agravada en cuanto a que, aunque teóricamente para diferenciarlas se le añade debajo del carácter una linea horizontal, ésta, de manera general en las muestras que se han obtenido, no se encuentra. Como consecuencia de esto, se produce una confusión a la hora de transcribir los textos, de manera que si se encuentra ^{a}, el sistema puede determinar que se trata de una *a*, generando de esta manera una tasa de error que se arrastrará durante toda la ejecución.

Otro de los factores determinantes que afecta en el reconocimiento, es el tamaño de la imagen que le introducimos como entrada al modelo. Como se comentó en el punto anterior, a la imagen original, tras aplicarle la detección de la embarcación y se complementa la fase de detección al completo, tendremos como salida una imagen de baja resolución y de un tamaño lo suficientemente inferior al original, como para afectar de manera directa a los resultados. Es por ello que se debe realizar un reescalado a ese recorte intentando mantener

la proporción. En este caso se hace uso la interpolación definida como un proceso matemático mediante el cual, se calculan datos a partir de otros ya conocidos.

Previamente a trabajar con el *dataset* definitivo, se procedió a ejecutar una serie de pruebas sobre una pequeña muestra del conjunto con el objetivo de comprobar si existe alguna diferencia notable sobre la aplicación de algún u otro de los métodos de interpolación de imagen clásico. Esto es necesario debido a que los modelos de reconocimiento son sensibles al tamaño de la misma.

Los métodos empleados para ello fueron: interpolación lineal, cubica, de área, vecino más cercano y *lanczos4*. La principal diferencia es la velocidad de ejecución y la complejidad de cada uno, así como en su aplicación general, por ejemplo, en el caso de la interpolación lineal, su uso predominante es para imágenes pequeñas en las que se buscan realizar cambios rápidos, mientras que, *lanczos4*, se aplica para aquellas imágenes digitales que se desean ampliar con gran numero de detalles, aunque su tiempo de procesamiento es demasiado lento. En nuestro caso especifico, las diferencias entre ellos han resultado insignificantes por lo que se optó por utilizar la lineal.

En esta fase, se procederá a seguir con los modelos que se introdujeron en la fase de detección para ejemplificar el modo de ejecución de los dos tipos de modelos. Por ello, y comenzando con la ejecución de una sola etapa, *paddleOCR* persigue los siguientes pasos, comenzando con recibir como entrada las imágenes de cada región de interés que se obtuvieron como salida en la fase de detección. Tras ello, y como se comento previamente, los modelos son sensibles al tamaño de entrada, por lo que se asignó un tamaño estándar mínimo que debía tener cada recorte, y en el caso de que este no se cumpliera, se debía pasar por un proceso de reescalado, en el que manteniendo la proporción, se consiguiera aumentar el tamaño de ésta. Tras tener la imagen con el tamaño correcto, esta pasa por una red neuronal convolucional, que se encargará de extraer todas las características, a la vez que se procesan de manera secuencial a medida que se van obteniendo con el objetivo de mantener el orden lectura acorde se encuentre en el recorte. En este punto, se puede proceder de dos manera, bien haciendo uso de *RNN*, o bien haciendo uso de redes *transformer*, aunque en nuestro caso para poder realizar una comparación con el modelo de dos etapas, se dispuso a hacer uso de las *RNN*.

Las RNN que se utilizan, poseen una memoria que nos permite almacenar al corto plazo los vectores de característica que se obtiene como salida de las CNN, y son estos vectores lo que se tratan de manera secuencial manteniendo las dependencias que existen entre los caracteres que forman el texto. El siguiente paso, y el más importante es el proceso de transcribir las secuencias de características en una salida de texto propiamente, y para ello, se hace uso de la técnica denominada *Connectionist Temporal Classification*, *CTC* en adelante, la cual genera una secuencia de texto a la cual no tenemos que indicarle donde termina o comienza cada palabra, sino que es ella misma la que busca entre todas posibilidades, eliminando además aquellos caracteres que se hayan podido repetir, de que palabra o palabras

se corresponden con lo obtenido del paso anterior, generando de esta manera una salida. Por ultimo, en nuestro caso además, se procede a estructurar la salida obtenida en un fichero de texto, con el objetivo de que sea mas accesible para el usuario, donde no solo encontrará el texto, sino además a que imagen exacta pertenece y mas información sobre la evaluación de los resultados, como puede ser la tasa de acierto o la tasa de error por cada carácter traducido en función del fichero de *ground truth* que nos sirve para realizar estas comparaciones.

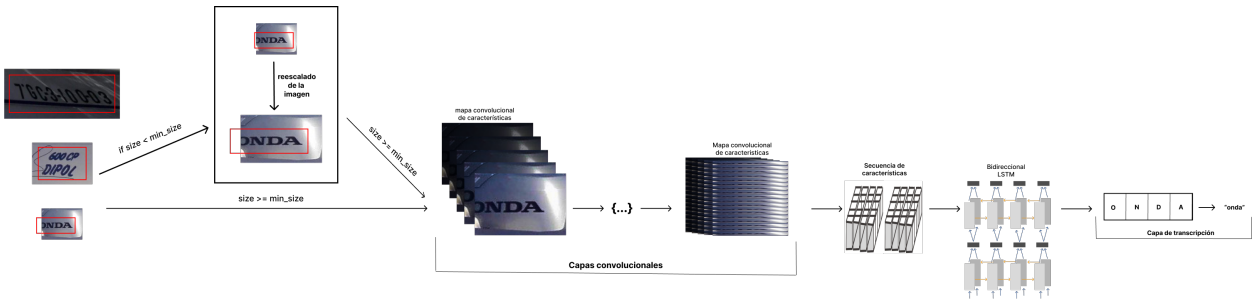


Ilustración 4.13: Diagrama de flujo para la fase de reconocimiento de *PaddleOCR*. Elaboración propia.

Por otro lado, nos encontramos con CLIP4STR, que forma parte de los modelos en dos etapas que combina por un lado, el enfoque *CLIP*, que permite relacionar tanto la información de los aspectos visuales como es la forma que tiene el texto, junto con una predicción de texto basada en características lingüistas. Para ello, primero se deben ejecutar una serie de pasos que comienzan con la entrada de la región recortada con el tamaño original. Como pasaba con *paddleOCR*, se debe reescalar a un tamaño mínimo de 224x224 píxeles, (aunque acepta imágenes de menor tamaño, se ha comprobado que se generan errores que interrumpen la normal ejecución del modelo)

Posteriormente esta imagen, se pasa a un *ViT*, también denominado *vision transformer*, cuya función es procesar cada imagen dividiéndola en múltiples regiones mas pequeñas, para extraer de cada una de éstas todas las características visuales que permitan identificar formas o contornos que representan la estructura que posee el texto en esa imagen. A medida que se procesan las subregiones, se combinan de manera que acabamos creando una representación global en forma de vector, y, a su vez, paralelamente, el texto candidato, que puede provenir de un diccionario o un módulo de predicción previa, se transforma en un vector numérico, denominado *embedding* haciendo uso de un red basada en *transfomer*. Finalmente, se procede a comparar los vectores que se han obtenido de procesar las imágenes, con el texto candidato que se encuentra almacenado en el vector numérico, evaluando la similitud de ambos, de manera que el texto cuyo vector se encuentre mas próximo al vector correspondiente a la subregión se determinará como la transcripción mas probable.

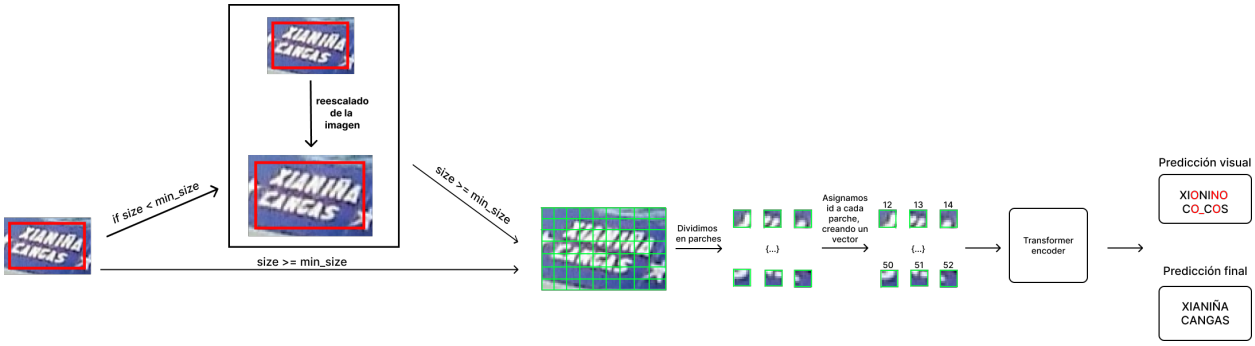


Ilustración 4.14: Diagrama de flujo para la fase de reconocimiento de *Clip4STR*. Elaboración propia.

Capítulo 5

Análisis de resultados

En esta sección, se procederá a mostrar, en primer lugar los resultados de manera general obtenidos de los diferentes modelos existentes para la detección y reconcomiendo de caracteres, que tradicionalmente se han utilizado en contextos donde, las imágenes o los ficheros de texto están en condiciones controladas. Estos modelos se han ejecutados en un nuevo contexto como es el marítimo, donde no habían sido aplicados previamente, por lo que estos primeros resultados proporcionarán una visión sobre las tasas de error por carácter y palabra, así como las tasas de acierto y precisión.

Posteriormente, se profundizará en los resultados específicos acorde a la clasificación establecida en el apartado 4.1, de creación del *dataset*. Para cada uno de los tipos de imágenes, se analizará el rendimiento en detalle de cada modelo.

Por último, también se mostraran estos resultados de manera visual, permitiendo comprender de esta manera el motivo de por que un modelo con unas determinadas imágenes, era capaz de ofrecer resultados excelentes, pero en otros, este rendimiento disminuye de manera considerable, aportando así una visión más completa de los factores y desafíos que se han comentado a lo largos del desarrollo del proyecto.

5.1. Resultados generales

En este apartado, se introduce la evaluación del desempeño demostrado por los diferentes modelos de detección y reconocimiento en función de las principales métricas existentes y que son claves en este contexto. En este trabajo, se ha optado por realizar estas pruebas con cinco modelos que teóricamente, en sus respectivos *papers* y ofrecían buenos resultados en entornos controlados y/o que presentan buenas características, con el objetivo de comprobar si se pueden llegar a utilizar en entornos mas variados como es el marítimo-portuario, y con que grado de confianza. Estos modelos son: PaddleOCR, EasyOCR, PyTesseract, KerasOCR y una combinación de MixNet con Clip4STR.

Todos los modelos se ejecutaron sobre el mismo *dataset* de manera independiente y siendo evaluado con la siguientes métricas ampliamente utilizadas en el OCR: tasa de error por carácter (CER)y la tasa de error de palabras (WER), las cuales representan el porcentaje de resultados incorrectos en términos de caracteres y de palabras respectivamente. Junto a la tasa de acierto, son las tres principales métricas y utilizadas con mayor frecuencia a la hora de analizar los resultados en la detección y reconocimiento. De igual manera, también se harán uso de métricas menos conocidas pero que también representan puntos importantes tales como la cantidad de información que se ha perdido en base a las de referencia (WIL), la proporción de palabras preservadas (WIP) y la tasa de error de coincidencia (MER), que representara la probabilidad de que una coincidencia entre el texto predecido y el de referencia sea incorrecta. Por ultimo, se aplicará la distancia de Levenshtein, nos ofrece la similitud entre los textos reconocidos y los de referencias, teniendo en cuenta que procesa todas aquellas transformaciones que se tendrían que realizar, bien sea, insertar, eliminar, o sustituir por otros caracteres, para a partir del texto obtenido, lograr alcanzar el texto de referencia.

Al analizar los resultados, se puede determinar que PaddleOCR, ofrece el mejor rendimiento en términos de métricas clave como son el CER, WER y tasa de acierto, obteniendo en esta última un resultado del 69 % sobre el conjunto de las imágenes variadas. Esto se debe a su capacidad de reconocer correctamente la mayoría de las palabras y/o textos en este entorno tan desafiante ya que incluye casos diversos como imágenes de barcos que se encuentran a una distancia considerable de la cámara, condiciones adversas como puede ser la meteorología o mala calidad de la imagen, entre otros. En contraposición, tenemos a EasyOCR y KerasOCR que tienen el peor desempeño con tasas de error tanto de caracteres como por palabras bastante altas. En cuanto a la tasa de acierto que presenta KeraOCR, ésta es la más baja de los cinco modelos, representado de esta manera que más de dos tercios de los caracteres que se reconocen son incorrectos. Por último, la combinación de Mixnet, que se utiliza para el proceso de detección y Clip4str, para el reconocimiento, ofrece un rendimiento competitivo en comparación con Paddleocr, con un potencial considerable y siendo más efectivo que los modelos de una sola etapa tradicionales.

	Paddleocr	Easyocr	Pytesseract	Kerasocr	MixNet+Clip4str
CER	30,26 %	60,35 %	57,75 %	69,07 %	41,10 %
WER	70,44 %	101,77 %	102,83 %	97,21 %	71,81 %
MER	67,15 %	85,80 %	69,79 %	81,83 %	71,82 %
WIL	69,96 %	87,44 %	70,58 %	83,12 %	71,81 %
WIP	30 %	12,46 %	3,16 %	15,63 %	28,18 %
Distancia levensthein	3,48	6,71	6,29	7,45	4,72
Tasa de acierto	69 %	40,30 %	43,56 %	31,34 %	57,4 %

Cuadro 5.1: Comparación de los modelos OCRs

5.2. Resultados específicos por tipo de imagen

A la vista de los resultados comentados en el apartado anterior donde no se realizó distinción de las imágenes que conformaban el *dataset* utilizado para ejecutar la inferencia, en este segundo análisis, se procedió a realizar de nuevo la prueba considerando la clasificación de las imágenes realizadas en el apartado 4.1. Esta nueva ejecución nos permitirá poder contextualizar los resultados obtenidos, ofreciendo así la posibilidad de comprender de una mejora manera, en que entornos y áreas, los modelos actuales de OCR, tienen un mejor rendimiento, cuales son los factores que generan un mayor impacto sobre los resultados, y además, se añade la capacidad de identificar ciertos patrones de comportamiento que se puedan dar en condiciones concretas como son las imágenes nocturnas, en condiciones adversas, o en función del tamaño y la distancia de captura.

En los siguientes puntos, se mostrarán los cuadros de resultados correspondientes a cada tipo de *dataset* y que resultados se obtienen de cada modelo, así como una pequeña muestra visual de cada uno de los tipos de imagen.

- ***Condiciones adversas.***
  El primer conjunto de imágenes sobre las que se ejecutan las inferencia, son las referentes a todas aquellas que dificulten de alguna manera, bien sea por factores externos como los rayos de sol incidiendo sobre el casco, condiciones meteorológicas difíciles, o por problemas con la lectura de los textos.

  Como se puede apreciar todos los modelos presentan grandes dificultades, que generan amplias tasas de error tanto en caracteres (CER), como en palabras (WER) y solo PaddleOCR y Pytesseract consiguen superar el 50 % de tasa de acierto; mientras que KerasOCR es el modelo que mayor dificultad posee a la hora de ser capaz de extraer caracteres.

  La siguiente tabla recoge una comparación de los recortes de referencia que forman el *ground truth* junto con los resultados obtenidos de cada uno de los modelos.

- ***Entorno nocturno.***
  Las imágenes que hacen referencia al entorno nocturno, poseen diferentes factores ex-

	Paddleocr	Easyocr	Pytesseract	Kerasocr	MixNet+Clip4str
CER	41,85 %	66 %	47,25 %	76,59 %	71,03 %
WER	82,18 %	99,77 %	83,96 %	99,22 %	93,75 %
MER	82,18 %	92,34 %	58,02 %	92,97 %	93,75 %
WIL	83,33 %	93,24 %	58,44 %	92,97 %	93,75 %
WIP	16,67 %	6,76 %	2,81 %	5,47 %	6,25 %
Distancia Levenshtein	4,55	7,11	4,9	7,66	7,5
Tasa de acierto	55,12 %	31,9 %	51,99 %	22,59 %	29,41 %

Cuadro 5.2: Resultados de la ejecución de los modelos con imágenes de tipo condiciones adversas.

Ground truth	PaddleOCR	EasyOCR	Pytesseract	KerasOCR	Mixnet+Clip4str
 Texto GT: *ARAW*	Sin resultados	Sin resultados	 Texto: -	 Texto: *AR*	Sin resultados

Cuadro 5.3: Comparación de modelos OCR en condiciones adversas

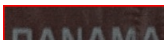
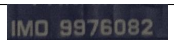
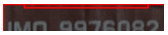
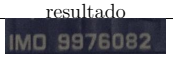
ternos determinantes que pueden afectar de manera considerable a los resultados que se obtienen de la ejecución. Estos son que no sólo la falta de luz hace que sea no sean visibles los textos, sino que, existen casos en los que los barcos pesqueros, poseen focos para cuando son las temporadas de pesca, puedan navegar de manera segura, y estos propios puntos de luz generan sombras y brillos sobre las bandas y casco, que a su vez puedan generar efectos ópticos de que hay una cierta letra que en realidad es otra similar.

Como se puede apreciar en el cuadro resultado, PaddleOCR es el que presenta la mayor precisión, con una gran diferencia sobre el resto de los modelos. Este rendimiento destaca sobre el resto de los modelos, que enfrentan mayores dificultades al procesar este tipo de imágenes, obteniendo una diferencia de un 20 % sobre el segundo que mejores resultados tiene.

	Paddleocr	Easyocr	Pytesseract	Kerasocr	MixNet+Clip4str
CER	10.07 %	64.00 %	46.37 %	59.23 %	24,06 %
WER	30.00 %	87.50 %	71.43 %	83.33 %	75 %
MER	30.00 %	87.50 %	57.14 %	72.22 %	75 %
WIL	35.00 %	93.75 %	57.14 %	83.33 %	75 %
WIP	65.00 %	6.25 %	0.00 %	16.67 %	25 %
Distancia levensthein	1.00	6.50	5.71	6.33	3
Tasa de acierto	91.38 %	46.94 %	51.85 %	50.00 %	78.57 %

Cuadro 5.4: Resultados de la ejecución de los modelos con imágenes de tipo nocturnas.

A continuación, en el cuadro inferior, se presentan los *crops* del *ground truth* comparados con los resultados de detección y transcripción proporcionados por los modelos.

Ground truth	PaddleOCR	EasyOCR	Pytesseract	KerasOCR	Mixnet+Clip4str
 Texto: *GREEN AANEKOSKI*	 Texto: *GREEN AANEKOSKI*	 Texto: *4Cfni AAnEvoCVi*	 Texto: Sin resultado	 Texto: Sin resultado	 Texto: *GREEN*  Texto: *AANEKOSKI*
 Texto: *PANAMA*	 Texto: *PANAMA*	 Texto: *G Gca*	 Texto: Sin resultado	 Texto: Sin resultado	Sin resultados
 Texto: *IMO 9976082*	 Texto: *IMO 9976082*	 Texto: *Cs*	 Texto: Sin resultado	 Texto: *9978082 imo*	 Texto: *9978082*

Cuadro 5.5: Comparación de modelos OCR en entorno nocturno

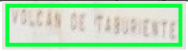
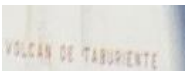
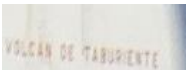
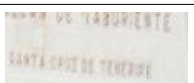
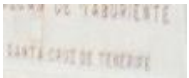
- ***Barcos grandes.***
  Las imágenes de las embarcaciones que poseen una eslora superior a 21 metros que incluyen tanto barcos de recreo, pesquero como veleros, los cuales, hacen que los textos deban ser de un mayor tamaño frente al resto de los *datasets*, aunque estos puedan estar mayor distorsionados debido a la curvatura que poseen las embarcaciones.

  Los resultados en este caso muestran que PaddleOCR con la mayor tasa de acierto, sobresalen frente a los otros modelos alcanzado casi el 80 % , sugiriendo que este modelo es el que hace frente de una mejora manera las textos distorsionados debido a su arquitectura. Por otro lado EasyOCR y Pytesseract mantienen un rendimiento medio a lo largo de todas las pruebas ejecutadas, con tasas de acierto similares: 48,74 % y 44,77 % respectivamente.

	Paddleocr	**Easyocr**	**Pytesseract**	**Kerasocr**	**MixNet+Clip4str**
CER	21,04 %	51,26 %	56,64 %	67,5 %	47,28 %
WER	63,14 %	90,93 %	102,75 %	98,01 %	67,74 %
MER	56,09 %	80,51 %	67,81 %	76,87 %	67,74 %
WIL	58,63 %	83,42 %	68,81 %	78,98 %	67,74 %
WIP	41,37 %	15,11 %	2,75 %	21,02 %	32,26 %
Distancia levensthein	2,51	5,72	6,31	7,03	5,03
Tasa de acierto	79,8 %	48,74 %	44,77 %	34,85 %	51,25 %

Cuadro 5.6: Resultados de la ejecución de los modelos con imágenes de barcos grandes.

En las siguiente tabla, encontraremos la respectiva comparación entre los recortes de referencia que forman el *ground truth* y los resultados obtenidos tanto de detección como la transcripción por parte de los modelos.

Ground truth	PaddleOCR	EasyOCR	Pytesseract	KerasOCR	Mixnet+Clip4str
 Texto: *DE TABURIENTE*	Sin resultado	Sin resultado	Sin resultado	Sin resultado	Sin resultado
 Texto: *ARMAS*	Sin resultado	Sin resultado	Sin resultado	Sin resultado	 Texto: *ARMAS*
 Texto: *VOLCAN DE TABURIENTE*	 Texto: -	 Texto: -	 Texto: -	 Texto: *reloie d luust*	Sin resultado
 Texto: *SANTA CRUZ DE TENERIFE*	 Texto: *S.CE TARURENT*	 Texto: -	 Texto: *pe*	 Texto: -	Sin resultado

Cuadro 5.7: Comparación de modelos OCR con Ground Truth de barcos grandes

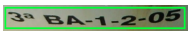
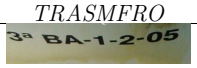
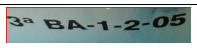
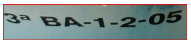
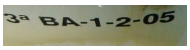
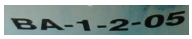
- ***Barcos pequeños.***
  Cuando hablamos de barcos pequeños, hacemos referencia a aquellas embarcaciones que posean una eslora inferior a 21 metros. La complicación en este caso viene derivada del tamaño que posee el barco con respecto al resto de la imagen, lo que a su vez genera que los *píxeles* se encuentran más próximos entre ellos lo que deriva en que se puedan crear solapamientos entre cada uno de ellos, dificultando la labor de reconocerlos.

  En este caso, el modelo que peor rendimiento presenta es KerasOCR, teniendo que realizar mas de 8 cambios en la totalidad de las palabras, para alcanzar la palabra de referencia. En contra, tenemos que, de nuevo, PaddleOCR y MixNet con Clip4str, muestran un rendimiento igual, siendo la diferencia entre ambos despreciable, ya que es inferior al 1 %.

	Paddleocr	**Easyocr**	**Pytesseract**	**Kerasocr**	**MixNet+Clip4str**
CER	35,29 %	63,61 %	51,03 %	77,35 %	32,32 %
WER	76,88 %	94,59 %	93,51 %	91,46 %	75 %
MER	75,62 %	86,89 %	65,96 %	88,96 %	75 %
WIL	76,46 %	87,32 %	67,11 %	88,96 %	75 %
WIP	23,54 %	12,68 %	2,37 %	8,54 %	25 %
Distancia levensthein	3,96	7,12	5,73	8,66	5,5
Tasa de acierto	63,23 %	35,73 %	48,77 %	23,34 %	62,07 %

Cuadro 5.8: Resultados de la ejecución de los modelos con imágenes de barcos pequeños.

En las siguientes imágenes, encontraremos la respectiva comparación entre las imágenes de referencia y las obtenidas finalmente.

Ground truth	PaddleOCR	EasyOCR	Pytesseract	KerasOCR	Mixnet+Clip4str
 Texto: *TRANS-MALLERO*	 Texto: *TRASMFRO*	 Texto: *tRasMaLLERO*	 Texto: *WOW*	 Texto: *nzlos ss*	 Texto: *TRASMALLERO*
 Texto: *3ª BA-1-2-05*	 Texto: *3 BA-1-2-05*	 Texto: *32 BA-1-2-05*	 Texto: -	 Texto: -	 Texto: *BA-1-2-05*

Cuadro 5.9: Comparación de modelos OCR con Ground Truth de barcos pequeños

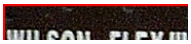
- ***Barcos cargueros***.
  En cuanto a los cargueros, el total de las imágenes muestran embarcaciones que se dedican específicamente al transporte de contenedores. La mayor dificultad que nos podemos encontrar al realizar la inferencia sobre ellas es que no sólo se puede encontrar texto referente al buque, como puede ser el nombre, la nacionalidad o el *IMO*, sino que, a su vez, los contenedores poseen identificadores propios. Este punto genera que se deba realizar un proceso de filtrado sobre los textos que se obtienen, de manera que si no cumplen con el patrón que debe seguir, por ejemplo, la matrícula, se descarta.

  Con respecto a los resultados, claramente PaddleOCR es el que posee una mayor tasa de acierto y las menores tasas de error con respecto al resto de modelos.

	Paddleocr	**Easyocr**	**Pytesseract**	**Kerasocr**	**MixNet+Clip4str**
CER	23,62 %	55,41 %	65,54 %	57,41 %	41,59 %
WER	53,17 %	90,69 %	128,24 %	87,89 %	61,84 %
MER	49,21 %	79,19 %	78,24 %	68,07 %	61,84 %
WIL	51,69 %	81,13 %	79,12 %	70,75 %	61,84 %
WIP	48,31 %	18,87 %	4,41 %	26,12 %	38,16 %
Distancia levensthein	2,82	5,72	7,62	5,86	4,61
Tasa de acierto	76,53 %	46,16 %	37,23 %	43,14 %	56,47 %

Cuadro 5.10: Resultados de la ejecución de los modelos con imágenes de cargueros.

Como en los casos anteriores, se procede a mostrar un ejemplo de los resultados obtenidos de manera visual

Ground truth	PaddleOCR	EasyOCR	Pytesseract	KerasOCR	Mixnet+Clip4str
 Texto: *WILSON FLEX III*	 Texto: -	 Texto:*A San ai T*	 Texto: *'Tl ebm.) Fiano see ok ae*	 Texto: -	 Texto: *FLEXIII*
 Texto: *IMO 9911458*	 Texto: *HO9911458*	 Texto: -	 Texto: *3ª BA-1-2-05*	 Texto: -	sin resultado
 Texto: *MADEIRA*	 Texto: *MADEIRA*	 Texto: *n Te*	 Texto: *ds ee*	 Texto: *an tasn dadeira*	Sin resultado
 Texto: *ARKON*	Sin resultado	Sin resultado	Sin resultado	Sin resultado	 Texto: *ARKON*
 Texto: *WILSON*	Sin resultado	Sin resultado	Sin resultado	Sin resultado	 Texto: *WILSON*

Cuadro 5.11: Comparación de modelos OCR con Ground Truth de cargueros

- ***Distancia larga***.
  En esos casos, se presenta una necesidad evidente de tener que realizar diferentes transformaciones como recortes a los diferentes barcos que se puedan encontrar en las imágenes, además de reescalar esos recortes con el fin de hacer visible algún tipo de carácter. Por este motivo, el rendimiento que presentan los modelos se ve disminuido, tal es así que en los modelos EasyOCR, Pytesseract y KerasOCR llegan a mostrar unas tasas de error superiores al 100 % en la tasa de error de palabras. Esto es debido a que el entorno es de los más demandantes que conforman el *dataset*, y aunque se le apliquen las diferentes modificaciones anteriormente comentadas, la calidad de la imagen continúa siendo muy pobre, y por ende, los resultados.

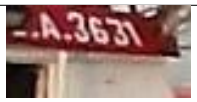
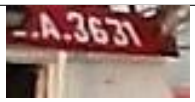
  Finalmente, PaddleOCR (modelo de una sola etapa) y la combinación de MixNet con Clip4str (modelo que se ejecuta en dos etapas), posee una tasa de acierto que se puede considerar como correcta, y la diferencia entre ambos es de menos de un 3 %.

  En las siguientes imágenes, encontraremos la respectiva comparación entre las imágenes de referencia y la obtenida finalmente.

- ***Distancia corta.***
  El modelo combinatorio formado por MixNet y Clip4STR presenta resultados sobresalientes en términos de precisión, superando el 80 % de tasa de acierto. Este rendimiento es particularmente relevante, al considerar uno de los desafíos mencionados en el apartado 2.3, como el uso frecuente del carácter ª, el cual, debido a la proximidad de las

	Paddleocr	Easyocr	Pytesseract	Kerasocr	MixNet+Clip4str
CER	39,94 %	67,92 %	65,97 %	76,13 %	41,04 %
WER	85 %	128,66 %	101,28 %	103,67 %	73,48 %
MER	82,08 %	90,89 %	75,26 %	90,96 %	73,48 %
WIL	90,61 %	92,24 %	75,96 %	91,34 %	73,48 %
WIP	10,27 %	9,63 %	4,66 %	12,05 %	26,52 %
Distancia levensthein	5,08	8,64	6,94	9,39	4,27
Tasa de acierto	58,8 %	35,39 %	37,23 %	23,58 %	61,48 %

Cuadro 5.12: Resultados de la ejecución de los modelos con imágenes procedentes desde distancia largas.

Ground truth	PaddleOCR	EasyOCR	Pytesseract	KerasOCR	Mixnet+Clip4str
 Texto: *E.A.3631*	 Texto: *.A.3631*	 Texto: *74.3631*	 Texto: -	 Texto: -	Sin resultado
	 Texto: -	 Texto: *nun*	 Texto: -	 Texto: *ulíltin*	 Texto: *QUEEK-BOSEREEMO*

Cuadro 5.13: Comparación de modelos OCR con Ground Truth de distancia larga

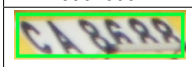
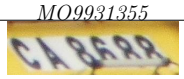
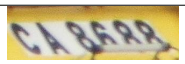
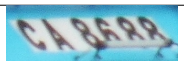
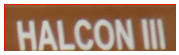
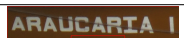
embarcaciones a la zona de captura de las imágenes, las matriculas son mas visibles, y por ende, el carácter se diferencia con mayor precisión. En cuanto a la distancia Levenshtein, esta nos revela que el modelo solo ha realizado, en promedio, una transformación para obtener el resultado correcto.

En contrapuesta, los modelos EasyOCR, PyTesseract y KerasOCR muestran un rendimiento que se mantiene con los resultados observados en los puntos anteriores. Esto nos indica que son más adecuados para aplicaciones genéricas en las que no existen entornos tan demandantes y los factores externos no afectan de igual manera, como puede ser el escaneo de documentos o la lectura de carteles y señales de tráfico.

	Paddleocr	Easyocr	Pytesseract	Kerasocr	MixNet+Clip4str
CER	21,46 %	57,68 %	60,58 %	60,66 %	14,13 %
WER	66,06 %	107,32 %	108,64 %	104,56 %	58,97 %
MER	61,18 %	85,05 %	74,07 %	74,38 %	58,97 %
WIL	62,28 %	87,01 %	74,75 %	76,06 %	58,97 %
WIP	37,72 %	12,05 %	2,11 %	20,16 %	41,03 %
Distancia levensthein	2,2	6,04	6,27	6,28	1,51
Tasa de acierto	79,77 %	43,3 %	41,05 %	38,67 %	82,6 %

Cuadro 5.14: Resultados de la ejecución de los modelos con imágenes procedentes desde distancia corta.

En las siguientes imágenes, encontraremos la respectiva comparación entre las imágenes de referencia y la obtenida finalmente.

Ground truth	PaddleOCR	EasyOCR	Pytesseract	KerasOCR	Mixnet+Clip4str
 Texto: *IMO 9931355*	 Texto: *MO9931355*	 Texto: *7 40? S*	 Texto: -	 Texto: -	Sin resultados
 Texto: *CA 8688*	 Texto: -	 Texto: *62r*	 Texto: -	 Texto: -	 Texto: *CA8688*
 Texto: *HALCON III*	 Texto: *HALCONII*	 Texto: *HALCON*	 Texto: *HALCON iil*	 Texto: -	 Texto: *HALCON*  Texto: *III*
 Texto: *ARAUCARIA I*	 Texto: *ARAUCARIA*	 Texto: *ARAUCARIA*	 Texto: *la eer tae ae*	 Texto: *araucarta*	 Texto: *ARAUCARIA*
 Texto: *PILOT*	Sin resultados	Sin resultados	Sin resultados	Sin resultados	 Texto: *PILOT*
 Texto: *898*	 Texto: *898*	 Texto: *898*	 Texto: *BF*	 Texto: -	 Texto: *898*

Cuadro 5.15: Comparación de modelos OCR con Ground Truth de distancia corta

5.3. Reconocimiento de matrículas internacionales

Dado los resultados prometedores obtenidos al ejecutar el modelo sobre un conjunto de datos exclusivamente nacionales, surge la necesidad de ampliar su aplicabilidad a un contexto internacional, donde los patrones de las matrículas de las embarcaciones difieren significativamente de los estudiados. Con este objetivo, se propone analizar la capacidad del modelo para identificar embarcaciones de distintos países. En línea con los intereses comerciales de la empresa QAISC, se desarrolló un programa que verifica si el texto detectado cumple con las normativas específicas de matriculación de Alemania, Italia y Francia, permitiendo así evaluar su eficacia en estos nuevos contextos.

Los patrones de matricula de los países anteriormente comentados son los siguientes:

- **Italia**. Su patrón destaca por poseer una organización similar a la estructura de árbol. En los nodos raíces, se encuentra la abreviatura de las ciudades que poseen el rango de autoridad portuaria, y es a partir de ellas de las que se generan el resto de patrones. En un nivel inferior, encontramos las matriculas asociada a las oficinas de los distritos marítimos, las cuales se distinguen de las anteriores, al incluir un dígito previo a la abreviatura. Por último, se encuentra el tercer tipo de matricula, cuyo formato es

similar al anterior, ya que también posee el dígito previo a la abreviatura, pero en este caso esta asociada a una oficina marítima local.

A este sistema de codificación de matriculas que posee Italia, hay que añadirle, posterior a la abreviatura, entre tres y cuatro dígitos, dependiendo del año en el que se haya matriculado.

(a) Ejemplo de patrón de autoridad portuaria

(b) Ejemplo de patrón de distrito marítimo

(c) Ejemplo de patrón de Roma

(d) Ejemplo de patrón de puerto

Ilustración 5.1: Ejemplos de patrones de embarcaciones de Italia. Fuente: ShipSpotting

- **Francia**. En el caso de Francia, nos encontramos con cuatro formatos de matriculas dependiendo del año de registro y del tipo de embarcación. En primer lugar, comenzaremos explicando el patrón para las embarcaciones mas antiguas. Este esta formado unicamente por seis dígitos. El siguiente patrón, consiste en una actualización del previo, en el que se añade, antes de los dígitos, el código del puerto en el que se atraca el barco. El ultimo patrón es referente a las embarcaciones de recreo, ademas de ser el más actual, y consiste en añadir, entre el código de puerto y los dígitos, un carácter empezando por la letra ***A***, y siempre y cuando en ese puerto se hayan agotado las 999999 combinaciones posibles de los dígitos.

  El ultimo patrón se corresponde a las embarcaciones de la guardia costera, el cual solo posee un carácter seguido de tres dígitos.

(a) Ejemplo embarcación con patrón actual (b) Ejemplo embarcación con patrón antiguo

(c) Ejemplo de la guardia costera francesa

Ilustración 5.2: Ejemplos de patrones de embarcaciones de Francia. Fuente: ShipSpotting

- **Alemania**. En el caso de las embarcaciones alemanas, su matricula posee un patrón que está conformado en un primer lugar por la abreviatura de las diferentes ciudades y distritos marítimos del país, en la que haya sido registrada dicha embarcación. En segundo lugar, el patrón también incluye un intervalo de entre dos y cinco dígitos.

(a) Ejemplo de patrón de autoridad portuaria

(b) Ejemplo de patrón de distrito marítimo

Ilustración 5.3: Ejemplos de patrones de embarcaciones de Alemania. Fuente: ShipSpotting

Gracias a este avance, se ha logrado mejorar la verificación de si los textos reconocidos en las imágenes corresponde, a un texto simple y común, o a una matrícula. Para ello, se ha desarrollado un *script* que compara el texto reconocido con los patrones específicos de matrícula de cada país.

Este proceso de comparación se ha implementado utilizando, en primer lugar, expresiones regulares que representa la estructura de las matriculas de los diferentes países. Dado que estos se encuentran almacenados en un fichero externo, existe la posibilidad de modificarlos de manera que, se añadan nuevas expresiones que representen otros países. A través de estas expresiones, se mide la distancia entre el texto de referencia y el texto obtenido en la ejecución del reconocimiento. Para determinar la similitud entre ambos, se aplican técnicas de comparación de cadenas como es la distancia Levenshtein, lo que permite identificar si el texto corresponde a una matrícula válida o si se trata de otro tipo de texto.

Capítulo 6

Conclusiones y líneas futuras

En este Trabajo de Fin de Título se ha realizado una investigación exhaustiva, tanto teórica como práctica, sobre el rendimiento de diversos modelos de detección y reconocimiento de caracteres. El análisis incluyó modelos tradicionales y de reciente desarrollo, considerando tanto arquitecturas de una etapa como de múltiples etapas. Como resultado, se estableció una base sólida que permite iniciar experimentos detallados con cada uno de estos enfoques, facilitando su comparación y evaluación en diferentes contextos.

Asimismo, también se ha cumplido con el objetivo de crear un *dataset* amplio y diversificado con clases representativas parea evaluar el desempeño de cada modelo en los diferentes escenarios. Este conjunto de datos incluye imágenes obtenidas tanto de día como de noche; con muestras de barcos de diferentes tamaños, bien con una eslora superior a 21 metros de eslora, como inferiores, además de cargueros dedicados al transporte de mercancías. Por último también tendrá imágenes con variaciones en la distancia entre la embarcación y las cámaras que las captan.

A través de la ejecución de todos los modelos y la obtención de sus respectivos resultados, se ha podido identificar y comprobar los aspectos más influyentes cuando el entorno es tan cambiante, tal y como es el marítimo. A pesar de que los modelos puedan presentar debilidades, los factores externos, poseen un mayor impacto en estos resultados, entendiéndose éstos como la distancia o la perspectiva desde la que se obtienen las imágenes de los barcos, además de los reflejos generados por los rayos del sol en el barco, así como el tamaño o la fuente utilizada para escribir el nombre o la matrícula de la embarcación.

De los resultados obtenidos, se puede determinar que los modelos con mejor desempeño en general son *PaddleOCR*, siendo un modelo de ejecución de una sola etapa, y la combinación de *Mixnet* con *Clip4str*. Cada uno de ellos muestran mejores resultados en entornos específicos; por ejemplo, mientras que *Paddleocr*, muestra un rendimiento excelente con el *dataset* de noche, *Mixnet* con *Clip4str* lo posee con el conjunto de imágenes obtenidas desde una distancia lejana.

Tras la obtención de los resultados comentados anteriormente, se podrían considerar diferentes líneas de trabajo con las que mejorar y avanzar hacia un prototipo mayor. En primer lugar, se encuentra la posibilidad de implementar funcionalidades adicionales como la capacidad de no sólo tratar con bloques de imágenes estáticas, sino también evaluar los modelos que han obtenido mejores resultados, mediante vídeos cortos o *timelapses* de no más de una duración determinada. Esto aportaría la capacidad de obtener de un mismo barco múltiples imágenes desde diferentes perspectivas, usándolas como entrada de datos y calculando un promedio, de manera similar a como se realiza con los sistemas de video vigilancia en los supermercados para el control de entrada y salida de los vehículos a partir de su matricula. De igual manera, posterior a esa ejecución, se podría aumentar el caso de estudio a directamente aplicar la ejecución de los modelos sobre el *streaming* de las diferentes cámaras de seguridad que poseen los puertos de Canarias.

Esta mejora potencial también se podría complementar con la expansión del conjunto de casos de estudio, incorporando embarcaciones que posean abanderamiento de otros países tanto europeos como en otros continentes, lo que enriquecería la proyección del proyecto.

Asimismo, y con el objetivo de mejorar el proceso de detección de las embarcaciones en situaciones complejas, tales como oclusiones parciales o la detección del barco a través de cajas contenedoras, se explora la posibilidad de introducir técnicas de segmentación. En este caso se propone investigar dos estrategias de manera paralela, comenzando con la segmentación semántica, y continuando con la segmentación de instancias. La aplicación de estos métodos, posibilita diferenciar de manera precisa el contorno de la embarcación, pudiéndolo extraerla del fondo, para posteriormente aplicar sobre este recorte, el algoritmo de OCR fruto de este trabajo.

Capítulo 7

Glosario

Amura Costado del barco en la cual se comienza a estrechar para dar lugar a la proa del mismo.. 18

Ground-truth Archivo utilizado como referencia debido a que este contiene una relación de, en este caso, imagen y sus respectivos textos, además de las coordenadas en las que este se encuentra.. 56

Pantalán Pasarela flotante que se utiliza como punto de amarre de las embarcaciones, permitiendo acceder a éstas a pie.. 58

Scrapper Herramienta que nos permite descargar de manera automática toda aquella información que se encuentre disponible en una pagina web, pudiéndose organizar por categorías.. 56

Abanderamiento Acto administrativo que en España esta regido por el Real Decreto 1027/1989, de 28 de julio, por el cual una embarcación puede izar la bandera del correspondiente al pabellón español.. 18

Casco En náutica, es el armazón o la estructura externa de una embarcación que puede estar hecha de diferentes materiales como madera, hierro o fibra de vidrio, entre otros, que es fundamental para que el barco flote, ademas de ser determinante para la maniobildiad del mismo.. 17

Distrito marítimo Órganos descentralizados dependientes del Ministerio de Transporte, Movilidad y Agenda Urbana que nacen con el objetivo de atender y resolver problemas puntuales que puedan surgir relacionada con el registro, abanderamiento y tramitación de expedientes o notificaciones, entre otros. Identifican a una zona costera en la que destaque el puerto de la provincia marítima correspondiente.. 18

Eslora Longitud de la embarcación medida desde la parte mas saliente de la proa hasta la parte mas saliente de la popa, conocida de esta manera como *eslora máxima*. 16

Popa Parte trasera de una embarcacion, donde suelen encontrarse los motores en barcos de pequeña eslora. 17

Bibliografía

[1] Clova AI. *CRAFT: Character Region Awareness for Text detection*. 2024. URL: https://github.com/clovaai/CRAFT-pytorch.

[2] Jaided AI. *Jaided AI*. Sitio web. URL: https://www.jaided.ai/.

[3] Asana. *What is Kanban?* 2024. URL: https://asana.com/es/resources/what-is-kanban.

[4] Rowel Atienza. *Vision Transformer for Fast and Efficient Scene Text Recognition*. 2021. arXiv: 2105.08582 [cs.CV]. URL: https://arxiv.org/abs/2105.08582.

[5] Youngmin Baek et al. «Character Region Awareness for Text Detection». En: *arXiv* (abr. de 2019). arXiv: 1904.01941 [cs.CV]. URL: https://arxiv.org/pdf/1904.01941.

[6] Juan Ignacio Bagnato. *Breve Historia de las Redes Neuronales Artificiales*. Sep. de 2018. URL: https://aprendemachinelearning.com/breve-historia-de-las-redes-neuronales-artificiales/.

[7] Chandra Churh — Towards Data Science Chatterjee. *An Approach Towards Convolutional Recurrent Neural Networks*. Nov. de 2019. URL: https://towardsdatascience.com/an-approach-towards-convolutional-recurrent-neural-networks-a2e6ce722b19.

[8] CVAT. *Computer Vision Annotation Tool (CVAT)*. 2024. URL: https://www.cvat.ai/.

[9] *Definición de "distancia."^en el Diccionario de la lengua española (DLE)*. URL: https://dle.rae.es/distancia#otras.

[10] *Diccionario de la Real Academia Española*. 2024. URL: https://dle.rae.es/documento.

[11] 11 IONOS Digitalguide. *Graph Neural Network*. sin fecha. URL: https://www.ionos.es/digitalguide/online-marketing/marketing-para-motores-de-busqueda/graph-neural-network/.

[12] Yongkun Du et al. *SVTR: Scene Text Recognition with a Single Visual Model*. 2022. arXiv: 2205.00159 [cs.CV]. URL: https://arxiv.org/abs/2205.00159.

[13] Estudios Onavegas. *NOTOS: Identificar un Barco*. 2024. URL: https://www.estudiasonavegas.com/images/Apuntes/Juanitu/Juanitu-apuntes/NOTOS%20IDENTIFICAR%20UN%20BARCO.pdf.

[14] Estudios Onavegas. *NOTOS: Identificar un Barco*. 2024. URL: https://www.estudiasonavegas.com/images/Apuntes/Juanitu/Juanitu-apuntes/NOTOS%20IDENTIFICAR%20UN%20BARCO.pdf.

[15] Shancheng Fang et al. *Read Like Humans: Autonomous, Bidirectional and Iterative Language Modeling for Scene Text Recognition*. 2021. arXiv: 2103.06495 [cs.CV]. URL: https://arxiv.org/abs/2103.06495.

[16] Candela García Fernández. *Python en IA: aplicaciones prácticas y bibliotecas clave*. Dic. de 2023. URL: https://openwebinars.net/blog/python-en-ia-aplicaciones-practicas-y-bibliotecas-clave/.

[17] F. Filippidou y L. Moussiades. « Benchmarking of IBM, Google and Wit Automatic Speech Recognition Systems». En: *Artificial Intelligence Applications and Innovations*. 2020, págs. 73-82. DOI: 10.1007/978-3-030-49161-1_7.

[18] Python Software Foundation. *About Python*. 2024. URL: https://www.python.org/about/.

[19] Navdeep Singh — XenonStack Insights Gill. *CRNN for Text Recognition*. Mayo de 2023. URL: https://www.xenonstack.com/insights/crnn-for-text-recognition.

[20] S. Grossberg. «Recurrent neural networks». En: *Scholarpedia* 8.2 (2013). revision #138057, pág. 1888. DOI: 10.4249/scholarpedia.1888.

[21] Larry Hardesty. Abr. de 2017. URL: https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414.

[22] Sepp Hochreiter y Jürgen Schmidhuber. «Long Short-term Memory». En: *Neural Computation* 9.8 (1997), págs. 1735-1780. DOI: 10.1162/neco.1997.9.8.1735. URL: https://www.researchgate.net/publication/13853244_Long_Short-term_Memory.

[23] IBM. *CountVectorizer*. 2024. URL: https://www.ibm.com/reference/python/countvectorizer#:~:text=CountVectorizer%20is%20a%20class%20in,of%20word%20or%20token%20counts.

[24] IBM. *Neural Networks*. sin fecha. URL: https://www.ibm.com/es-es/topics/neural-networks.

[25] Oana Ignat et al. «OCR Improves Machine Translation for Low-Resource Languages». En: *Findings of the Association for Computational Linguistics: ACL 2022*. Ed. por Smaranda Muresan, Preslav Nakov y Aline Villavicencio. Dublin, Ireland: Association for Computational Linguistics, mayo de 2022, págs. 1164-1174. DOI: 10.18653/v1/2022.findings-acl.92. URL: https://aclanthology.org/2022.findings-acl.92.

[26] IHS Markit. *IMO Number Requests*. 2024. URL: https://imonumbers.ihs.com/.

[27] JaidedAI. *EasyOCR: Ready-to-use OCR with 80+ languages supported including Chinese, Japanese, Korean and Thai*. 2022. URL: https://github.com/JaidedAI/EasyOCR?tab=readme-ov-file.

[28] Fei Jiang y Zengfu Wang. «TextFourierNet: Arbitrary-shaped Scene Text Detection Based on Fourier Contour Modeling». En: *2022 China Automation Congress (CAC)*. 2022, págs. 1571-1576. DOI: 10.1109/CAC57257.2022.10055369.

[29] Hui Jiang et al. *Reciprocal Feature Learning via Explicit and Implicit Tasks in Scene Text Recognition*. 2021. arXiv: 2105.06229 [cs.CV]. URL: https://arxiv.org/abs/2105.06229.

[30] A Karthikeyan. *History of Neural Network*. Sep. de 2018. URL: https://medium.com/@karthikeyana97/history-of-neural-network-d1333760f0c5.

[31] Keras. *About Keras*. 2024. URL: https://keras.io/about/.

[32] Keras-OCR. *Keras-OCR Documentation*. 2024. URL: https://keras-ocr.readthedocs.io/en/latest/.

[33] Hui Li et al. *Show, Attend and Read: A Simple and Strong Baseline for Irregular Text Recognition*. 2019. arXiv: 1811.00751 [cs.CV]. URL: https://arxiv.org/abs/1811.00751.

[34] Minghui Liao et al. *Real-time Scene Text Detection with Differentiable Binarization*. 2019. arXiv: 1911.08947 [cs.CV]. URL: https://arxiv.org/abs/1911.08947.

[35] Minghui Liao et al. *Real-Time Scene Text Detection with Differentiable Binarization and Adaptive Scale Fusion*. 2022. arXiv: 2202.10304 [cs.CV]. URL: https://arxiv.org/abs/2202.10304.

[36] Wei Liu et al. *STAR-Net: A SpaTial Attention Residue Network for Scene Text Recognition*. Sep. de 2016. DOI: 10.5244/C.30.43.

[37] Shangbang Long et al. *TextSnake: A Flexible Representation for Detecting Text of Arbitrary Shapes*. 2020. arXiv: 1807.01544 [cs.CV]. URL: https://arxiv.org/abs/1807.01544.

[38] Aditya Mahajan. *EasyOCR: A Comprehensive Guide*. 2022. URL: https://medium.com/@adityamahajan.work/easyocr-a-comprehensive-guide-5ff1cb850168.

[39] Aditya Mahajan. «EasyOCR: A Comprehensive Guide». En: *Medium* (oct. de 2023). URL: https://medium.com/@adityamahajan.work/easyocr-a-comprehensive-guide-5ff1cb850168.

[40] MathWorks. *Neural Network Toolbox*. sin fecha. URL: https://la.mathworks.com/discovery/neural-network.html.

[41] Warren S. McCulloch y Walter Pitts. *A Logical Calculus of the Ideas Immanent in Nervous Activity*. 1943. URL: https://www.cs.cmu.edu/~./epxing/Class/10715/reading/McCulloch.and.Pitts.pdf.

[42] Memgraph. *Cosine Similarity in Python with scikit-learn*. URL: https://memgraph.com/blog/cosine-similarity-python-scikit-learn.

[43] Marc Mercier. *Deep Neural Network*. Nov. de 2022. URL: https://botpress.com/es/blog/deep-neural-network.

[44] Ministerio de Fomento. *Reglamento de Radiocomunicaciones*. 2024. URL: https://www.fomento.gob.es/AZ.BBMF.Web/documentacion/pdf/R16520.pdf.

[45] Ministerio de Fomento. *Reglamento de Radiocomunicaciones*. 2024. URL: https://www.fomento.gob.es/AZ.BBMF.Web/documentacion/pdf/R16520.pdf.

[46] Ministerio de Transportes, Movilidad y Agenda Urbana. *Identificación del Servicio Móvil Marítimo (MMSI)*. 2024. URL: https://www.transportes.gob.es/

marina-mercante/radiocomunicaciones/solicitud-mmsi/identificacion-del-servicio-movil-maritimo-mmsi.

[47] Mayank — Towards Data Science Mishra. *Convolutional Neural Networks Explained*. Ago. de 2020. URL: https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939.

[48] Andrew Morris, Viktoria Maier y Phil Green. En: *From WER and RIL to MER and WIL: improved evaluation measures for connected speech recognition*. Oct. de 2004. DOI: 10.21437/Interspeech.2004-668.

[49] NumPy. *NumPy*. 2024. URL: https://numpy.org/.

[50] Tesseract OCR. *Tesseract OCR*. 2024. URL: https://github.com/tesseract-ocr/tesseract.

[51] Tesseract OCR. *Tesseract OCR Documentation*. 2024. URL: https://tesseract-ocr.github.io/tessdoc/.

[52] OpenCV. *OpenCV*. 2024. URL: https://opencv.org/.

[53] Organización Marítima Internacional. *Esquema de números de identificación OMI*. 2024. URL: https://www.imo.org/es/OurWork/MSAS/Paginas/IMO-identification-number-scheme.aspx.

[54] Organización Marítima Internacional. *Preguntas Frecuentes (FAQ)*. 2024. URL: https://www.imo.org/es/About/Paginas/FAQs.aspx.

[55] International Maritime Organization. *Frequently Asked Questions*. 2024. URL: https://www.imo.org/es/About/Paginas/FAQs.aspx.

[56] PaddlePaddle. *PaddleOCR Algorithm Overview*. 2023. URL: https://github.com/PaddlePaddle/PaddleOCR/blob/release/2.6/doc/doc_en/algorithm_overview_en.md.

[57] PaddlePaddle. *PaddlePaddle Official Website*. 2024. URL: https://www.paddlepaddle.org.cn/en/.

[58] Pillow. *Pillow Documentation*. 2024. URL: https://pillow.readthedocs.io/en/stable/.

[59] Zhi Qiao et al. *SEED: Semantics Enhanced Encoder-Decoder Framework for Scene Text Recognition*. 2020. arXiv: 2005.10977 [cs.CV]. URL: https://arxiv.org/abs/2005.10977.

[60] Adrian Rosebrock. *Tesseract Page Segmentation Modes (PSMs) Explained. How to Improve Your OCR Accuracy*. Abr. de 2021. URL: https://pyimagesearch.com/2021/11/15/tesseract-page-segmentation-modes-psms-explained-how-to-improve-your-ocr-accuracy/.

[61] Sumit — Saturn Cloud Saha. *A Comprehensive Guide to Convolutional Neural Networks: The ELI5 Way*. Dic. de 2018. URL: https://saturncloud.io/blog/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way/.

[62] Ken Schwaber y Jeff Sutherland. *The Scrum Guide - 2020*. 2020. URL: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf.

[63] Baoguang Shi et al. *Robust Scene Text Recognition with Automatic Rectification*. 2016. arXiv: 1603.03915 [cs.CV]. URL: https://arxiv.org/abs/1603.03915.

[64] R. Smith. «An Overview of the Tesseract OCR Engine». En: *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*. Vol. 2. 2007, págs. 629-633. DOI: 10.1109/ICDAR.2007.4376991.

[65] Mingxing Tan y Quoc V. Le. *MixConv: Mixed Depthwise Convolutional Kernels*. 2019. arXiv: 1907.09595 [cs.CV]. URL: https://arxiv.org/abs/1907.09595.

[66] TensorFlow. *TensorFlow*. 2024. URL: https://www.tensorflow.org/?hl=es-419.

[67] The Pew Charitable Trusts. *El número OMI explicado*. 2017. URL: https://www.pewtrusts.org/es/research-and-analysis/fact-sheets/2017/05/the-imo-number-explained.

[68] Javier Torres. *Redes Neuronales Recurrentes*. Sep. de 2019. URL: https://torres.ai/redes-neuronales-recurrentes/.

[69] *Trello*. URL: https://trello.com/es.

[70] Unión Internacional de Telecomunicaciones. *UIT: Comprometida para conectar el mundo*. 2024. URL: https://www.itu.int/es/Pages/default.aspx.

[71] Universidad Europea. *¿Qué es el producto mínimo viable (PMV)?* 2024. URL: https://universidadeuropea.com/blog/producto-minimo-viable/#:~:text=dinero%20y%20esfuerzo.-,%C2%BFQu%C3%A9%20es%20el%20producto%20m%C3%ADnimo%20viable%3F,las%20necesidades%20de%20los%20clientes.

[72] Pengfei Wang et al. «A Single-Shot Arbitrarily-Shaped Text Detector based on Context Attended Multi-Task Learning». En: *Proceedings of the 27th ACM International Conference on Multimedia*. MM '19. ACM, oct. de 2019. DOI: 10.1145/3343031.3350988. URL: http://dx.doi.org/10.1145/3343031.3350988.

[73] Pengfei Wang et al. «PGNet: Real-time Arbitrarily-Shaped Text Spotting with Point Gathering Network». En: *CoRR* abs/2104.05458 (2021). arXiv: 2104.05458. URL: https://arxiv.org/abs/2104.05458.

[74] Wenhai Wang et al. *Shape Robust Text Detection with Progressive Scale Expansion Network*. 2019. arXiv: 1903.12473 [cs.CV]. URL: https://arxiv.org/abs/1903.12473.

[75] Yuxin Wang et al. *From Two to One: A New Scene Text Recognizer with Visual Language Modeling Network*. 2021. arXiv: 2108.09661 [cs.CV]. URL: https://arxiv.org/abs/2108.09661.

[76] Wikipedia. *Cosine similarity*. URL: https://en.wikipedia.org/wiki/Cosine_similarity.

[77] Zi Ye et al. «A Comprehensive Survey of Graph Neural Networks for Knowledge Graphs». En: *IEEE Access* 10 (2022), págs. 75729-75741. DOI: 10.1109/ACCESS.2022.3191784.

[78] Deli Yu et al. *Towards Accurate Scene Text Recognition with Semantic Reasoning Networks*. 2020. arXiv: 2003.12294 [cs.CV]. URL: https://arxiv.org/abs/2003.12294.

[79] Xiaoyu Yue et al. *RobustScanner: Dynamically Enhancing Positional Clues for Robust Text Recognition*. 2020. arXiv: 2007.07542 [cs.CV]. URL: https://arxiv.org/abs/2007.07542.

[80] Ebin Zacharias, Martin Teuchler y Bénédicte Bernier. *Image Processing Based Scene-Text Detection and Recognition with Tesseract*. Abr. de 2020. arXiv: 2004.08079 [cs.CV]. URL: https://arxiv.org/pdf/2004.08079.

[81] Yu-Xiang Zeng et al. *MixNet: Toward Accurate Detection of Challenging Scene Text in the Wild*. 2023. arXiv: 2308.12817 [cs.CV]. URL: https://arxiv.org/abs/2308.12817.

[82] Chengwei Zhang et al. *SPIN: Structure-Preserving Inner Offset Network for Scene Text Recognition*. 2021. arXiv: 2005.13117 [cs.CV]. URL: https://arxiv.org/abs/2005.13117.

[83] Shi-Xue Zhang et al. *Deep Relational Reasoning Graph Network for Arbitrary Shape Text Detection*. 2020. arXiv: 2003.07493 [cs.CV]. URL: https://arxiv.org/abs/2003.07493.

[84] Shuai Zhao et al. *CLIP4STR: A Simple Baseline for Scene Text Recognition with Pre-trained Vision-Language Model*. 2024. arXiv: 2305.14014 [cs.CV]. URL: https://arxiv.org/abs/2305.14014.

[85] Xinyu Zhou et al. *EAST: An Efficient and Accurate Scene Text Detector*. 2017. arXiv: 1704.03155 [cs.CV]. URL: https://arxiv.org/abs/1704.03155.

[86] Yiqin Zhu et al. *Fourier Contour Embedding for Arbitrary-Shaped Text Detection*. 2021. arXiv: 2104.10442 [cs.CV]. URL: https://arxiv.org/abs/2104.10442.