UNIVERSIDAD AUTONÓMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

TRABAJO FIN DE MÁSTER

Sistema de clasificación de estilos caligráficos chinos mediante página web

Máster Universitario en Ingeniería informática

Autor: Qiang Sun

Tutor: Juan Alberto Sigüenza Pizarro
Departamento de ingeniería informática

Junio, 2021

UNIVERSIDAD AUTÓNOMA DE MADRID
Escuela Politécnica Superior
Francisco Tomás y Valiente, N.º 11
Cantoblanco, Madrid, 28049
España
Sistema de clasificación de estilos caligráficos chinos mediante página web
Qiang Sun, Junio 2021
IMPRESO EN ESPAÑA - PRINTED IN SPAIN

Agradecimientos

Dedicado a mis padres y a mi prometida, por estar siempre ahí, por quererme, apoyarme, aconsejarme y darme todo lo impensable siempre. Sin vosotros nada en mi vida hubiera sido posible.

Le agradezco a mi tutor, Juan Alberto Sigüenza Pizarro, por todo el tiempo, esfuerzo y paciencia que ha invertido en mí y en este proyecto, supervisando siempre la correcta ejecución y resolviendo pacientemente todas mis dudas, sin todo esto este proyecto no hubiese sido el mismo.

Finalmente, quisiera transmitir todo mi agradecimiento a los profesores/as que me han permitido realizar sus asignaturas de forma semipresencial debido a mi situación de trabajo.

Abstract

During the last decade, thanks to the great technological advances we have had in all areas, certain customs as traditional as writing are being replaced by keyboard and other digital communication formats.

Writing belongs to a very important part of any culture, as it is the traditional way of inheriting knowledge from our ancestors by centuries after centuries. The writing is part of all cultures to transmit long-lasting knowledge though the time. The writing in Chinese culture had a fundamental role in its 5000 years of history. The Chinese script have not only been increased and enriched over the time, but it has also evolved into five major types of calligraphy, each style with its characteristics, that defines particular periods of the Chinese history.

In this project I have worked on creating a Chinese calligraphy style classification model to help the beginners who want to learn and improve more about Chinese writing styles. This is possible thanks to the frontend with which the user could upload images or taken by photo for the later categorization.

I have developed both parts of the application:

- The front-end layer based on common use and highly demanded web technologies as bootstrap, that allows the usage on different devices.

- The back-end business logic layer based on python API framework and deep learning model for style classification and using third part API for OCR character recognition and English translation.

Keywords: *Machine learning, Bootstrap, API, Image classification, OCR*

Resumen

Durante las últimas décadas, los avances tecnológicos conseguidos están cambiando el modo en el que se realizan muchas actividades, afectando incluso a las más tradicionales y arraigadas. La escritura es una de ellas, que está siendo cambiado desde la escritura tradicional a mano por la escritura actual con teclado (ya sea físico o digital).

La escritura es y ha sido crucial para todas las culturas, principalmente por el hecho de que ha sido el método fundamental para la transmisión de información e ideas entre generaciones. La cultura china es un ejemplo de ello. La escritura ha sido la piedra angular que ha conseguido perdurar la cultura china durante 5.000 años. A lo largo de ellos la escritura se ha ido desarrollando, enriqueciéndose y evolucionando. Durante estos cinco milenios se han agrupado en cinco grandes estilos caligráficos principales, cada uno de los cuales corresponde a una etapa distinta de la historia china.

En este proyecto se ha trabajado en la creación de un modelo para la clasificación de cada uno de estos cinco estilos caligráficos permitiendo a los interesados profundizar en el conocimiento de la cultura china y su escritura. Para ello, tendrán que usar la interfaz de usuario creada en la que se pueden subir fotos con la caligrafía en cuestión para que el modelo las clasifique.
El proyecto consta de dos partes:

- La interfaz de usuario, basada en tecnologías web muy demandadas y usadas como el Bootstrap, que permite su uso en diferentes dispositivos.

- Procesamiento, usando Python API como marco de referencia, modelos de aprendizaje profundo para la clasificación de estilos y API de terceros para el reconocimiento óptico de los caracteres y su traducción al inglés.

Palabras clave: aprendizaje automático, Bootstrap, API, Reconocimiento Óptico de caracteres, clasificación de imágenes, estilo caligráfico

Contenido

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. Introducción y motivación

1.1. Motivación

Durante los últimos años las técnicas de reconocimiento de imágenes, ya sea en reconocimiento de caras, de caracteres, de matrículas de coches o de cualquier otro objeto, se han convertido en una de las áreas que más se ha desarrollado y evolucionado, con la contribución de otras ciencias, principalmente de la estadística y de la biometría, se ha convertido en un área muy transversal, con mucho potencial aún sin desarrollar.

Numerosos creadores, ingenieros y filósofos han tratado el tema de los avances tecnológicos y cómo dichos avances pueden generar efectos colaterales en la sociedad. Sin embargo, esto no es una novedad ya que nunca se puede descubrir el alcance total de cualquier innovación o invento. En este trabajo no se alude a otros usos que los algoritmos puedan tener y se centra únicamente en la técnica y la estadística que hay detrás.

Como ya se ha comentado anteriormente, el área del reconocimiento de imágenes se encuentra en constante desarrollo. El propósito de este trabajo es acercar y determinar la viabilidad del empleo de estos avances para determinar el estilo caligráfico de un texto. En concreto, y como se profundiza más adelante, la escritura china difiere radicalmente con la latina, siendo la principal diferencia la no existencia de un alfabeto limitado de letras, sino que está formado por caracteres que pueden tener significado en sí mismos o asociarse con otros dentro del mismo bloque. Esto dificulta en gran medida su reconocimiento, ya que el número de caracteres distintos que conforman la escritura china es inmenso y, a su vez, es muy difícil de reconocer para aquellas personas y algoritmos que componen todas sus palabras a partir de una veintena o treintena de letras (idiomas latinos). A lo largo de las últimas décadas diversos autores han realizado proyectos de esta índole, aunque suelen estar más centrados en el puro reconocimiento de caracteres y no en el reconocimiento del estilo caligráfico. Ejemplos de esto, y que han ayudado a enfocar la dirección del presente trabajo son: Y. Ge y Q. Huo [1], Y.S-Cheng, G. Su y H. Li [2] o L. Jung y W. Liao [3].

Esto significa un gran desafío, ya que el método usado para este reconocimiento de estilos caligráficos debe tener una gran fiabilidad puesto que, aunque la muestra de referencia sea grande, el número de veces que se repite cada carácter es relativamente bajo. Este problema se ha afrontado haciendo uso de la metodología *Deep Learning*.

1.2. Objetivo

El objetivo que se pretende lograr con este proyecto es estudiar la viabilidad de creación de un proceso automático mediante técnicas de *Machine Learning*, visión

artificial y combinación de otros algoritmos cuya finalidad principal es el reconocimiento de los diferentes estilos caligráficos chinos mediante imágenes. Para ello, se han realizado una serie de labores que empieza con las tareas previas de aprendizaje e interiorización de conceptos, el posterior desarrollo y entrenamiento de una red neuronal enfocado a este problema y la puesta en marcha de la web que servirá de interfaz de usuario. Los cometidos detallados se listan a continuación:

- Hacer acopio de una muestra lo suficientemente grande sobre los distintos estilos caligráficos chinos para el fin previsto.

- Estudio y análisis de las diferentes metodologías en las que se basan las técnicas de reconocimiento, clasificación de imágenes y segmentación actuales con sus ventajas y desventajas.

- Desarrollo y combinación de metodologías *Machine Learning* aplicado al caso concreto de los estilos caligráficos.

- Entender y aprender los conocimientos sobre las tecnologías más demandadas en el sector IT para la creación de servicios y aplicaciones web. Con este conocimiento, se habilita una interfaz de usuario en la que éste puede interactuar con el algoritmo proporcionando imágenes para su posterior clasificación. Además, el usuario tiene la posibilidad de corregir al modelo entrenado en la muestra inicial para mejorar la funcionalidad/fiabilidad.

- La realización de todas estas labores, junto con otras de menor trascendencia pero que también han requerido un gran esfuerzo y la resolución de pequeños problemas que no he considerado necesario mencionar, me ha proporcionado un gran conocimiento sobre este campo. Y, quizás más importante, ha hecho que mi curiosidad con respecto a este tema – refiriéndome más bien al de clasificación de imágenes mediante técnicas de machine learning y no al de estilos caligráficos en particular – aumente, ya que los horizontes que abre son inmensos y puede tener aplicaciones en un gran número de desarrollos.

1.3. Estructura del documento

La memoria se estructura de la siguiente manera. Se comienza haciendo un breve resumen del estado actual de la ingeniería de datos en el ámbito del reconocimiento de imágenes, particularizando en el caso de aquellos ligados a los caracteres chinos. A partir de ahí se describe el proyecto en sí. El apartado tercero se ocupa de exponer en lo que consiste el proyecto, así como las metodologías en las que se ha basado.

El trabajo continúa en el cuarto punto con un repaso de la arquitectura que ha sido necesaria para llevar a cabo el proyecto. El punto quinto, probablemente el culminante, contiene a su vez tres apartados en los que se describe por un lado el

funcionamiento de la red neuronal utilizada, y la lógica que se ha seguido para elegirla, por otro los sistemas usados para mantener el servidor en funcionamiento, y por último se describe la creación de la interfaz de usuario.

El apartado sexto describe las pruebas llevadas a cabo para asegurar el correcto funcionamiento de la página web. Por su parte, en el apartado séptimo se analiza el proyecto creado, dándose unas pinceladas sobre por dónde podría avanzarse en esta área.

Por último, y como mandan los cánones, se acaba con dos apartados que contienen los apéndices y la bibliografía usada.

2. Estado del arte

La caligrafía china es un arte visual muy singular originaria de sinogramas y que se ha desarrollado durante los miles de años de historia de esta civilización. La caligrafía china es uno de los elementos esenciales en la cultura china que se distingue de otras caligrafías como la occidental, la georgiana, etc. por el uso de los sinogramas chinos como base.

Para poder realizar la clasificación es muy importante resaltar el tipo de datos de entrada capturada que se va a analizar. El reconocimiento de la escritura a mano, en general, se puede dividir en dos modalidades: en línea o imagen estática, la primera modalidad, la máquina reconoce la escritura mientras el usuario escribe mediante la trayectoria del trazo, la velocidad de pintado y los componentes conexos. Por el contrario, el reconocimiento de la escritura mediante imágenes se lleva a cabo una vez que se ha completado la escritura, empleando escáner o cámaras para convertir la imagen de la escritura en un conjunto de información pixeladas.

En los siguientes subapartados, se describe el estado del arte del reconocimiento de la escritura china, basándose en una extensa revisión de la literatura, historia, artículos, estudios y patentes. Se examinan los algoritmos de preprocesamiento y post procesamiento de imágenes para texto, los sistemas experimentales actuales y algún producto comercial similar

2.1. Sistema de clasificación de imágenes

El procesamiento de imágenes es un campo que ha evolucionado de manera importante en los últimos años. Desde el punto de vista de un usuario, cuando hablamos de reconocimiento de imágenes uno podría pensar en las imágenes que se buscan en Google, donde las imágenes están catalogadas y donde, tras introducir nuestras palabras clave, este buscador nos ofrece aquellas que según la codificación más se ajusten a lo solicitado. O también podríamos pensar en nuestro teléfono móvil, ya que algunos modelos ya cuentan con separación automática de distintas clases, como podría ser paisajes, retratos o fotografías nocturnas.

Sin embargo, desde el punto de vista del investigador, el procesamiento de imágenes y su posterior clasificación requiere una serie de pasos que varían dependiendo del contenido de la imagen.

En los próximos subapartados, se expone de manera breve los últimos avances en unos tipos de sistemas de clasificación de imágenes. Estos son el reconocimiento facial, el reconocimiento de firmas, y por último el reconocimiento de estilos caligráficos, que se desarrollará en profundidad, ya que es el objeto del trabajo. No obstante, se quiere indicar que ésta no es una clasificación exhaustiva de los

distintos subtipos de reconocimiento de imágenes, sino una exposición breve de los tres principales con más relación con el presente estudio.

2.1.1. Sistemas de reconocimiento facial

Como siempre en este trabajo, se empieza por dar el punto de vista de un usuario, para después ir de menos a más en cuanto a complejidad de la información se refiere. Probablemente, hoy en día cuando se piensa en reconocimiento facial lo primero que se viene a la cabeza sea el bloqueo y desbloqueo del móvil mediante la cámara delantera. También se podría mencionar todas las cámaras de vigilancia que hay en determinadas calles y aeropuertos, en donde se vigila y se graba por motivos de seguridad.

Cuando se piensa en identificar inequívocamente a una persona mediante una imagen se podría pensar en la estatura, la forma física o el color del pelo. Sin embargo, todas estas características son demasiado vagas (incluso algunas pueden ser alteradas) para distinguir a una persona de otra. No obstante, nuestra cara sí tiene una serie de características específicas que la hacen ideal para poder distinguir personas entre sí.

La primera de ellas es, por supuesto, que habitualmente no va cubierta, por lo que es fácilmente analizable a simple vista. Véase por ejemplo cómo en las protestas contra los gobiernos los manifestantes suelen ir con la cabeza tapada e incluso con gafas. Segundo, en una superficie relativamente pequeña de nuestra piel como es la cara, se encuentran los ojos, la boca, la nariz y las orejas. Lo cual permite analizar muchos elementos al mismo tiempo, como por ejemplo de la biometría (iris, retina), distancias, simetrías, tamaños, pliegues, lunares, etc.

Todos estos elementos han hecho del reconocimiento facial un eje vanguardista dentro del reconocimiento de imágenes.

Hoy en día hay fundamentalmente dos tipologías de análisis para el reconocimiento facial: técnicas basadas en la apariencia y técnicas basadas en modelos. En cualquier caso, el primer paso para cualquiera de estas dos familias de técnicas y para otros tipos de sistemas de reconocimiento es el conocido como entrenamiento. Este consiste básicamente en proveer a nuestro sistema, independientemente de cuál sea, una base de datos de imágenes que luego se requiera reconocer. Lo que sí varía dependiendo de la técnica utilizada son las características que ha de tener la imagen (resolución, vistas, etc.) ya que cada técnica requiere *inputs* diferentes.

2.1.2.1 Sistemas basados en la apariencia

Los sistemas basados en la apariencia utilizan un set de imágenes de referencia a partir del cual se procesan una serie de parámetros. Una vez que las imágenes de la base de datos de referencia están procesadas se puede empezar a

analizar las imágenes que realmente se quiere reconocer (objetivo). Estas imágenes se procesan y se extraen los parámetros requeridos, siempre que la calidad de la imagen sea suficiente. A partir de ahí, con unas medidas preestablecidas, se comparará con la base de datos de referencia para comprobar a cuál o cuáles podría pertenecer dicha cara.

Dentro de los sistemas basados en la apariencia hay diferentes tipos. A continuación, se hace una breve mención por no ser objeto principal del tema:

Principal Component Analysis. Algoritmo de reducción dimensional, que facilita la comparación entre los parámetros de las imágenes de referencia. Se realiza una transformación ortogonal de los datos que proporcionan una nueva matriz de datos. En este proceso, a la par que reduce la dimensión de nuestros datos, se intenta que los nuevos que se generen mantengan la mayor varianza posible (es decir, que haya la mayor diferencia posible entre ellos para poder clasificarlos) [4]. De esta manera, se analizan sólo los componentes principales que son aquellos que permiten determinar de forma correcta y fácil a qué categoría pertenecen los caracteres analizados. En la siguiente figura se puede observar un ejemplo de reducción de dimensionalidad. Mientras que, en el gráfico de la izquierda (suponiendo que sólo haya dos categorías) el problema radicaría en encontrar el hiperplano que mejor separe los puntos pertenecientes a las dos categorías, en el de la derecha el problema se ha reducido a encontrar simplemente la línea, facilitando de esta manera el análisis. Fuera de este ilustrativo ejemplo la reducción de dimensionalidad es mucho más drástica y por lo tanto el proceso más eficiente.

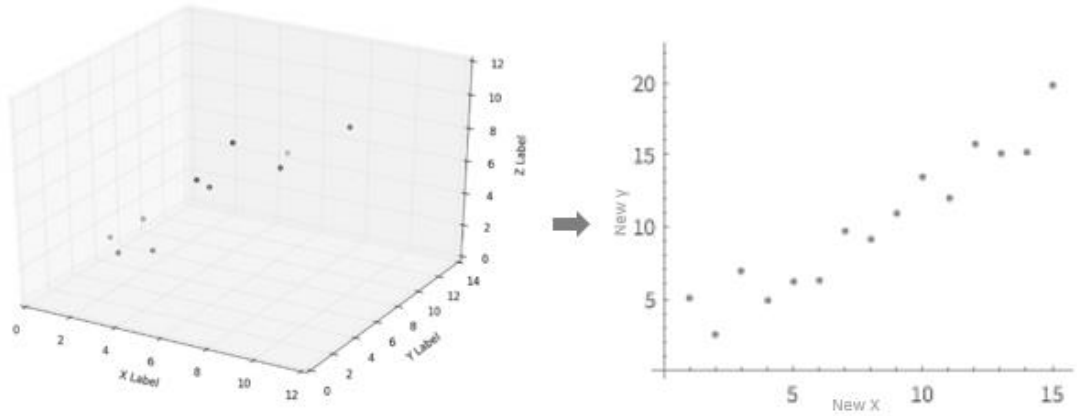

Figura 1 Reducción de dimensionalidad. [5] [6]

Linear Discriminant Analysis. En cierto punto es similar a un análisis de regresión. Se trata de encontrar el algoritmo que distingue entre distintas categorías de nuestra variable dependiente a través del análisis de nuestras variables continuas independientes. El algoritmo ha de minimizar las diferencias de los elementos pertenecientes a la misma clase y maximizarlas

con elementos de otras clases. En este trabajo [7] Jena, OP et al comprueban que el *Linear Discriminant Analysis* es ligeramente más eficiente que el *Principal Component Analysis* en el reconocimiento de números. En este otro sin embargo [8], Martínez AM y Kak AC argumentan que dependiendo del tamaño de la muestra de entrenamiento (y otros factores), el método más eficiente varía entre los anteriores.

Locality Preserving Projections. [9] Tiene la misma base que el *Principal Component Analysis*, este método transforma de manera lineal los datos originales, de manera que las distancias entre los puntos se mantienen intactas (de ahí lo de proyecciones que preservan la posición). Dicha transformación lineal, además de mantener la distancia inicial, tiene como objetivo resolver de manera óptima la asignación de categorías en base de dichos datos.

Cosine Transformation. [10] También muy usado en el procesamiento de señales y en la compresión de datos porque es capaz de transformar una serie de datos en términos de una suma de funciones sinusoidales (*cosine* en inglés, de donde deriva su nombre), resultando en datos más compactos y, por tanto, ocupen menos espacio. En nuestro ámbito, de lo que se aprovecha esta transformación es que los píxeles contiguos suelen tener alta relación y, por tanto, alta correlación. Digamos que mediante este método se extrae la variación entre píxeles contiguos, procediendo después a la categorización.

2.1.2.2 Sistemas basados en modelos

Los sistemas basados en modelos son más complejos e intentan reconstruir la imagen en 3D a partir de las diferentes vistas de las que se dispone información para después contrastarla con la base de datos de imágenes. Con los datos biométricos obtenidos de las imágenes, se intenta reconstruir una imagen que coincida con los modelos que se han construido previamente con las imágenes a partir de la base de datos (imágenes de referencia). Después se evalúa la similitud con dichas imágenes para completar el reconocimiento facial.

Los sistemas basados en modelos son más lentos en cuanto a velocidad de procesamiento y requieren tanto una mayor calidad de las imágenes como un mayor análisis previo. Sin embargo, son más robustos ante cambios en la iluminación, orientación o incluso de expresión de la cara.

2.2. Sistema de reconocimientos de carácter por imagen

En primer lugar, se expone de un modo sistemático el estado actual de los sistemas de reconocimientos de caracteres en general. A continuación, se hace un repaso de las distintas características que han de tener las imágenes para poder ser analizadas correctamente. Posteriormente se detalla un proceso de reconocimiento de caracteres de principio a fin. Por último, se detallan las metodologías más utilizadas hoy en día.

2.2.1. Reconocimiento de caracteres

En el procesamiento de un texto, es decir, para la conversión de éste a un formato procesable de ordenador (por ejemplo, UTF-8), se utilizan diferentes métodos según el texto sea manuscrito o un texto impreso (escrito por una máquina). Para los textos impresos se habla de Reconocimiento Óptico de Caracteres, u *Optical Character Recognition* en inglés (de ahí que en este trabajo se hable de OCR). Este método es relativamente más sencillo puesto que las fuentes usadas en la escritura a máquina tienen el mismo patrón.

Por otro lado, para los textos manuscritos se usan tanto sistemas de Reconocimiento Inteligente de Caracteres o *Intelligent Character Recognition* (ICR) como sistemas de Reconocimiento de Manuscrito Natural o *Natural Handwriting Recognition* (NHR).

La diferencia principal entre ellos es que mientras que el OCR y el ICR tienen un paso inicial de aislamiento individual de cada carácter, los NHR trabajan con palabras y líneas directamente. El NHR se considera un avance frente a los otros dos métodos ya que éstos exigen una fase de preprocesamiento previa que limita la eficiencia del proceso, e incluso, ante la imposibilidad de reconocer puntualmente un carácter dentro de una serie, también su eficacia.

En los subsiguientes apartados se hablará fundamentalmente del OCR por ser objeto de este trabajo. No obstante, el conocer el estado del arte en los otros sistemas se considera importante y complementario, de ahí que también se hayan mencionado, aunque de una manera más breve.

En mayor o menor medida, todos los sistemas de reconocimiento de caracteres hacen frente a los mismos problemas.

2.2.1.1 Requerimientos que ha de cumplir una imagen para su procesamiento

El contexto de la imagen. Manchas de tinta en la imagen, pequeñas anotaciones al margen, la inclusión de pequeños símbolos o dibujos, tablas, etc. pueden confundir a la máquina, y hacer pasar por texto cosas que no lo son. Cuanta más limpieza haya alrededor del texto a analizar, más tasa de éxito en el reconocimiento tendrá. Este problema es aún peor cuando se trata de reconocer

extractos de texto dentro de una fotografía o, en general, en imágenes en las que el texto es escaso y hay una gran cantidad de trazos alrededor [15].

La luz también es un factor importante. Más que haya mucha o poca, siendo los extremos siempre malos, lo importante es que esté equilibrada a lo largo de la imagen, tratando de evitarse grandes contrastes. El flash, o el que pase un rayo de luz por no cerrar bien el escáner son ejemplos claros que disminuyen la tasa de éxito.

Otro punto importante es que la imagen no esté rotada, ya que esto podría confundir a nuestra máquina. El ejemplo extremo de rotación es por supuesto el de 180º, en las que una M pasa a ser una W, pero no hace falta irse tan lejos para provocar distorsiones que menoscaben la tasa de éxito.

La aparición de una tercera dimensión (la profundidad) bien sea por mala calidad del sensor, o por movimiento del texto en el momento de conseguir la instantánea, puede devenir en problemas o incluso en duplicación de alguna línea, lo que puede confundir a nuestra máquina y reducir la tasa de éxito.

2.2.1.2 Reconocimiento Óptico de Caracteres - OCR

El Reconocimiento Óptico de Caracteres (OCR) tiene unas fases comunes sea cual sea el algoritmo a utilizar. Algunas de ellas son compartidas también por el ICR y NHR. Estas fases tienen dos momentos, el primero, en el que se entrena al algoritmo con una muestra de referencia, y el segundo, el cual constituye el Reconocimiento Óptico de Caracteres en sí, en el que se analiza una imagen objetivo y se les asigna a las clases previamente establecidas a partir de la muestra de reconocimiento [16].

a. **Preprocesamiento**. Consiste en limpiar o despojar al texto en cuestión de contenido innecesario. Básicamente trata de eliminar o reducir al máximo todos los problemas que se han mencionado en el apartado anterior.

b. **Segmentación**. Es el proceso por el cual se van aislando los componentes de la imagen de lo que es el fondo. Primero se separa cada línea, después cada palabra y por último cada carácter. Una vez segmentados, se procede a clasificar cada componente, siendo este un paso crucial del proceso.

   Dentro de la segmentación se pueden distinguir varios tipos de enfoques, siendo el *Top-Down* el más común, aunque también hay modelos *Bottom-Up* e híbridos. En los últimos tiempos ha habido un auge en la literatura en cuanto al método óptimo para segmentar, siendo una de las propuestas que mayor éxito ha reportado la de Shinde, A. A. y Chougule, D. G., [17](98%). En ella se mezclan métodos de proyección horizontal y vertical.

c. **Normalización**. En esta fase se ordenan todos los caracteres que ha sido posible extraer en la fase de segmentación y se preparan según el algoritmo a utilizar. En esta fase toda la información no necesaria ha sido filtrada.

d. **Obtención de parámetros clave**. Una vez que tenemos preparados todos los caracteres obtenidos a partir de la imagen, llega el momento de calcular los parámetros que utilizaremos para asignar nuestras imágenes. Aunque esta fase de obtención de parámetros clave, cronológicamente se realiza antes que la clasificación, el método para clasificar cada imagen habrá que elegirlo de antemano, ya que los parámetros a obtener varían según los diferentes métodos de clasificación. Para el método a elegir, cuando lo que se trata de reconocer es un texto con un alfabeto como base, hay una extensa literatura (por ejemplo: Sharma, O. P. et [18]).

   Sin embargo, en este trabajo se introduce la novedad de que lo que se trata de obtener no es el texto en sí, sino descifrar el estilo caligráfico utilizado. Con la dificultad añadida que ya se comentó previamente, de que no se trata de un alfabeto con un número de caracteres limitado.

e. **Fase de clasificación**. En esta fase se comparan los parámetros obtenidos con los que se obtuvieron en la muestra de referencia, asignando así las clases mediante un algoritmo de decisión. La diferente tipología de los algoritmos de decisión se trata en otro apartado de este trabajo, por no constituir propiamente una fase. Nos referimos, por señalar los más importantes, a Redes Neuronales (o *Neural Network*), Comparación de plantillas (*Matching templates*) [19], Máquinas de vector soporte (o *Support Vector Machine*) [20] y otras técnicas estadísticas. Además, existe la posibilidad de combinar dos o más de estos métodos.

f. **Post-procesamiento**. Esta fase consiste en realizar una evaluación de los resultados obtenidos siempre que sea posible. Esta evaluación, en caso de realizarse, servirá para mejorar la clasificación ya que lo que en realidad se estaría haciendo es incrementar la muestra de referencia.

2.2.2. Métodos para la clasificación

Redes neuronales (*neural network*)

Este método recibe el nombre porque funciona como una imitación de la red de neuronas biológicas en las que – simplificando mucho el proceso - las neuronas van transmitiendo los impulsos nerviosos de una a otra, pero la información que transmite cada una de ellas no es la misma que recibió, ya que cada neurona realiza un proceso de simplificación, de manera que la información neuronal se va depurando a su paso por la red.

Toda red neuronal contiene una capa de entrada, en la que se reciben los datos reales tras su normalización, y una de salida, que es el output de la red. Además, suelen tener lo que se denominan capas ocultas (con un número variable). El *Deep Learning*, de hecho, se basa en la utilización de un gran número de capas.

Mediante la muestra de referencia, en la que el diseñador conoce el resultado a obtener, se va enseñando a la red neuronal de manera que, mediante la transmisión de información entre cada una de las neuronas, se optimice la tasa de acierto. El equivalente a los impulsos en el sistema neuronal del ser humano son los pesos y parámetros que se le aplican a los datos iniciales. Estos parámetros se irán reajustando a medida que se aumente la muestra de referencia para, como ya se ha dicho, maximizar la tasa de éxito del sistema de clasificación.

En la imagen se muestra el funcionamiento de una red neuronal con tan solo una capa oculta.

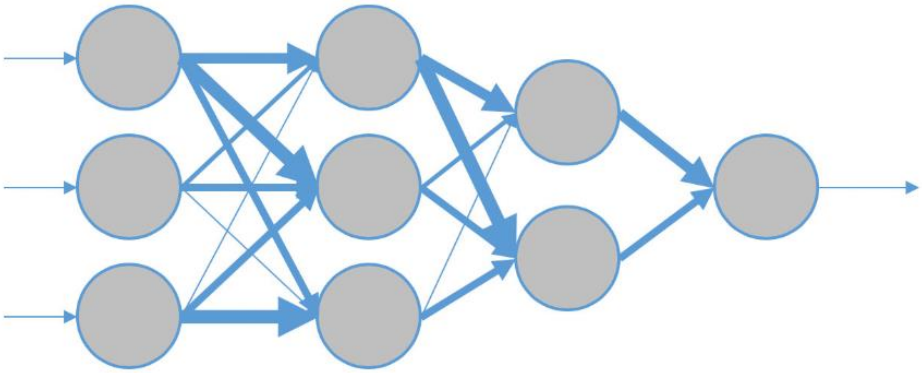

Figura 2 Ejemplo de red neuronal con dos capas ocultas. [21]

Mediante el *Deep Learning* o Aprendizaje Profundo, se realiza un proceso de ajuste automático – que hoy se conoce como aprendizaje automático – por el cual se van realizando transformaciones no lineales a las variables iniciales. Estas transformaciones tienen lugar en las flechas de la imagen, teniendo la capa de la que parte la flecha los datos iniciales y la siguiente los datos modificados. Dentro del *Deep Learning* las características de nuestros datos, es decir, los parámetros a estudiar tienen diferentes jerarquías, siendo las jerarquías superiores las que contarán con un mayor peso a la hora de producir un output (en este trabajo el output es la asignación de la imagen a un estilo caligráfico).

Este aprendizaje o ajuste continuo puede ser supervisado o no supervisado. La diferencia fundamental entre ambos es que en el supervisado se tiene un mayor conocimiento de las variables input y se sabe el output concreto que el algoritmo debería arrojar, mientras que en el no supervisado las variables son tratadas como aleatorias y no se conoce el resultado final, con lo que básicamente se agrupa en clústers similares cada una de las observaciones, maximizando la diferencia entre objetos de distintos clústers y minimizándola entre objetos del mismo clúster.

Comparación de plantillas (*template matching*) [19]

Quizás es el método más intuitivo a simple vista, pero es el que también contiene una mayor probabilidad de error. Consiste en, a través de la captación de píxeles, formas y curvaturas, comparar con las plantillas (muestra de referencia) y detectar con cuál tiene más semejanzas. Como se ve, este método es muy sensible a posibles distorsiones y es el menos sofisticado en cuanto al algoritmo de análisis.

Máquinas de vector soporte (*Support Vector Machine*) [20]

Dado un número finito de características y un número también finito de categorías, los vectores soporte son creados a partir de un algoritmo supervisado que, tras tener en cuenta las características, separa por hiperplanos cada categoría. Para explicarlo de manera sencilla, el algoritmo crea unas fronteras que delimitan la categoría a la que un objeto pertenece en base a las categorías recogidas. Es supervisado porque necesita la muestra de entrenamiento para crear las fronteras.

Intuitivamente se puede pensar en el caso en el que sólo hay una dimensión con la que categorizar (imagínese el lector puntos distribuidos a lo largo de una recta). En este caso, el estar entre unos determinados puntos de esta línea determina la categoría.

Otros métodos estadísticos

De entre el resto de los métodos estadísticos, y con el objetivo de hacer una mención breve de aquellos más destacables, destacan el análisis de *clustering*, y el criterio del vecino más cercano. Estos métodos pueden combinarse entre sí o complementar a otros métodos como criterio adicional.

El criterio de vecino más cercano, o de *nearest neighbour* [22] por su nombre en inglés, se basa en buscar dentro de la muestra de referencia al objeto con el que haya una mayor similitud en los parámetros. Puede ser combinado con otros métodos y depende en gran medida de la decisión sobre el método para calcular la similitud.

El análisis de *clústers* o grupos consiste, como ya se ha mencionado en otros apartados, en detectar las diferentes características que suelen darse en ciertos objetos, con el objetivo de separarlos en grupos. En el momento en el que el algoritmo detecte que un objeto tiene unas determinadas características, lo asigna a un grupo.

2.3. Sistema de reconocimientos de estilos caligráficos

Primeramente, se expone de un modo sistemático el estado actual de los sistemas de reconocimientos de estilos caligráficos en general. Después se hace un repaso de los distintos estilos caligráficos dentro del idioma chino; a continuación,

se analiza un software similar para completar el estado del arte y poder establecer comparaciones. Por último, se enlazan ambos temas y se da una pincelada de la situación del reconocimiento de estilos caligráficos chinos.

Pese a que la literatura y la investigación existente se centra más en el reconocimiento de caracteres, el presente trabajo se centra en el reconocimiento de estilos caligráficos, más concretamente en los estilos caligráficos chinos.

El análisis de caracteres y el de estilos caligráficos tiene similitudes y diferencias, ventajas o inconvenientes. Los pasos iniciales son los mismos, puesto que, a partir de una imagen, bien sea escrita a mano -trazo más libre- o bien haya sido generada por ordenador -homogeneidad- se trata primero de identificar cada uno de los trazos y particularidades de la imagen, para después categorizarla.

Mientras que en el reconocimiento de caracteres es fundamental la detección individual, el reconocimiento del estilo caligráfico cuenta con una cierta ventaja: no se trata de determinar el carácter individual sino el estilo del conjunto. En este sentido, cuanta mayor extensión tenga el texto y más claro sea el trazo, más posibilidades de éxito.

Por otro lado, muchas veces la diferencia entre diferentes estilos no es grande ni clara, principalmente en la escritura a mano. Como se detalla más adelante en el trabajo, entre la caligrafía corriente y la cursiva china la identificación puede llegar a ser muy difícil. Algunos estilos sólo se diferencian en ciertos trazos, haciendo imposible identificarlos si éstos no aparecen en el texto que se quiera analizar.

Dentro de la literatura actual que trata sobre el análisis de estilos caligráficos, el de la escritura china - del que se da cuenta en este trabajo- ha generado un gran interés. En ello radica tanto el gran número de personas que lo usan, como el fuerte arraigo tradicional. En este texto se ha usado como referencia el trabajo de Chen, Y.S. et al [23], ya que en él se hace un resumen bastante completo de los diferentes métodos y además se circunscribe, como el presente trabajo a la escritura china.

2.3.1. Métodos comunes

Como en el apartado de reconocimiento de caracteres por imagen, este trabajo se centra en los métodos de categorización, una vez que el proceso de extracción de la imagen se considera realizado. Es decir, las características de los píxeles, sus diferencias, etc. se encuentran ya debidamente preparados para el análisis. Todos los métodos de obtención y optimización de parámetros que en adelante se detallan, necesitan – como ya se comentó en apartados previos – una muestra previa o inicial de entrenamiento y calibración.

Función softmax [24]

De una manera simplificada, se trata de una regresión logística - en nuestro caso multinomial ya que hay cinco posibles estilos caligráficos a reconocer –. Analizando los datos obtenidos de la imagen, esta función asigna la probabilidad de que este texto pertenezca a cada uno de los estilos.

Máquina de vector soporte (*Vector support machine*)

Método también estudiado a propósito del reconocimiento óptico de caracteres; consiste fundamentalmente en separar la muestra en tantas categorías como categorías existan para el propósito (en el presente trabajo, cinco estilos caligráficos), de manera que la tasa de éxito conjunta sea la mayor posible. Si para simplificar pensamos en tres dimensiones y dos categorías, el vector soporte sería el hiperplano que separa ambas categorías.

k-Vecinos más cercanos [22]

Es un método no paramétrico. Consiste en comparar el texto a estudiar con la muestra de entrenamiento y comprobar las diferencias. Tras agregar todos los datos en un conjunto, se establece un número k de elementos de la muestra de entrenamiento (los k elementos más parecidos respecto al que está bajo estudio), y el estilo caligráfico prevalente en dichos elementos será el que determine la categoría del elemento estudiado. En la siguiente figura, los triángulos y los cuadrados son las dos categorías de la muestra de entrenamiento ordenadas en un eje bidimensional; el punto es el elemento del cual tenemos que determinar su categoría. El círculo de trazo continuo significa elegir k=3, mientras que el círculo de trazo discontinuo k=5. En este ejemplo, la elección de un k u otro haría diferente la categoría obtenida. Con un número mayor de k se tienen más objetos con los que comparar, pero la distancia es mayor.

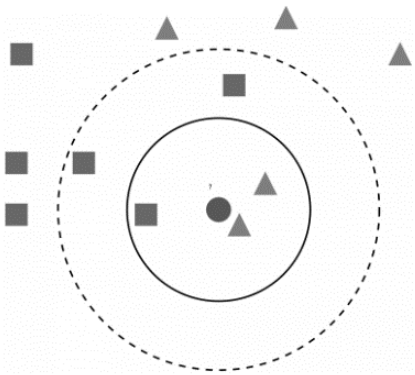

Figura 3 k-vecinos más cercanos. [25]

Random forests [26]

La idea de este método es, mediante la utilización de un número de árboles (modelos) que simbolizan conexiones entre los datos, asignar una categoría por cada árbol (modelo) y después hacer un promedio. El modelo de cada árbol es diferente, y aunque suelen contener mucho ruido, el promedio posterior y la utilización de varios se encarga de reducirlo. Está considerado como uno de los métodos con más tasa de acierto cuando la muestra de entrenamiento es grande, y, si como pasa en el presente trabajo, no hay diferentes niveles dentro de una categoría.

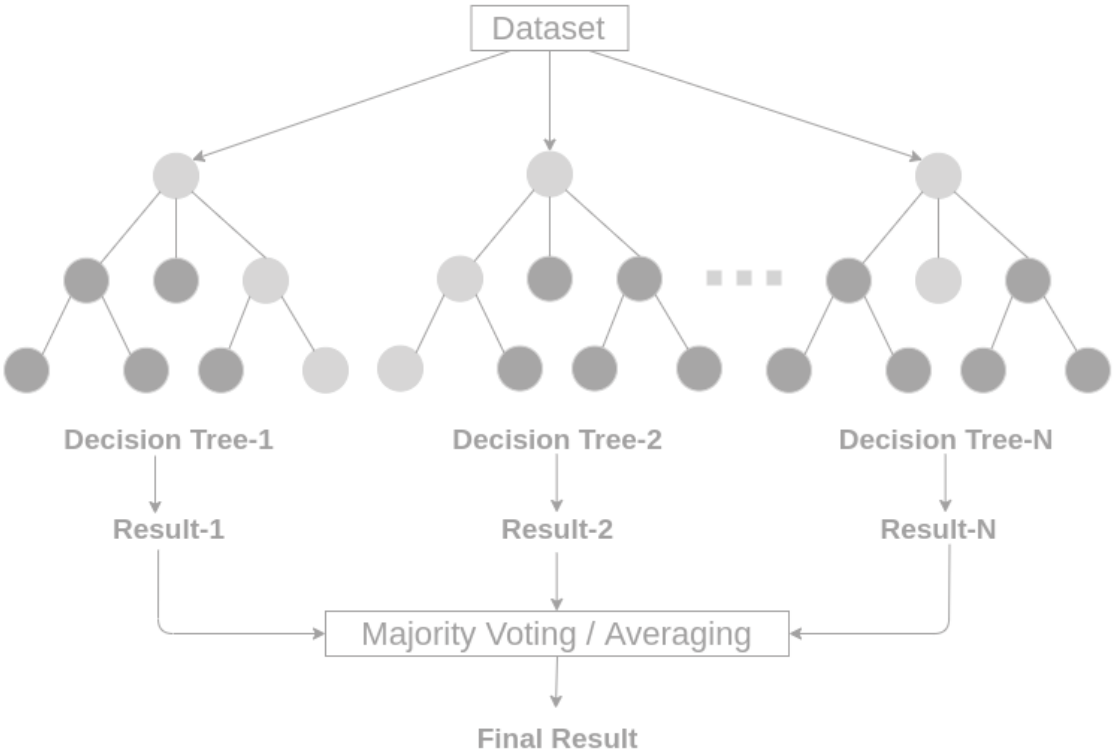

Figura 4 Random forest. [27]

2.4. Estilos caligráficos chinos

Caligrafía es una palabra procedente del griego cuyo significado originario es "escritura hermosa" (a veces también se puede encontrar traducido como "grabado hermoso"). Dejando a un lado perspectivas subjetivas, es cierto que la escritura china en general pone un gran cuidado en la manera de trazar cada uno de los caracteres.

Cabe indicar que, a diferencia de la escritura latina basada en una serie finita de letras denominada alfabeto, la escritura china está basada en caracteres que pueden formar palabras uniéndose a otros o por sí mismos. El hecho de que el número de caracteres sea muy grande podría dificultar (y de hecho dificulta) la detección de

cada uno de los estilos caligráficos. Sin embargo, tal y como se explica más adelante, se podrán diferenciar por las pautas de trazo de cada uno de ellos.

Con independencia del dialecto y caracteres usados, dentro de lo que se denomina escritura china pueden distinguirse 5 estilos caligráficos diferentes.

Escritura de sello (o de estilo Zhuan)

Es el tipo de escritura más antigua, y de hecho tiene su origen en los grabados hechos en piedra más que en la escritura en papel o papiro de por sí. Es por ello por lo que las trazas de este estilo son más libres, siendo las pautas algo menos genéricas con respecto al resto de estilos basados en la escritura con pincel y tinta.

Incluso dentro de este estilo se distinguen dos tipos, el sello grande y el sello pequeño. El sello grande fue el primero en aparecer a partir de los conocidos como caracteres arcaicos. El pequeño sello fue una estandarización del gran sello.

Las características principales de este estilo son los siguientes:

- Difícil de leer y escribir.
- Emplea líneas finas y puntiagudas en los extremos.
- Intenta cubrir toda la superficie asignada al carácter.
- Los caracteres se separan de forma uniforme tanto vertical como horizontalmente.

Figura 5 Escritura de sello o Zhuan

Estilo de los escribas (o estilo Li)

La administración de un imperio requiere de una escritura más homogénea, y a pesar de que el pequeño sello había resultado una estandarización del gran sello, no era suficiente. Para ello se procedió a una simplificación, que recibe el nombre de “estilo de los escribas” debido a que eran principalmente los funcionarios los que la usaban.

Las características principales del estilo Li son los siguientes:

- Utilización de un trazo horizontal ondulado.
- Empleo de un trazo horizontal inferior grueso que da una sensación de estabilidad.
- Frecuentemente alineados en la parte superior.
- Escritura rápida.
- Margen entre caracteres de forma regular.

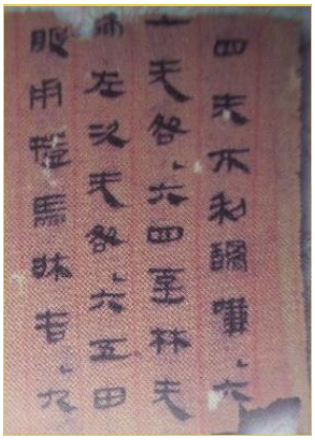

Figura 6 Escritura de los escribas o Li

Estilo regular (o estilo Kai)

Los calígrafos fijan definitivamente unas reglas claras de trazo, simplificando los ángulos y líneas, manteniendo un conjunto de trazos llamados fundamentales en los cuales se basa el resto del sistema, aunque manteniendo los puntos principales establecidos en los estilos mencionados previamente. Es más legible, y debido a ello ha permanecido prácticamente inalterado durante largo tiempo.

Las características principales del estilo regular son los siguientes:

- Caracteres más estructurados y cuadrados.
- Velocidad de escritura moderada.
- Separación entre trazos.
- Ofrece gran nitidez y legibilidad.

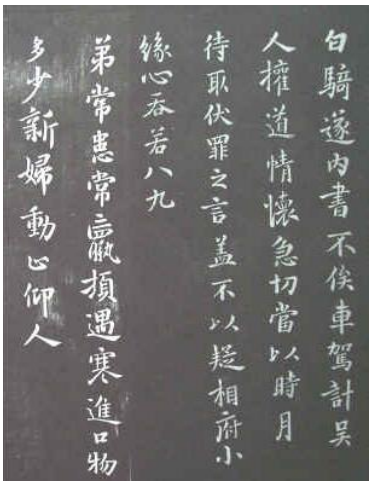

Figura 7 Escritura estándar o Kai

Estilo corriente (o estilo Xing)

Aunque el nombre de este estilo ha sido traducido en diversas formas, se trata de una deformación del estilo regular, que se produce casi de manera natural al escribir cotidianamente. Es decir, se trata del estilo regular pero modificado por el uso para hacer la escritura más rápida (de ahí la traducción escogida de "corriente").

Sus características principales son las siguientes:

- Trazos continuos y suaves.
- Es muy común su empleo en los apuntes y las notas personales.
- Velocidad de escritura rápida.
- Simplifica trazos.

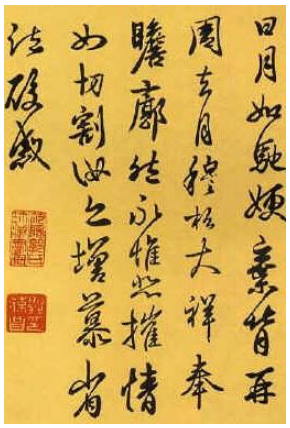

Figura 8 Escritura corriente o Xing

Cursiva (o estilo Cao)

Aunque este estilo tiene diversidad de nombres, en el presente trabajo se ha optado por la denominación de "cursiva", que tiene su paralelismo en el sistema caligráfico latino. Otros nombres que recibe son "escritura loca" o incluso

“escritura de hierba”. Es una escritura desarrollada muy rápidamente por lo que carece de propósito estético y puede ser a veces difícil de entender. Siguiendo con los paralelismos, sería algo parecido a lo que coloquialmente se conoce como “escritura de médico” en España.

Las características de este estilo son los siguientes:

- Se emplean pocos segundos para escribir un carácter.
- La mayoría de los trazos necesarios para formar un carácter se escriben de forma continua.
- Los márgenes entre caracteres son muy irregulares.
- Existen casos donde el carácter se simplifica de forma radical empleando Taquigrafía.

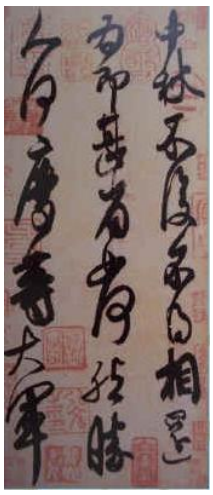

Figura 9 Escritura cursiva o Cao

A pesar de las diferencias existentes entre los diferentes estilos caligráficos, cuya detección es el objeto de este trabajo, hay una serie de normas inherentes a todos los estilos: los trazos básicos, el orden en el que se perfilan cada una de las líneas (horizontal – vertical, derecha – izquierda, arriba - abajo) y el hecho de que tengan que estar contenidos dentro de un cuadrado no trazado (también llamado imaginario).

2.5. Software para el reconocimiento de imágenes con texto en chino

Para completar el estado del arte, se pasa a señalar y describir brevemente un software similar en cuanto a la funcionalidad que este trabajo propone, aunque con propósitos distintos.

El software en cuestión es *EasyScreenOCR*[1], y el funcionamiento es el siguiente: el usuario sube entre 1 y 5 imágenes con texto en chino (formato imagen) al servicio web, y en menos de 30 minutos obtiene un documento de texto. Además del servicio web, también puede ser descargado para Mac, Windows, Android o

[1] https://easyscreenocr.com/

iOS, y tiene la capacidad de transformar texto en más de 100 idiomas, incluido por supuesto el chino.

La diferencia fundamental que aporta el software que propone este trabajo es que lo que detecta es el estilo caligráfico en lugar de transformar el texto. Esta diferencia es importante porque en la escritura china se da una gran importancia al dibujo y la adhesión a una serie de normas.

En el caso en el que superase al ojo humano en tasa de éxito, este software podría ser utilizado para analizar imágenes con textos chinos antiguos y poder así datar correctamente la pervivencia de cada estilo.

No hay que olvidar que, aunque hoy se use mayoritariamente el chino simplificado (en la China continental) y en algunas áreas chino tradicional (sólo es usado en Hong Kong, Taiwán y Macao), en otras épocas la escritura china tenía una caligrafía muy diferente, que hoy se ha recuperado precisamente por la escritura a máquina.

3. Definición del proyecto

3.1. Metodología

En este punto se definirán tanto el detalle como el orden de ejecución de cada una de las etapas que se han seguido durante el desarrollo de este proyecto. Las fases del proyecto coinciden con las que se emplean en la mayoría de los proyectos de software tradicionales, siguiendo un desarrollo secuencial como define el Modelo en Cascada retroalimentada:

a. **Análisis**: En esta fase las tareas principales son la recogida de requisitos y el estudio de mercado para ver si ya existe alguna solución similar al problema a tratar.

b. **Diseño**: Esta etapa está dedicada a realizar tanto el diseño del modelo de predicción como la aplicación web aplicaciones. Esto incluye, por parte de la web: el diseño de pantallas, navegación y protocolo de comunicación. Por la parte de modelo se define la estructura de la red neuronal a utilizar.

c. **Desarrollo**: En esta etapa es en la que más esfuerzo y tiempo se ha invertido dentro del presente trabajo. En ella se codifican los diseños definidos en la fase anterior, se realiza el proceso de entrenamiento y se procede a depurar la red neuronal.

d. **Integración y pruebas**: En esta etapa se complementan las pruebas realizadas en la fase de desarrollo (que iban encaminadas a clasificar los elementos con el menor número de errores posible). Las pruebas de esta etapa tienen como objetivo fundamental comprobar la integración del desarrollo para asegurar la funcionalidad de la aplicación. Adicionalmente se lleva a cabo la validación cruzada de la red entrenada para corroborar su efectividad ante muestras independientes. Tras finalizar todas las pruebas correctamente, se ponen en marcha tanto la página web como la API en los entornos habilitados.

e. **Documentación**: En las etapas anteriores se ha ido realizando un gran trabajo de documentación, pero es en esta fase cuando se ha dado formato a todo lo recogido para tratar de crear una memoria lo más detallada y explicativa posible. En ella se recogen los aspectos más importantes del proyecto y se establece la relación entre los elementos de cada una de las fases. Finalmente, el documento incluye posibles mejoras o aspectos a tratar más en detalle en un trabajo futuro, así como conclusiones obtenidas en el proyecto.

f. **Mantenimiento**: La delimitación de la fase de mantenimiento es difícil de estimar, ya que depende de múltiples factores. Durante esta fase se mejora, actualiza y/o modifica aquello que se considere necesario en la página web

o en la API, a partir de la retroalimentación obtenida en cada una de las diferentes fases anteriores. En el caso de este proyecto, la fase de mantenimiento se centra fundamentalmente en los servicios de la API ofrecida, así como en el mantenimiento de la página y el dominio. Adicionalmente, esta fase también tiene como objetivo la mejora constante de la red neuronal a partir del incremento del número de inputs con los que entrenarla.

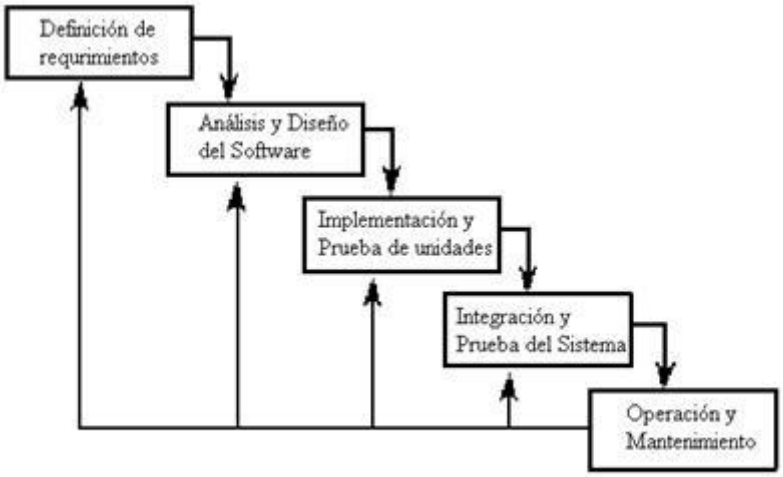

Figura 10 Ciclo de vida - Cascada retroalimentada

3.2. Ciclo de vida de un modelo de Machine Learning

Dentro de cada fase de la metodología en cascada retroalimentada aplicada en este proyecto, podemos hacer un énfasis en cómo se va creando el modelo de clasificación y sus diferentes estados:

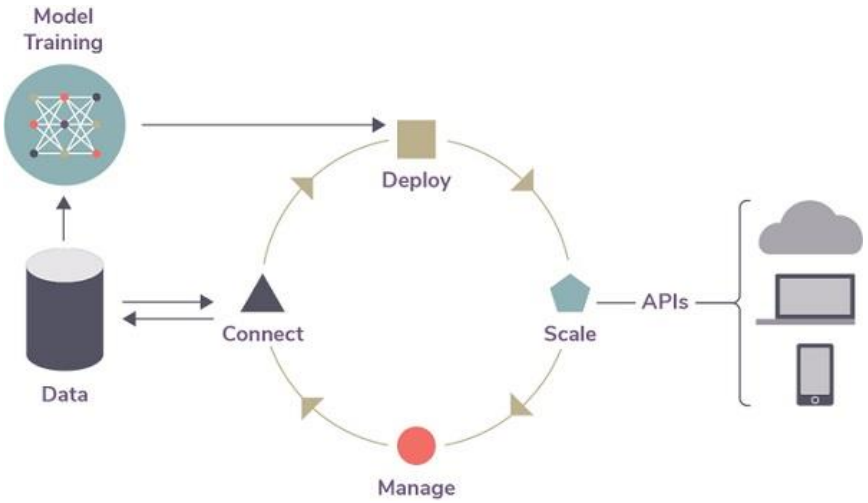

Figura 3-2 Ciclo de modelo ML [28]

- ***Model Traning***: Esta fase corresponde a la etapa inicial del proyecto (Análisis, Diseño y desarrollo). Las distintas tareas llevadas en esta fase son la elección de algoritmo a aplicar como la recolección de datos para realizar el entrenamiento de ésta.

- ***Deploy***: Una vez validado que el modelo resultante de la fase anterior cumple con las expectativas reales. Se despliega el modelo en entornos productivos.

- ***Scale***: El modelo desplegado será utilizado en los distintos programas requeridos.

- ***Manage & Connect***: Hay que monitorear el funcionamiento de los modelos desplegados para refinar o reentrenar si fuese necesario.

4. Arquitectura del sistema

Para proporcionar a los usuarios una forma sencilla de interactuar con la red neuronal entrenada, se desarrolla una página web. Esta página permitirá a los usuarios subir una imagen (sobre la que es posible realizar recortes), de la que se extraerán caracteres para su posterior evaluación. Por ello, se ha implementado una solución completa de Cliente – Servidor que nos permite separar desde el punto de vista lógico las tareas a realizar en cada capa.

El sistema de intercambio empleado en la comunicación de la interfaz y servidor es Notación de Objetos de JavaScript (JSON)[2]. Este formato nos permite intercambiar los datos de forma sencilla, ligera y compacta, ya que resulta fácil de tratar tanto para los programadores como para la máquina.


```
{
    "final_result": "Estilo_Cao",
    "details": {
        "Estilo_Cao": 0.347,
        "Estilo_Kai": 0.269,
        "Estilo_Li": 0.0454,
        "Estilo_Xing": 0.3353,
        "Estilo_Zhuan": 0.0032
    },
    "CN": [
        "人",
        "和",
        ""
    ],
    "EN": [
        "people",
        "with",
        ""
    ]
}
```


Figura 11 Formato JSON

Primeramente, se recorta cada carácter de manera individual, de tal manera que cada imagen contiene un carácter. Después, aplicando el sistema de intercambio de información JSON, las imágenes viajan en formato codificado de base 64. Este formato consiste en codificar la imagen a nivel byte para convertirla en una representación de texto.

Tanto la página como los servicios web utilizan el protocolo de Transferencia de Hiper-Texto (HTTPS). que básicamente es una mejora en cuanto a la capa de seguridad del protocolo HTTP. Aunque actualmente nuestro sistema no requiera de autenticación por parte del usuario, no se descarta su incorporación en un futuro para mayor iteración y más modelos de predicción. Este protocolo nos proporciona una conexión segura entre el servidor y el cliente que no puede ser interceptada por personas no autorizadas. Para ello se utiliza un certificado SSL emitido por una autoridad reconocida. En este caso se ha empleado la certificación de *Let's Encrypt*[3], una autoridad de certificación gratuita, automatizada, abierta y sin ánimo de lucro.

[2] https://www.json.org/json-es.html
[3] https://letsencrypt.org/es/

El cliente realizará peticiones al servicio web y recibirá la respuesta a dicha petición. El servidor recibe y procesa la petición, y envía la respuesta, mostrando los resultados de una manera que resulta intuitiva al usuario.

Al carecer de la funcionalidad de registro de usuario, no existe la necesidad de establecer una base de datos de usuarios en este proyecto. La necesidad de almacenar información sobre la corrección de estilo caligráfico se lleva a cabo mediante el uso del propio sistema de ficheros del servidor remoto Linux.

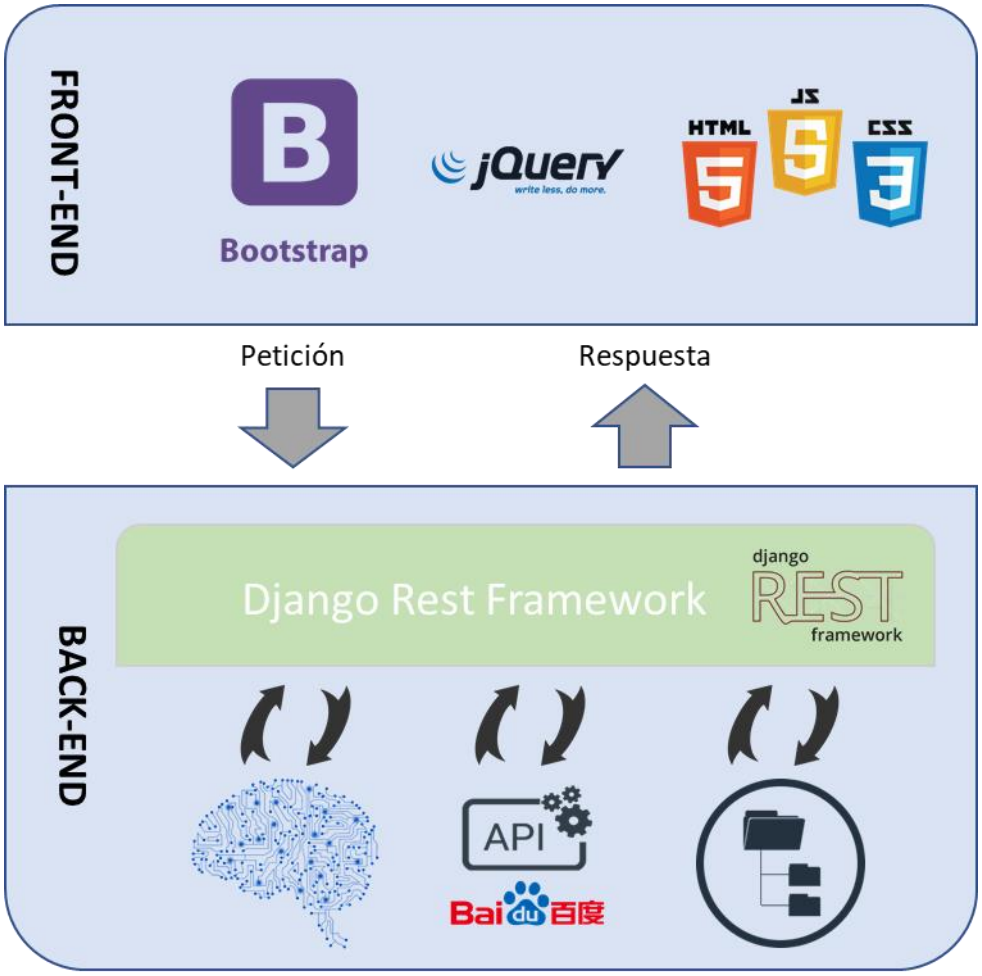

Figura 12 Arquitectura modular completa

Toda esta arquitectura modular completa se construye a partir de la base *Representacional State Transfer* (REST) descrito por Roy Fielding en los años 2000 [29], es un estilo arquitectónico que provee estándares entre el sistema informático en la web, facilitando la comunicación entre éstas.

En el proyecto concretamente se implementa un sistema **RESTful**, caracterizado por renunciar los estados y separar las preocupaciones del cliente y del servidor de forma independiente.

4.1. Características principales

De forma muy resumida, se puede concluir que las ventajas de esta arquitectura son los siguientes:

Ambas capas están muy débilmente relacionadas. El cliente y el servidor son sistemas independientes y separados físicamente, que solamente intercambian solicitudes de servicios y respuestas en JSON. Al cliente le es trasparente el funcionamiento interno del API y al servidor le es trasparente el manejo o representación de los datos proporcionados mediante respuestas.

Protocolos asimétricos y recursos. Entre cliente y servidor se establece una relación de muchos a uno. Esto se traduce en que el servidor tendrá que atender a muchos clientes de forma simultánea.

Independencia de tecnologías y lenguajes. El desarrollo de ambas capas no está ligado. Esto permite que se puedan desarrollar los distintos agentes según necesidad o de la forma que resulte más cómodo a la hora de trabajar. Cabe destacar que podemos utilizar el mismo servidor para distintas aplicaciones de distintas plataformas como: aplicaciones Android, aplicaciones iOS, páginas web o incluso ser integrada para otras APIs.

Gran escalabilidad, flexibilidad y fiabilidad. Por parte de la API se pueden implementar nuevas funcionalidades o mejorar las existentes de forma incremental, sin que ello tenga impacto en el otro agente web. Además, se puede aumentar o migrar a otros servidores más potentes, de manera que se pueda proporcionar servicios a más clientes al mismo tiempo y de forma más rápida. A esto se le puede sumar la escasa importancia de la ubicación, un servidor puede estar en el mismo equipo que el cliente o en otra red mientras que el dominio esté bien configurado.

Experiencia del usuario. Las respuestas y peticiones entre la página web y el servidor son datos planos en formato JSON, por lo que el tiempo de transferencia es mucho menor que la petición de una página completa.

Protocolo sin estado. Esto significa que la comunicación entre el cliente y el servidor siempre contiene toda la información necesaria para realizar la solicitud. No hay estado de sesión en el servidor, se mantiene completamente del lado del cliente. Si el acceso a un recurso requiere de autenticación, entonces el cliente necesita autenticarse con cada solicitud.

Almacenamiento en caché. El cliente, el servidor y cualquier componente intermediario pueden almacenar en caché recursos para mejorar el rendimiento ya que la memoria caché ofrece mayor velocidad de lectura.

Se define uniformemente todas las operaciones. Esto simplifica la arquitectura, ya que todos los componentes siguen las mismas reglas para hablar entre ellos. También facilita la comprensión de las interacciones entre los diferentes componentes del sistema. Se requieren varias restricciones para lograr esto. Se tratan en el resto del capítulo.

5. Tecnología aplicada y su implementación

Como ya se ha explicado en la revisión de la literatura actual, el campo en el que se mueve este proyecto es un campo en continuo crecimiento y en el que se están produciendo muchos avances, tanto desde el punto de vista de la modelización, como el del manejo más eficiente de los servidores y el aumento de la capacidad, sin olvidarnos de las funcionalidades que mejoran y facilitan la interpretación y el manejo del usuario.

A continuación, separando por el preprocesamiento de la imagen, el modelo (red neuronal artificial), back-end y front-end, se detallan las tecnologías y métodos elegidos para la implementación del proyecto. Se podría considerar el apartado más importante del proyecto, ya que se centra en los detalles técnicos de cada uno de los componentes, y permite exponer todo lo aprendido a lo largo del proyecto.

5.1. Algoritmo de segmentación

Para poder ofrecer al usuario una mejor experiencia de uso, se ha empleado un conjunto de algoritmos de preprocesamientos básico para la segmentación de una imagen de caracteres en sinogramas individuales. Hay que matizar que el alcance de este algoritmo está enfocado a escrituras no conexas, lineales y sin mucho ruido o elementos que no son caracteres en el input.

La mayoría de las funciones utilizadas en este apartado, están embebidos en la librería libre de OpenCV[4], una de las librerías más populares en el ámbito de visión artificial desarrollado en C++.

Los pasos de preprocesamiento son los siguientes:

Binarización, umbralización y transformación

Primero se transforma las imágenes a formato binario. Para ello, la imagen se lee en forma de escala de grises y, a continuación, se aplica la umbralización OTSU [30]. Este algoritmo hace honor a su creador Nobuyuki Otsu, quien ha empleado la varianza de la dispersión de los niveles de grises para encontrar un valor de umbral que minimice la varianza de pesos entre clases.

[4] https://opencv.org/

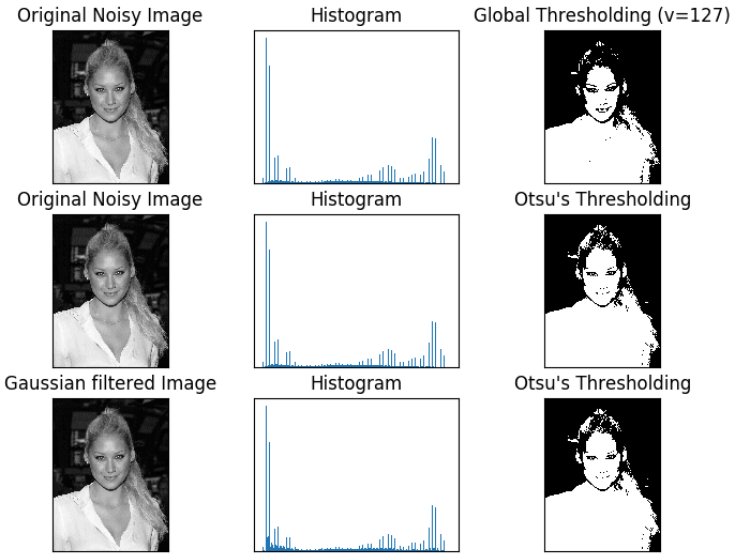

Figura 13 Umbral de Otsu en el procesamiento [31]

Debido a que los caracteres suelen formarse con trazos muy finos, conteniendo muchos huecos y separaciones irregulares entre diferentes trazos, se aplica una operación morfológica de cierre con el objetivo de superar este impedimento y así poder diferenciar cada carácter. Esta operación consiste en una combinación de dos operaciones morfológicas fundamentales: dilatación seguida de erosión.
Dilatación consiste en definir un núcleo de tamaño predefinido y recorre los píxeles de la imagen original de forma completa, transformando aquellos grupos de píxeles a 1 cuando uno de los píxeles del contenido/núcleo es igual a 1 (reducir ruido), en el caso de erosión, se ha de cumplir que todos los píxeles del núcleo tengan el valor 1 para obtener 1 como resultado de la operación.
Aquí $\oplus$ y $\ominus$ indican la operación de dilatación y erosión:

$$A * B = (A \oplus \mathrm{B}) \ominus \mathrm{B}$$

Figura 14 Operación de cierre

Método de segmentación en líneas

Tras adaptar la transformación donde se disponen de una serie de caracteres con menos trazos finos y en forma de bloque, tiene lugar el proceso de segmentación horizontal de la imagen. Se emplea un método de escaneo de líneas sobre las imágenes binarias donde los caracteres tienen el valor 1 con 0 como fondo. La iteración se realiza por cada fila, analizando si la fila está vacía o no. De tal forma que, una fila está vacía cuando todos los valores de los píxeles de la fila son 0. Véase la siguiente ecuación para encontrar el conjunto ordenado de filas no vacías. dónde f (*x, y*) es la imagen, y(*i*) se refiere al vacío de esa fila. Por otro lado, Θ (*i, j*) se refiere al vacío de la columna *j* de la fila *i*

$$\in (i) = \{1, if\ \forall_j f(i,j) = 0; 0, en\ otros\ casos\}$$

$$\theta([i_1, i_2], j) = \{1, if\ \forall_{j \in |i_1, i_2|} f(i,j) = 0; 0, en\ otros\ casos\}$$

Una vez obtenidas las filas no vacías, se procede a crear los rectángulos basados en columnas.

$$Foreach\ \in (i)\ i_1 =$$
$$first(\in (i))i_2 =$$
$$last(\in (i))rects =$$
$$\forall_{|i_1,i_2|, j \in \min(domain(f), \max(domain(f)))} \theta([i_1, i_2], j)$$

Método de localización de contornos

Después de aplicar las transformación y segmentación en líneas individuales, se puede recurrir a varias técnicas de visión artificial para obtener los cuadros delimitadores de cada sinograma. Tras de probar varios métodos, se opta por el algoritmo *Topological Structural Analysis of Digitized Binary Images by Border Following* de Suzuki, S. y Abe, K. [32] que se encuentra implementado en OpenCV para encontrar los contornos de una imagen binaria. Después de aplicar este algoritmo, podemos obtener una lista de cuadros delimitadores como la figura 15.

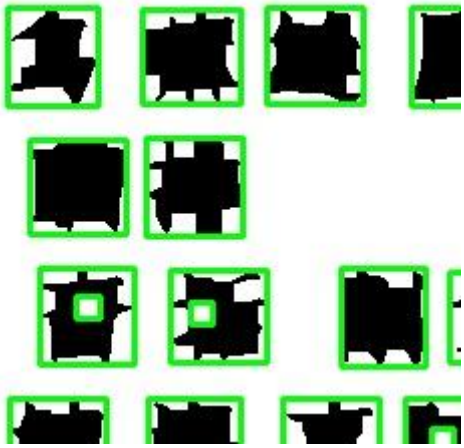

Figura 15 Detección de cuadros de contorno de los caracteres

En resumen, dado que los bordes exteriores y los bordes de los agujeros tienen una correspondencia uno a uno con las componentes conectadas de 1 píxel, el algoritmo determina las relaciones de contorno entre los bordes de una imagen binaria en forma rectangular.

Se puede observar que el algoritmo anterior tiene el defecto de detectar bordes en los pequeños agujeros que forman los caracteres chinos, para solucionar se ha probado a definir una relación de distancia entre los bordes detectados, esto no ha dado resultados buenos ya que existen bordes a fusionar con distancia muy larga al centro del sinograma como se puede observar en la figura 16.

$$distancia = \sqrt{\left(\frac{x_{1,1}+x_{1,2}}{2}\right) - \left(\frac{x_{1,1}+x_{1,2}}{2}\right)^2 + \left(\frac{y_{1,1}+y_{1,2}}{2}\right) - \left(\frac{y_{1,1}+y_{1,2}}{2}\right)^2}$$

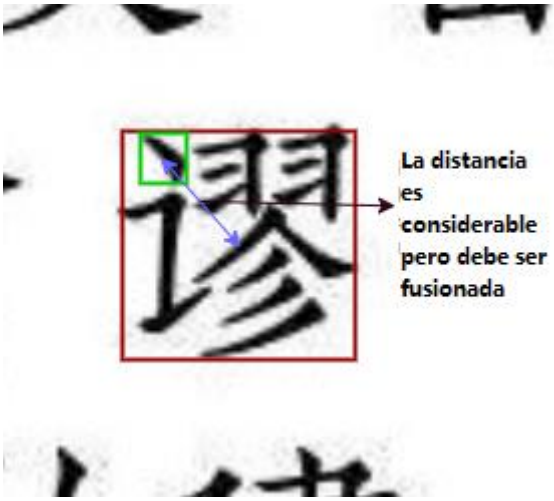

Figura 16 Fusión por distancia

Por otro lado, también se ve el solapamiento de estas cajas delimitadoras. Una buena medida de detección de solapamiento es IoU (*Intersection over Union*). Se detecta el rectángulo de intersección de estas imágenes con los puntos de inicio de la esquina inferior derecha y el punto de la esquina superior izquierdo, entonces la

unión será área del rectángulo 1 + área del rectángulo 2– área de intersección de ambos. Gracias a esto se logra suprimir todas las cajas que se encuentra dentro de intersección de otras.

Valores atípicos

El último paso se centra en la detección y exclusión de valores atípicos de las cajas delimitadoras.

En términos de filas, no se necesita ningún tipo de detección de valores atípicos dado que, si un carácter tiene un espacio horizontal entre sus partes, otro carácter junto a él cubrirá la fila en la exploración.

El recorrido de columnas en cada fila puede predecir a veces resultados erróneos y no deseados en los espacios verticales entre las partes del carácter (porque los caracteres de otras filas no ayudarán ya que no están alineados verticalmente).

Para la detección de estos, se emplea una versión ligeramente modificada de la regla estadística empírica [33]. En esta regla, también llamada regla 68-95-99.7, en general se utiliza la desviación estándar, pero como la desviación estándar debería ser muy baja para este conjunto de datos, en su lugar se toma una parte del primer momento (0.25). A continuación, se muestra la ecuación de la detección de valores atípicos. Aquí, μ es la media y σ es la desviación estándar.

$$outlier(x) = \{1, \mu - 0.25 * \mu \ < \ x \ < \ \mu + 0.25 * \mu; \ 0, en\ otros\ casos\}$$

Al revisar todos los valores atípicos de los cuadros de contornos detectados en la subsección anterior, si la anchura es menor que la anchura media, ese rectángulo se fusionará con el rectángulo más pequeño situado antes o después de mismo.

Por otro lado, si un ancho atípico es mayor que el ancho promedio, entonces simplemente se divide ese rectángulo en 2 rectángulos a través del centro.

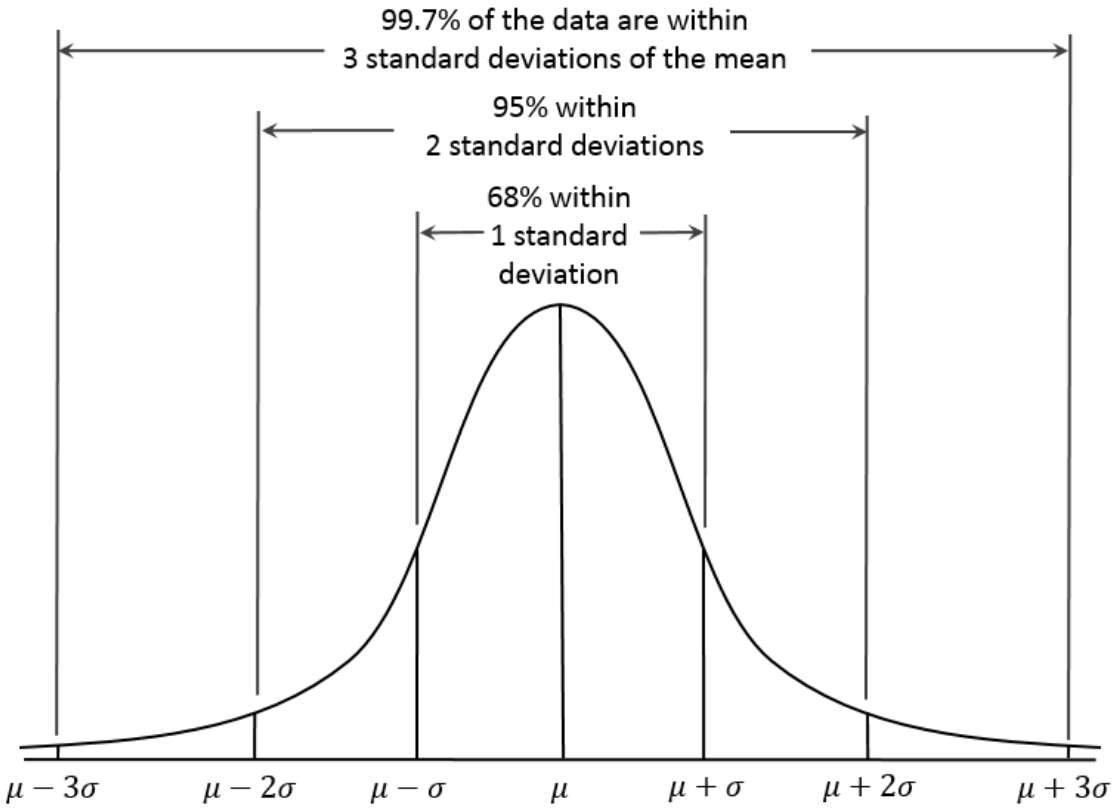

Figura 17 Regla empírica basado en la distribución normal [34]

5.1.1. Resultados

Para este enfoque se realiza una detección de precisión muy sencilla a nivel de máscara de caracteres identificados, esto no se puede realizar con precisión a nivel de carácter ya que no disponemos de la distribución original de los datos de entrada, por ello se toma la diferencia binaria por píxel de cada imagen y luego se divide la diferencia por el número total de píxeles de esa imagen. Esto es lo mismo que hacer la precisión de la intersección de las predicciones. A continuación, se muestra la ecuación empleada dónde la *g* es la máscara de la imagen original y *p* es la imagen de la máscara predicha.

$$\sum |g(x,y) - p(x,y)|$$

El resultado de comparación entre las máscaras de los datos de entrada contra la máscara predicha es del **97.8%** sobre una muestra total de 60.000 imágenes generadas al azar con el contenido de caracteres distribuido de forma aleatoria en diferentes líneas.

Cabe matizar que existe también la posibilidad de emplear algunos modelos de aprendizaje profundo para este preprocesamiento. Actualmente, los dos mejores sistemas de reconocimiento de caracteres son *EAST text detector* (An Efficient and

Accurate Scene Text Detector) y *Tesseract* [5]de Google. El principal problema del detector de texto EAST es que propone cuadros delimitadores de caracteres únicamente para la tarea de reconocimiento mediante el modelo LSTM. Y por esta razón, no es necesario que los cuadros de contornos delimiten exactamente el carácter en cada carácter. Por lo tanto, si aplicamos la métrica de precisión anterior, el enfoque de búsqueda de contornos implementado tendrá un rendimiento mucho mayor. Por otra parte, el enfoque actual es capaz de detectar los cuadros de contornos de aproximadamente 500 imágenes por segundo, mientras que cualquier modelo CNN será mucho más lento que esto.

5.2. Red neuronal artificial convolucional

El objetivo de este apartado es exponer de manera detallada cuál ha sido el proceso llevado a cabo para la obtención del modelo que clasificará los estilos caligráficos. En él se comienza mencionando los datos que se han utilizado para construir el modelo, los métodos utilizados y especificando el proceso paso por paso hasta obtener la red neuronal convolucional.

Después se explica la metodología particular utilizada dentro del Deep Learning, que son las Redes Neuronales Convolucionales. Así mismo, se especifica cómo se ha adaptado la mencionada metodología para el tratamiento de las imágenes de este proyecto.

Por último, se hace un repaso de los pasos seguidos para elegir el modelo, explicando desde las fases de entrenamiento hasta los parámetros de éxito que se han analizado. Lo interesante de este proyecto será ver cómo el modelo que en un principio parecía predecir mejor los estilos, tras un análisis más detallado resulta no ser el óptimo.

5.2.1. Datos de trabajo

Selección de fuentes de datos

Debido a que no se ha encontrado ninguna fuente de datos que cumpla con las necesidades que requiere este proyecto, se decide elaborar un set de datos propio. Esta elaboración se hace a partir de la extracción de imágenes desde distintos archivos que contienen fuentes de letras para ordenadores.

Todas las fuentes utilizadas en este proyecto han sido descargados desde:

http://www.fonts.net.cn/
https://www.freechinesefont.com/

[5] https://opensource.google/projects/tesseract

En una primera instancia, se recopilan varios archivos de fuentes para cada estilo caligráfico, destacando las escrituras de personajes históricos famosos como:

- **Estilo Cao** de *Mao ZeDong*, político y dirigente del Partido Comunista de China y fundador de la República Popular China.

- **Estilo Kai** de *Ding Yong Kan*, licenciado en caligrafía en la universidad de JiangSu, actual vicepresidente de *China Financial Calligraphers association.*

- **Estilo Xing** de *Sun ZhongShan*, Primer presidente de la República de China y fundador de Partido Nacionalista Chino.

No obstante, cabe matizar que todas las fuentes recopiladas son inputs que simulan a escritos a manos elaborados digitalmente por profesionales, esto se puede observar en las figuras 18 y 19.

Extracción de fuentes

Para la extracción se ha empleado un script sencillo de Python ejecutado bajo el programa open source de *FontForge*, Este software – creado originalmente por George Williams - [35] permite crear, modificar y convertir fuentes de contornos y mapas de bits para múltiples tipos de archivos de fuentes.

Se mantiene la siguiente estructura de carpeta para cada extracción:

./categoría_principal/subcarpeta_fuente/imágenes individuales

Siendo la *categoría principal* los cinco estilos caligráficos chinos (Kai, Xing, Li, Zhuan y Xing), y la *subcarpeta fuente* el nombre de cada archivo de fuente del que se extraen las imágenes particulares.

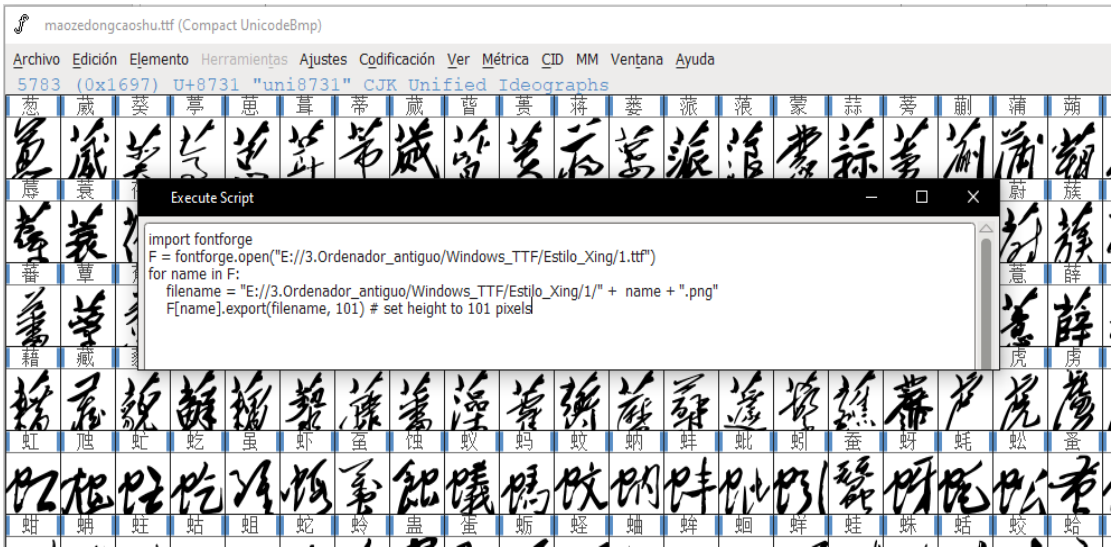

Figura 18 FontForge y exportación a png

Se puede observar que los caracteres extraídos tienen las siguientes características:

No tienen distorsiones, ruido ni rotaciones, como se podría esperar de una muestra obtenida a partir de textos puros.

Todas las imágenes tienen el mismo tamaño.

Los caracteres están en color negro y el fondo está en color blanco.

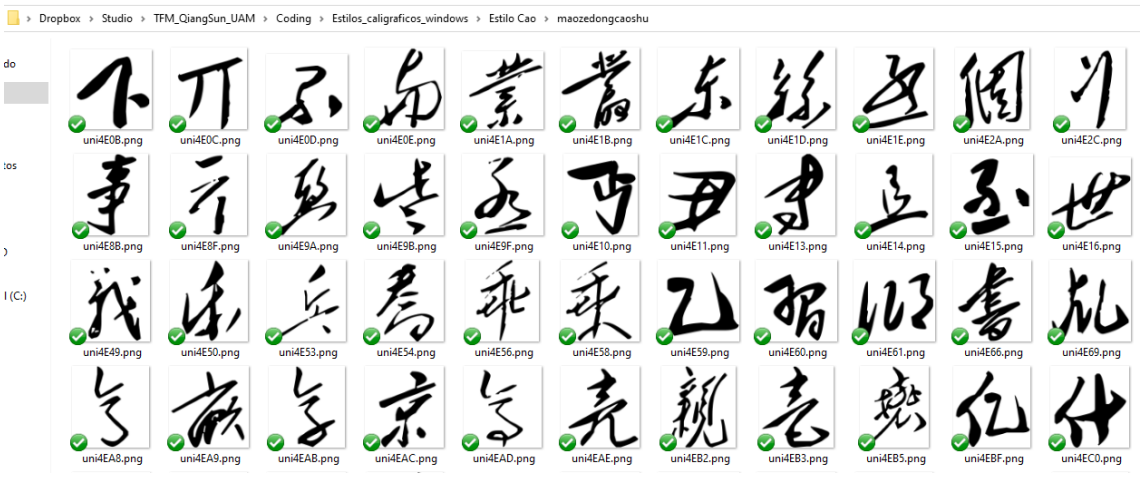

Figura 19 Imágenes individuales exportadas

Limpieza, generalización y reclasificación por clases

Tras recopilar suficientes muestras de sinogramas chinos, se procede a la limpieza de estos, con el objetivo de depurar falsos predictores de signos de puntuación, números y algunos caracteres kanji o pertenecientes a los alfabetos ruso/griego. Para realizar esta tarea se ha procedido a eliminar mediante un script en Python todas aquellas imágenes cuyo contenido o características no cumplan con las siguientes condiciones:

- Imágenes no vacías.
- Debe contener la secuencia “uni” en el nombre del fichero (Carácter Unicode).
- No estar contenido en una lista recopilada previamente, que contiene signos, números, signos de puntuación, etc… no deseados con nombre “uniXXX.png” ya que estos se tratan de falsos predictores

Después de limpiar los posibles observaciones anormales y extremas (*outliers*) de las subcarpetas, se empieza a normalizar las imágenes extraídas a un estándar común de 60x60 píxel. Después se agrupan en cinco carpetas principales, una por

cada uno de los cinco estilos caligráficos chinos, mediante un discreto script en Python.

Para evitar problemas de nombre con archivos duplicados entre distintas fuentes, se concatena el nombre de cada subcarpeta a los nombres individuales.

5.2.2. Estructura empleada

Tras finalizar la recopilación de muestras suficientes, se debe elegir el tipo de algoritmo a aplicar. Dentro del ámbito de Machine Learning, podemos categorizar 2 áreas principales: Aprendizaje supervisado (*supervised*) y aprendizaje no supervisado (*unsupervised*) dependiendo del tipo de problema y si nuestros datos están categorizados o no.

Para este proyecto en concreto se ha decidido utilizar un modelo de *Deep Learning* basado en Redes Neuronales Convolucionales (*CNN*). En este tipo de arquitectura de aprendizaje supervisado se emplea la modelización de redes neuronales artificiales, en las que las neuronas se corresponden a los campos receptivos. Estas neuronas están conectadas entre sí y van transmitiéndose la información tras aplicarle sus algoritmos, de tal manera que cada una de ellas va modelando dicha información hasta producir el output – Es decir, la información se trasmitirá capa por capa dónde las primeras llegan detectar formas simples mientras que las capas más profundas reconocen ya formas más complejas.

Se ha elegido esta arquitectura por los siguientes motivos:

Está optimizada para la detección, segmentación y categorización de imágenes.

Ya existen numerosos estudios previos en los que se han conseguido buenos resultados utilizando esta arquitectura [36]. No obstante, cabe señalar que en algunos casos su propósito es distinto al de este proyecto ya que detectan cada sinograma como una clase [37].

Permite volver a entrenar para nuevas tareas de clasificación a partir de redes/modelos ya entrenados, mediante el incremento de la muestra o aprovechando los inputs de los usuarios. Este nuevo entrenamiento puede produce una mayor precisión.

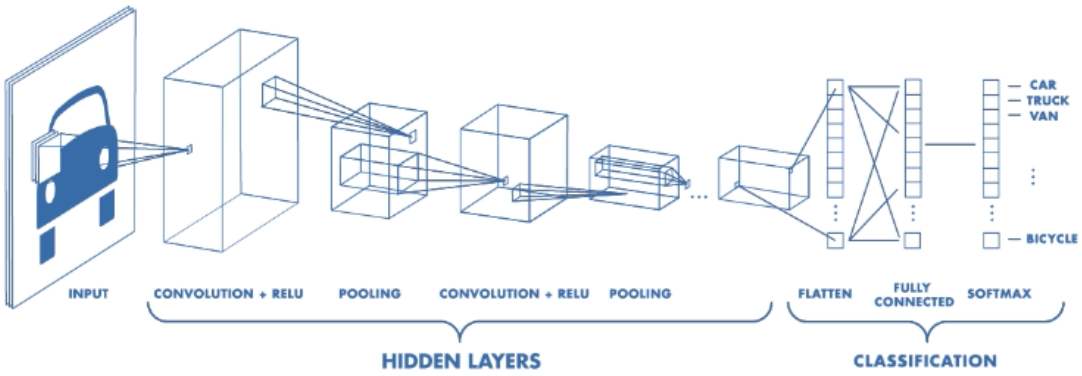

Figura 5-3 Arquitectura de una red convolucional [38]

Dentro la estructura de las CNN se ha decidido utilizar las siguientes características por entender que se ajustan mejor a lo que se pretende obtener:

Input: Imagen de (60,60,3), donde 60 es el alto y ancho de la imagen, y 3 la profundidad de la imagen (colores RGB)

Convolution: La convolución es uno de los principales componentes de una red CNN. El término convolución se refiere a la combinación matemática de dos funciones para producir una tercera función. En este paso se emplea un filtro o *kernel* para producir un mapa de características.

Activation: la capa no sólo trasmite la entrada que recibe, sino que emplea un paso adicional de función de activación. Esta función toma el resultado de la suma ponderada de la entrada anterior, y lo trasforma una vez más antes de la salida. La función empleada en este caso es ReLU (*rectified linear unit*), que se define como: *R(z)=max (0, z)*.

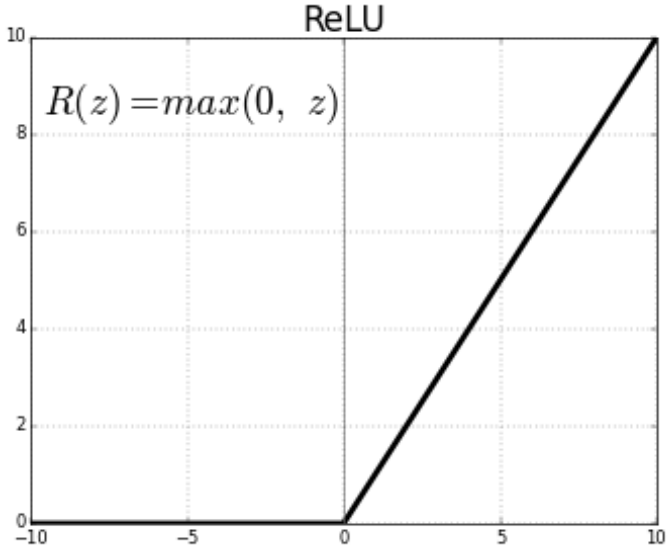

Figura 5-4 Función de activación ReLU

POOL: Función de agrupación; en este caso concreto se ha seleccionado el valor máximo de una cuadrícula prefijada (max_pooling). Esta función nos ayuda a reducir la dimensionalidad de la imagen.

DROPOUT: Este paso tiene el objetivo de reducir los sobreajustes y así evitar *overfitting* en los datos de entrenamiento.

Fully Connect layer: Capa de conexión completa, la cual conecta todas las características procedentes de neuronas previas y envía el valor de salida al clasificador. En este caso concreto, se opta por utilizar la función de activación softmax por las siguientes razones:

La función está orientada para problemas categóricos y disponemos de cinco categorías de estilo caligráfico.

Dado que se pretende mostrar al usuario los distintos resultados probabilísticos de la clasificación, Softmax nos devuelve una distribución de probabilidad discreta sobre las clases objetivas dónde la suma será del 100%.

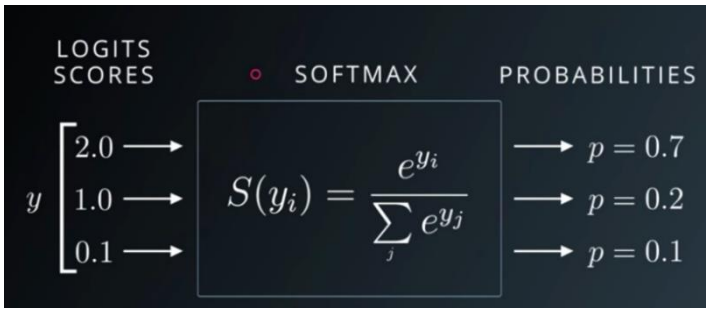

Figura 5-5 Función Softmax [39]

Por último, para compilar el modelo, hay que definir la función de perdida, el optimizador a utilizar y la métrica monitorear. En este caso concreto se utiliza:

Optimizador Adam (*Adaptive moment estimation*). Se trata de un algoritmo que combina ventajas de otros algoritmos y aprovechan el poder de los métodos de ratio de aprendizaje adaptativo para encontrar ratio de aprendizaje individuales para cada parámetro.

Función de perdida *categorical_crossentropy*: Cuantifica el error del output entre distintas categorías.

Métrica a monitorear precisión (*Accuracy*).

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 60, 60, 64)        1792
_________________________________________________________________
batch_normalization (BatchNo (None, 60, 60, 64)        256
_________________________________________________________________
activation (Activation)      (None, 60, 60, 64)        0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 30, 30, 64)        0
_________________________________________________________________
dropout (Dropout)            (None, 30, 30, 64)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 30, 30, 128)       204928
_________________________________________________________________
batch_normalization_1 (Batch (None, 30, 30, 128)       512
_________________________________________________________________
activation_1 (Activation)    (None, 30, 30, 128)       0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 15, 15, 128)       0
_________________________________________________________________
dropout_1 (Dropout)          (None, 15, 15, 128)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 15, 15, 512)       590336
_________________________________________________________________
batch_normalization_2 (Batch (None, 15, 15, 512)       2048
_________________________________________________________________
activation_2 (Activation)    (None, 15, 15, 512)       0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 7, 7, 512)         0
_________________________________________________________________
dropout_2 (Dropout)          (None, 7, 7, 512)         0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 7, 7, 512)         2359808
_________________________________________________________________
batch_normalization_3 (Batch (None, 7, 7, 512)         2048
_________________________________________________________________
activation_3 (Activation)    (None, 7, 7, 512)         0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 3, 3, 512)         0
_________________________________________________________________
dropout_3 (Dropout)          (None, 3, 3, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 4608)              0
_________________________________________________________________
dense (Dense)                (None, 256)               1179904
_________________________________________________________________
batch_normalization_4 (Batch (None, 256)               1024
_________________________________________________________________
activation_4 (Activation)    (None, 256)               0
_________________________________________________________________
dropout_4 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               131584
_________________________________________________________________
batch_normalization_5 (Batch (None, 512)               2048
_________________________________________________________________
activation_5 (Activation)    (None, 512)               0
_________________________________________________________________
dropout_5 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 5)                 2565
=================================================================
Total params: 4,478,853
Trainable params: 4,474,885
Non-trainable params: 3,968
_________________________________________________________________
```

Figura 5-6 Estructura del CNN final empleado

Es importante matizar que la estructura de red neuronal convolucional empleada y detalladas en los puntos anteriores es fruto de varias iteraciones de prueba y error. Entrenando el mismo set acotado de unos miles de input por cada categoría sobre diferentes combinaciones de estructura para seleccionar la estructura con mejor resultado. Se ha partido de una estructura básica de dos bloques de convolución y se ha ido añadiendo más números de capas, *Dropout*, probando distintos algoritmos de activación, etc…

Finalmente se utiliza una estructura de cuatro bloques de convolución y dos niveles de *fully connect*. Aunque el resultado con tres bloques es casi idéntico con cuatro, en torno al 97%.

Una vez que tenemos preparado el modelo, y antes de realizar el entrenamiento de este, se debe separar el set de datos. Para ello se realiza la siguiente distribución de los datos: 80% en training, 10% en validación y 10% en test para cada una de las 5 categorías. Se trata de una distribución estándar para este tipo de modelos, aunque se manejan otras alternativas. Lo común a todas es que la mayor parte debe dedicarse a entrenamiento.

5.2.3. Entrenamientos y resultado

Primera aproximación

En la primera ronda de entrenamiento se han utilizado los siguientes números de inputs recolectados desde las distintas fuentes, separándolos aleatoriamente en una distribución de:

- Conjunto de **entrenamiento** (*training*): 80% sobre el total de archivos. Es el conjunto principal que se utiliza para entrenar y ajustar la red neuronal.

- Conjunto de **validación** (*validation*): 10% sobre el total de archivos. Es el conjunto que se emplea para ajustar los parámetros en la fase de validación. Es sumamente importante ya que ayuda a validar la exactitud de la red neuronal y evitar así el exceso de ajustes (*overfitting*).

- Conjunto de **prueba** (*test*): 10% sobre el total de archivos. Es una muestra independiente que no se emplea en la fase de entrenamiento. Esta muestra proporciona una evaluación imparcial sobre el modelo final.

	Cao	***Kai***	***Li***	***Xing***	***Zhuan***	***Total***
Training set (80%)	37.672	43.894	40.982	49.518	69.613	**241.679**
Test set (10%)	4.710	5.487	5.123	6.190	8.702	**30.212**
Validation set (10%)	4.709	5.487	5.123	6.190	8.702	**30.211**
Total	**47.091**	**54.868**	**51.228**	**61.898**	**87.017**	**302.102**

Tabla 5-1 Distribución de datos - modelo 1

Como se puede apreciar, existe un gran desequilibrio entre los sets de datos del estilo Cao y Zhuan con respecto a Kai, Li y Xing. Esto se debe a que, a pesar de elegir cantidades iguales de fuentes a la hora de extraer los caracteres individuales, estas fuentes no siempre contienen la misma cantidad de caracteres válidos (algunos se han descartado en el proceso de limpieza). Esta diferencia en el número de inputs provoca disfunciones a la hora de la clasificación y, tal y como se explica más adelante, determinará el descarte de este primer modelo.

Para el entrenamiento del modelo se ha apalancado la función de *fir_generator* de la librería de *Keras.* Se trata de una librería especializada en funciones de redes neuronales basadas en la colaboración abierta dentro del *framework* de *Tensorflow*. Este *framework* consiste en un sistema de aprendizaje automático de código abierto, que permite aprovechar la capacidad de procesamiento gráfico de las tarjetas gráficas para el cómputo de los algoritmos. Estos procesadores gráficos (*GPU*) son mucho mejores a la hora de lidiar con este tipo de procesos debido a la inmensa cantidad de procesos simultáneos que permite llevar a cabo.

Los parámetros utilizados para el entrenamiento han sido los siguientes:

- Modelo *secuencial* descrito en el apartado anterior, este permite crear modelos capa por capa de forma secuencial como bien indica el nombre.

- Tamaño de *batch*: *128.* Se procesará en subconjunto por lotes formado de 128 imágenes.

- Épocas de entrenamiento: *10.* Esto marca el número de veces con las que el modelo será entrenado con los inputs de entrenamiento para ponderar los pesos de cada imagen de entrenamiento.

- Guardar el mejor resultado: *True.* Con esta opción habilita el guardado de la interacción de entrenamiento con mejor resultado sobre la métrica monitoreada.

El proceso de entrenamiento ejecutado en una tarjeta gráfica *NVIDIA 2070 8GB* duró aproximadamente dos horas y el mejor resultado ha sido durante la última iteración con un coeficiente de exactitud (*accuracy*) de 97.6%:

```
Epoch 10/10
1888/1889 [============================>.] - ETA: 0s - loss: 0.1383 - accuracy: 0.9547
Epoch 00010: val_accuracy improved from 0.96872 to 0.97620, saving model to data_model/model_small_weights.hdf5
1889/1889 [==============================] - 370s 196ms/step - loss: 0.1384 - accuracy: 0.9547 - val_loss: 0.0678 - val_accurac
y: 0.9762
1:01:47.386535
```

Figura 20 Resultado de entrenamiento red neuronal 1

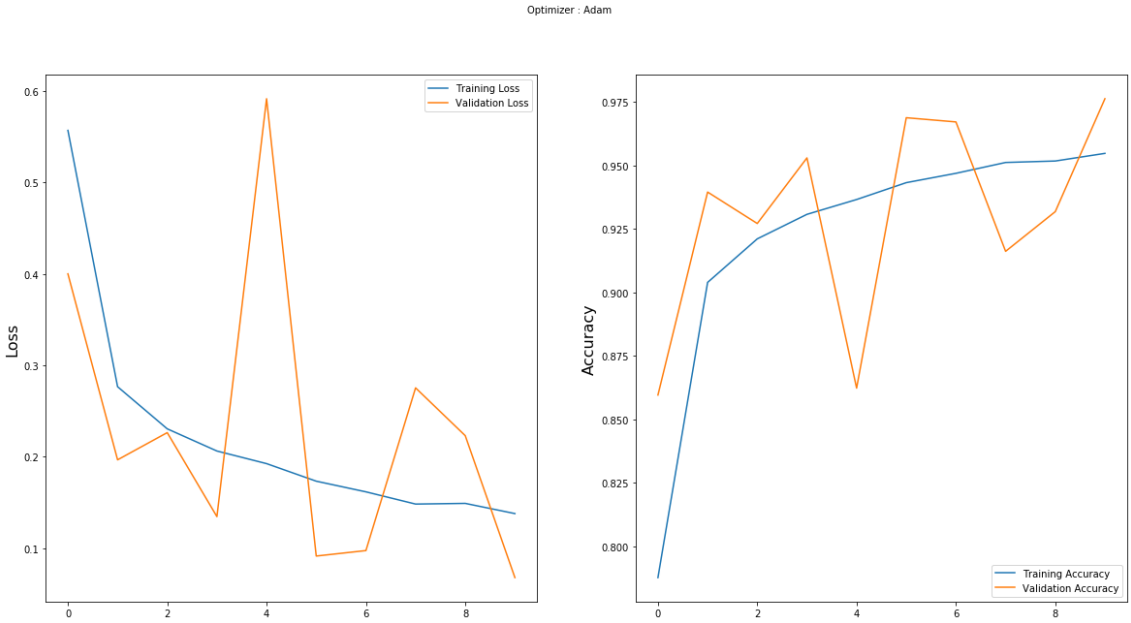

Figura 21 Training Loss & Validation Loss red neuronal 1

Tras aplicar el conjunto de pruebas sobre el modelo entrenado, se procede a calcular la matriz de confusión. Esta matriz indica la relación entre falsos positivos, falsos negativos, verdaderos negativos y verdaderos positivos dentro de los resultados:

		Predicción	
		Positivos	**Negativos**
Observación	**Positivos**	Verdaderos Positivos (VP)	Falsos Negativos (FN)
	Negativos	Falsos Positivos (FP)	Verdaderos Negativos (VN)

Tabla 5-2 Matriz de confusión

Esta es la matriz de confusión obtenida en el proceso de validación para las cinco categorías:

	Cao	***Kai***	***Li***	***Xing***	***Zhuan***
Cao	***4.555***	92	25	34	4
Kai	4	***5.388***	82	13	0
Li	3	91	***5.021***	7	1
Xing	35	214	81	***5.857***	3
Zhuan	0	27	40	7	***8.628***

Tabla 5-3 Matriz de confusión modelo 1

Se observa que el resultado del estilo Zhuan es casi perfecto por los pocos errores de falsos positivos o falsos negativos que se muestra, lo cual encaja con lo esperado ya que este estilo tiene características propias más distintivas que hacen que sea

más fácil distinguirlo. Sobre el resto de los estilos, los resultados en general son muy buenos, por lo que se procede a revisar las métricas complementarias del *accuracy:*

Precision: Es la relación entre las observaciones positivas correctamente previstas y el total de observaciones positivas previstas. Determina la calidad del modelo entrenado:

$$Precision = \frac{True\ positive}{True\ positive + False\ positive}$$

Recall: Es la relación entre las observaciones positivas correctamente pronosticadas y todas las observaciones de la clase real:

$$Recall = \frac{True\ positive}{True\ positive + False\ negative}$$

F1-score: La puntuación de la F1 es el promedio ponderado de las ratios de *Precision* y *Recall*, esta puntuación tiene en cuenta tanto los falsos positivos como los falsos negativos.

$$F1 - score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Support: el número de muestras de la clase correcta que se encuentran en esa clase

	Precision	***Recall***	***F1-score***	***Support***
Cao	0,99	0,97	0,98	4.710
Kai	0,93	0,98	0,95	5.487
Li	0,96	0,98	0,97	5.123
Xing	0,99	0,95	0,97	6.190
Zhuan	1,00	0,99	1,00	8.702
Accuracy			0,97	30.212
Macro avg	0,97	0,97	0,97	30.212
Weighted avg	0,98	0,97	0,97	30.212

Tabla 5-4 Informe detallado de métricas modelo 1

Según los resultados obtenidos, parece que estamos ante el modelo ideal para el proyecto. Sin embargo, tras recolectar varias muestras reales recolectadas manualmente de internet, y realizar la evaluación de éstas con la red neuronal (simulando el comportamiento del aplicativo web), el resultado de las clasificaciones obtenido tiende a sobrestimar al estilo Zhuan. Esto se debe a que es la clase con más datos en la muestra de entrenamiento. Por otra parte, el

resultado de clasificación con las imágenes de estilo Cao tiende a dar muchos resultados no esperados.

Red neuronal final

Para resolver el problema del entrenamiento anterior, se analiza en primer lugar los inputs de cada categoría. Se aprecia una gran diferencia entre el número de muestras del estilo Cao y las del estilo Zhuan, siendo las de Cao aproximadamente la mitad que las de estilo Zhuan. Para solucionar este problema existen varios tipos de técnicas, entre las cuales destacan:

- Aplicación de algoritmo de sobre-muestreo (*oversampling*) como *SMOTE* [40]: Consiste en generar datos ficticios a partir de características comunes entre las muestras de la clase con menor muestra.

- Aplicación de algoritmo de sub-muestreo (*undersampling*): Permite reducir el número de ejemplos en las clases mayoritarias, con el objetivo de conseguir unos números similares de ejemplares para cada clase.

- Aplicar técnicas de *Data Augmentation*. Consiste en crear muestras a partir de las existentes aplicando ciertas transformaciones como de rotación, volcado, ruido, brillo, etc.…

- Recopilar más datos para compensar las clases con menor números de muestras.

En este caso concreto, se opta por simplemente añadir más datos, ya que, con una inversión de tiempo, retratamiento y aplicando técnicas de *undersampling* aleatorio sobre las clases con demasiadas muestras se puede lograr disponer de unos datos iniciales distribuidos equitativamente entre las categorías. Además, la imprecisa identificación del estilo Cao mencionada anteriormente podría deberse a una variedad insuficiente en los inputs de los que se disponía.

Adicionalmente a la equiparación de muestras mencionadas en el apartado anterior, se decide aplicar algunas técnicas de *Data Augmentation* [41] (rotación aleatoria entre 0 a 5 grados hacia derecha o izquierda o el aumento/disminución de la imagen entre -5 y 5 por ciento). Todo ello tiene el objetivo de simular los posibles casos de recortes realizados a mano alzada que implementaremos más tarde en el front-end.

Tras realizar los ajustes mencionados se obtiene el siguiente conjunto de datos con más de un millón de muestras repartidas equitativamente entre las 5 clases a clasificar:

	Cao	***Kai***	***Li***	***Xing***	***Zhuan***	***Total***
Conjunto previo	47.083	52.460	51.288	60.739	87.017	**298.587**
Conjunto nuevo	68.000	68.000	68.000	68.000	68.000	**340.000**
Data Augment.	136.000	136.000	136.000	136.000	136.000	**680.000**
Total	**204.000**	**204.000**	**204.000**	**204.000**	**204.000**	**1.020.000**

Tabla 5-5 Input red neuronal 2

	Cao	***Kai***	***Li***	***Xing***	***Zhuan***	***Total***
Training set (80%)	163.200	163.200	163.200	163.200	163.200	**816.000**
Test set (10%)	20.400	20.400	20.400	20.400	20.400	**102.000**
Validation set (10%)	20.400	20.400	20.400	20.400	20.400	**102.000**
Total	**204.000**	**204.000**	**204.000**	**204.000**	**204.000**	**1.020.000**

Tabla 5-6 Distribución de datos – red neuronal 2

Tras obtener los nuevos datos, existen dos posibilidades de continuar el ejercicio:

- Aplicando nuevos entrenamientos sobre la red neuronal anterior ya entrenada.
- Entrenar desde el principio con el conjunto nuevo.

En este caso concreto, aplicar más entrenamiento sobre el conjunto anterior no sería complicado ya que no tenemos la necesidad de adaptar números de salidas de las capas ni las clases a predecir. Las clases siguen siendo las mismas. Pero por este mismo motivo, junto con el hecho de que el segundo conjunto de datos engloba al primero, se opta por entrenar desde un principio con la misma red definida, añadiendo unas iteraciones más sobre el set de entrenamiento (*epoch*).
Tras unas ocho horas de entrenamiento, concluyen las 15 iteraciones alcanzando en su mejor situación un valor de ratio de precisión sobre set de validación de 98,6% y un 97.5% sobre el set de prueba.

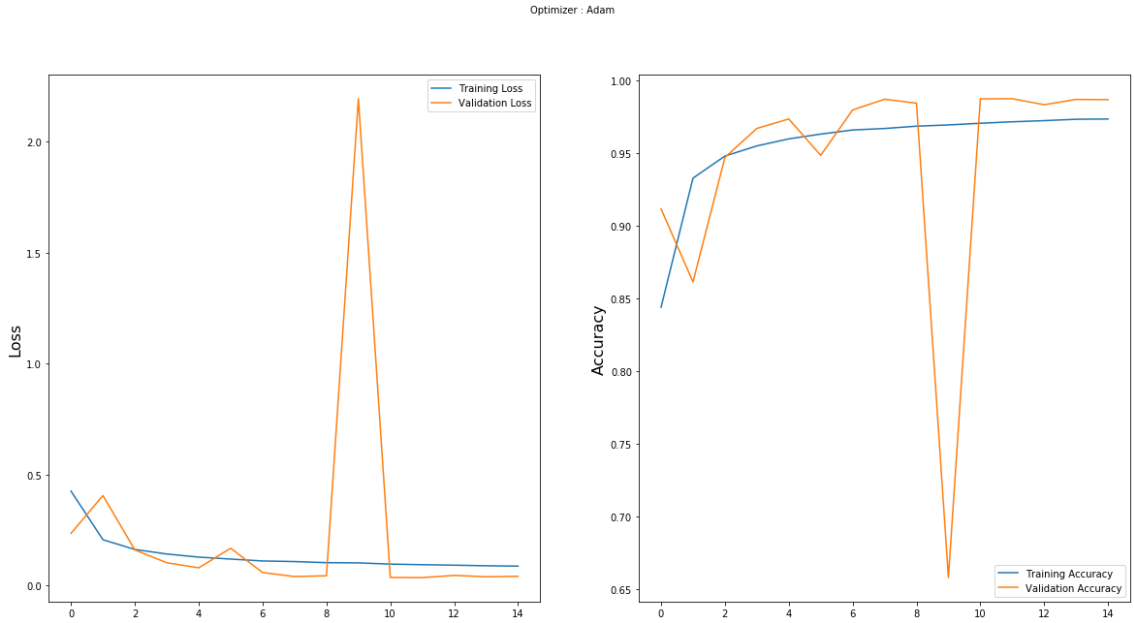

Figura 22 Resultado red neuronal 2

Tras aplicar el set de prueba sobre la red neuronal resultante, se obtienen las siguientes métricas:

	Cao	***Kai***	***Li***	***Xing***	***Zhuan***
Cao	***19.762***	322	15	252	2
Kai	48	***20.306***	6	33	0
Li	12	105	***20.225***	14	0
Xing	32	499	7	***19.820***	0
Zhuan	4	16	2	4	***20.331***

Tabla 5-7 Matriz de confusión red neuronal 2

Se registra mayor error entre la clasificación del estilo Cao y Xing, lo cual se puede explicar fácilmente ya que se tratan de 2 estilos muy parecidos (Cao se conoce como una escritura "corriente" y Xing como una escritura "andante"). El segundo mayor error se encuentra en estilo Kai ya que, dependiendo del carácter, hay bastantes similitudes con algunas letras del Cao o Xing.

	Precision	***Recall***	***F1-score***	***Support***
Cao	1,00	0,97	0,98	20.353
Kai	0,96	1,00	0,98	20.353
Li	1,00	0,99	0,98	20.356
Xing	0,98	0,97	0,98	20.358
Zhuan	1,00	1,00	1,00	20.357
Accuracy			0,98	101.777
Macro avg	0,99	0,99	0,98	101.777
Weighted avg	0,99	0,99	0,98	101.777

Tabla 5-8 Informe detallado de métricas red 2

Año	*Autor*	*Estudio*	*Método*	*Accuracy*
2015	Yuhao Zhang [37]	Deep Convolutional Network for Handwritten Chinese Character Recognition	Clasificación de carácter individual	97,70%
2016	Chen, Z, Yu-Sheng [23]	Machine Learning for Calligraphy Styles Recognition	Clasificación de estilo caligráfico	78,50%
2016	Boqi, Li [42]	Convolution Neural Network for Traditional Chinese	Clasificación de carácter individual	88,60%
2017	Zhang Jiulong; Guo Luming; Yang Su; Sun Xudong; Li Xiaoshan [43]	Detecting Chinese calligraphy style consistency by deep learning and one-class SVM	Clasificación de carácter individual	85,50%
2017	Gao Pengcheng,Gu Gang,Wu Jiangqin,Wei Baogang [44]	Chinese calligraphic style representation for recognition	Clasificación de carácter individual	94,22%
2018	Wen, Yuanbo [36]	Character style recognition based on full-page document	Clasificación de estilo caligráfico	96,70%
2020	**Resultado actual**		**Clasificación de estilo caligráfico**	**97,50%**

Tabla 5-9 Benchmark resultado

Como se podrá observar en la tabla anterior, el *accuracy* de la red neuronal convolucional entrenado es uno de los mejores hasta la fecha. No obstante, cabe señalar que, en la utilidad de la aplicación real, este resultado se verá afectado por una media ponderada de cada carácter clasificado de forma individual:

$$Resultado\ final\ predicta = argmax\left(\frac{1}{N}\sum_{i=1}^{N}(R_i)\right)$$

Siendo *N* el número de muestras a evaluar y *R* el resultado en forma de vector de cada predicción individualmente.

5.3. Back-end

En este apartado se hace un resumen pormenorizado de las herramientas y software utilizados para mantener la página web en funcionamiento. Es decir, abarca los pasos necesarios para poner en funcionamiento el servidor, el cómo se procesarán, dónde se almacenarán las imágenes enviadas y, como último elemento, un desglose final con los elementos necesarios para la traducción de los caracteres.

Además de detallar qué elementos se han empleado, también se menciona las características particulares de cada uno que han favorecido su elección y empleo. La capacidad de cumplir los requisitos de la página, y hacerlo de manera eficiente han sido los principales puntos para tener en cuenta.

5.3.1. DRF

DRF (*Django Rest Framework*) [45] es una herramienta de código abierto desarrollada en Python desde un proyecto financiado en parte con la colaboración de grandes empresas y por otra parte con las contribuciones voluntarias de usuarios. Este framework tiene su base en el famoso marco de aplicaciones web Django[6].

Se opta por emplear DRF por los siguientes motivos:

- Escrito en **Python**: Aparte de aportar ventajas como versatilidad, multiplataforma, multiparadigma y código abierto; permite integrar de forma muy sencilla la red neuronal -entrenada también en código Python-.

- **Simplicidad**: Ofrece una interfaz de programación de aplicaciones (API) navegable y fácil de testear por los desarrolladores; y una herramienta de interfaz de la línea de comandos (CLI) para controlar el proyecto desde consola.

- **Seguridad**: Integra autenticaciones OAuth1 y OAuth2 y tiene cobertura de prueba de código fuente.

- **Escalabilidad**: Resulta muy fácil de integrar cualquier función o aplicación adicional a tu proyecto.

- Dota de una amplia **documentación** y excelente **apoyo** comunitario por su base sobre Django.

Un proyecto DRF tiene la siguiente estructura:

[6] https://www.djangoproject.com/

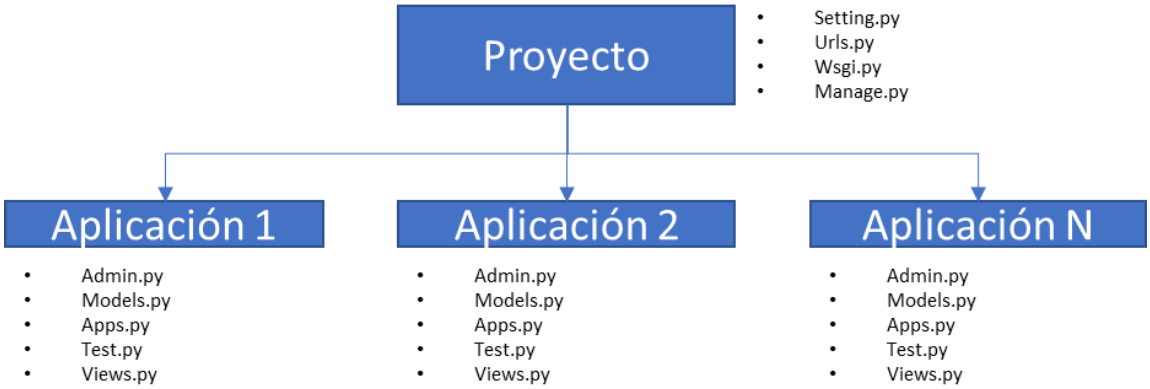

Figura 23 Proyecto DRF

A nivel de proyecto se distribuyen tanto las direcciones de rutas a los distintos API *views* definidos en las aplicaciones como las distintas configuraciones del proyecto. Un proyecto realizado con DRF a su vez puede contener diferentes aplicaciones, cada una de las cuales tiene usos distintos (aplicación de usuario, aplicación de administrador, etc.). Estas aplicaciones generalmente se componen de:

Admin.py: Se agrega la información de los modelos en CLI para poder realizar operaciones contra la base de datos de crear, leer, actualizar o eliminar (CRUD).

Models.py: Contiene la definición de los modelos de base de datos.

Apps.py: Descripción de la aplicación.

Test.py: Incluye las pruebas unitarias de la aplicación.

View.py: Abarca la lógica de construcción y tratamiento de los JSON recibidos/enviados.

En el caso concreto de la API, se desarrolla en un único proyecto (tfm_api) que contiene una única aplicación “api”. La configuración del Urls.py redirecciona las peticiones entrantes a la aplicación de “api” como muestra en la siguiente Figura:

```
from django.contrib import admin
from django.urls import path
import api.views

urlpatterns = [
    path('char_prediction/', api.views.char_prediction),
    path('save_image/', api.views.save_image),
]
```

Figura 24 Urls DRF

Dentro de la clase *view* de Django, los dos métodos de *API VIEW* utilizados son, los cuales sirven para leer y construir las peticiones al servidor, y para leer las respuestas de éste. No obstante, no han sido los únicos que se han incorporado. A continuación, se muestra una lista de otras funciones auxiliares implementadas que sirven para complementar el proyecto y facilitar la labor de responder cada una de las peticiones recibidas:

LoadingModel (): Esta función carga el modelo en la red neuronal convolucional para su posterior utilización en la clasificación de caracteres.

Model_prediction*(array_of_images)*: Esta es la función principal donde recibe una colección de imágenes en formato base64 y retorna el resultado medio de la predicción.

save_images *(all_images, email, correct_style)*: Esta función nos permite guardar en una ruta específica del sistema Linux los distintos aportes que se reciben desde el cliente web. Permitiendo así guardarlos y recopilar más muestras de información para posteriores refinamientos de la red neuronal.

getCharacterOCR*(all_images)*: Función que proporciona tanto el reconocimiento óptico de cada imagen como su traducción al inglés de cada imagen recortada.

updateBaiduTokens *()*: Función auxiliar del método anterior que posibilita la renovación de token de la API Baidu para utilizar su servicio.

5.3.2. Servidor y dominio empleado

Para alojar tanto la página web como la propia API, se utiliza una máquina de servidor compartida del proveedor *DigitalOcean*[7]. Se usa un plan contratado de CPU compartido, 2GB de RAM y 25GB de SSD para el almacenamiento, y con un límite mensual de 2TB de tráfico.

El hecho de contratar un servidor desde un proveedor se obtiene los siguientes beneficios:

- Garantía de disponibilidad
- Ahorro en la contratación de IP fija
- Evitar tener una máquina encendida las 24h del día.

[7] https://www.digitalocean.com/products/droplets/

Por parte de dominio, se ha solicitado las siguientes direcciones de páginas webs (*urls*):

https://chinese-calligraphy-tools-uam.website: para alojar la página web.

https://chinese-calligraphy-tools-uam-eps.website: para llamadas a API.

Tras solicitar los dominios, únicamente debemos configurarlo para que redireccione a la IP de la máquina del servidor. Esto hará que cualquier solicitud realizada sea dirigida a ellos directamente.

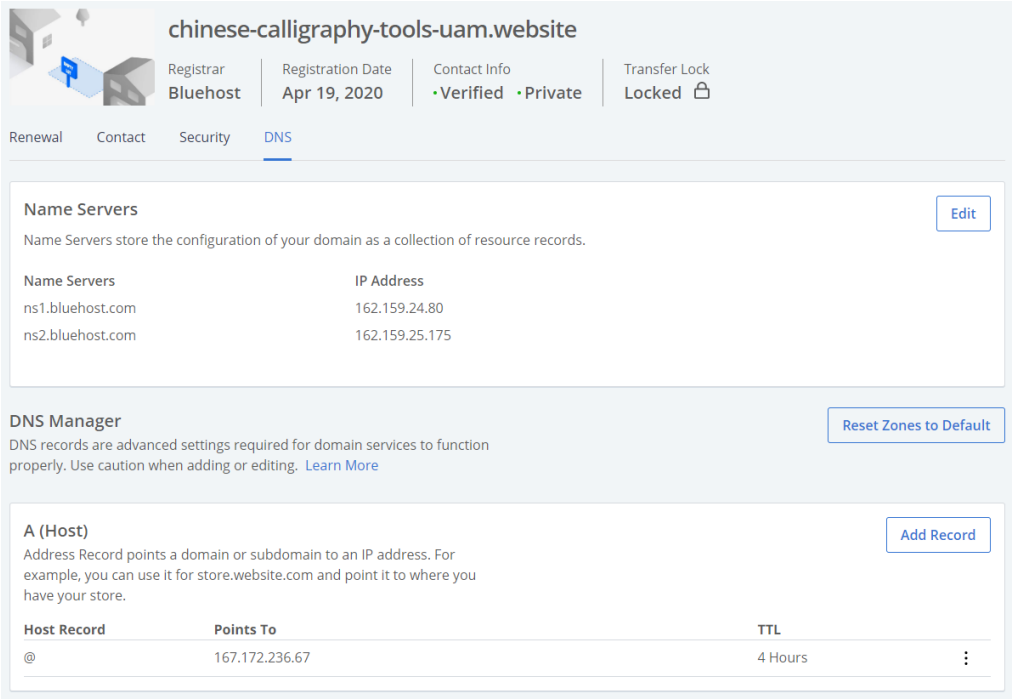

Figura 25 Dominio Bluethost[8]

Para poder utilizar los certificados *Let's Encrypt* en la web y habilitar el protocolo HTTPS, se ha empleado la herramienta de software libre y de código abierto Certbot [9]. Esto, además de configurar la firma de clave con la autoridad certificadora, admite a la vez configurar la renovación de estos certificados [46].

[8] https://www.bluehost.com/
[9] https://certbot.eff.org/

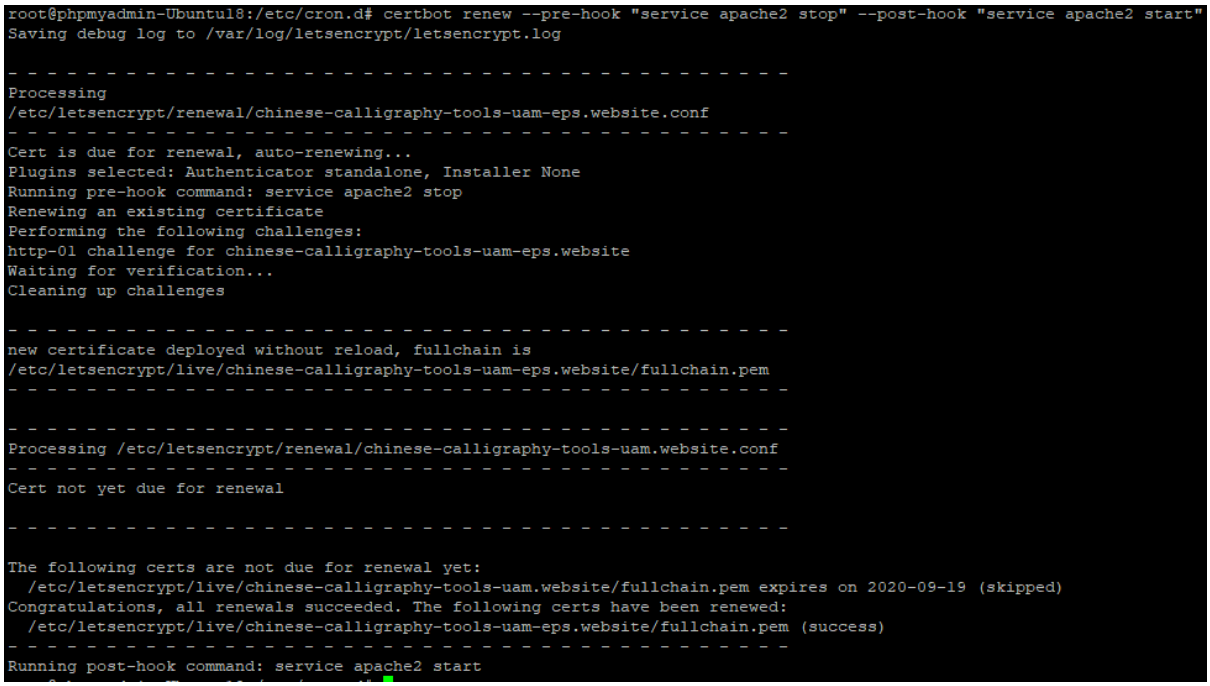

Figura 26 Renovación de certificado

5.3.3. Configuración de entorno y apache2

La instalación de apache2 y Python3 es llevada a cabo en la máquina del servidor para dar cobertura a la necesidad de alojar tanto la página web como desplegar la aplicación de DRF.

Apache es un conocido paquete de software que permite crear un servidor web de forma gratuita con código abierto, mantenida y desarrollada por la fundación de *Apache Software Foundation*[10]. Según datos estadísticos actuales, hasta un 37% de todas las páginas utilizan apache como servidor web [47].

Una vez instalado apache, se necesitan ciertos ajustes en la configuración para su correcto funcionamiento:

Configuración de *Uncomplicated Firewall* (*UFW*) [48] para permitir el tráfico en el puerto 80 (http) y 443 (https) de apache. Básicamente se trata de una versión simplificada de la configuración de iptables.

[10] https://www.apache.org/

```
/etc/apache2/sites-enabled$ ufw status
Status: active

To                         Action      From
--                         ------      ----
22/tcp                     LIMIT       Anywhere
443/tcp                    ALLOW       Anywhere
80/tcp                     ALLOW       Anywhere
3306                       ALLOW       Anywhere
8000                       ALLOW       Anywhere
```

Figura 27 UFW configuración

Se edita el fichero de configuración por defecto y el puerto 443 de la carpeta /etc/apache2/sites-available para redireccionar las peticiones al puerto 80 http al 443 https:

```
<VirtualHost *:80>
    ServerName www.chinese-calligraphy-tools-uam.website
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/html

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined

    RewriteEngine on
    RewriteCond %{SERVER_NAME} =chinese-calligraphy-tools-uam.website
    RewriteRule ^ https://%{SERVER_NAME}%{REQUEST_URI} [END,NE,R=permanent]
</VirtualHost>
```

Figura 28 Configuración HTTP

```
<IfModule mod_ssl.c>
<VirtualHost *:443>

    ServerName chinese-calligraphy-tools-uam.website

    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/html

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
    Include /etc/letsencrypt/options-ssl-apache.conf
    SSLCertificateFile /etc/letsencrypt/live/chinese-calligraphy-tools-uam.website/fullchain.pem
    SSLCertificateKeyFile /etc/letsencrypt/live/chinese-calligraphy-tools-uam.website/privkey.pem
</VirtualHost>

<VirtualHost *:443>
    ServerName chinese-calligraphy-tools-uam-eps.website

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined

    Include /etc/letsencrypt/options-ssl-apache.conf
    SSLCertificateFile /etc/letsencrypt/live/chinese-calligraphy-tools-uam-eps.website/fullchain.pem
    SSLCertificateKeyFile /etc/letsencrypt/live/chinese-calligraphy-tools-uam-eps.website/privkey.pem

    WSGIDaemonProcess chinese-calligraphy-tools-uam-eps.website python-home=/var/www/python-app/tfm_api_environment
    WSGIScriptAlias / /var/www/ejemplo/ejemplo/wsgi.py  process-group=chinese-calligraphy-tools-uam-eps.website

    <Proxy *>
        AddDefaultCharset Off
        Order deny,allow
        Allow from all
    </Proxy>
</VirtualHost>
</IfModule>
```

Figura 29 Configuración HTTPS

Después del último paso, se arranca el servicio de apache y se levanta la página web como el *endpoint* de la API

4.1.4.1 Instalación de virtualenv & red neuronal

Una vez que el servidor web está listo y operativo, se procede a la configuración del entorno de Python necesario para evaluar el modelo y también al despliegue del DRF.

Gracias a la versatilidad que proporciona Python con su herramienta de utilidad de *virtualenv*, se puede crear un entorno aislado que conecta sus propios directorios de instalación. Los conecta a pesar de que dichos directorios no comparten las bibliotecas con otros entornos *virtualenv* o las bibliotecas instaladas globalmente en el servidor, lo cual significa una gran ventaja. Esta herramienta con sus distintos paquetes configurados se extrae desde el *framework* de Anaconda - Windows mediante el siguiente comando:

```
$ Conda list -e > req.txt
```

Posteriormente se instalan todos los paquetes utilizados para el entrenamiento de la red neuronal, junto con algunas librerías extras necesarias para el DRF en el servidor Linux.

Se sube al servidor la red neuronal artificial entrenada mediante la aplicación de un cliente con protocolo de transferencia segura de archivos (*SFTP*) - WinSCP. Junto a ello también se sube la definición de la estructura de la red neuronal artificial en formato JSON, que será utilizada por las llamadas desde la capa de direccionamiento de DRF.

5.3.4. Baidu API & Google translate

Con el objetivo de ofrecer al usuario una experiencia completa en el ciclo de predicción de cada carácter, se ha realizado un estudio de los distintos servicios OCR de caracteres chino que hay en el mercado. Tras este análisis, el candidato final ha sido el servicio de OCR Baidu [49] por los motivos que se exponen a continuación:

No tiene la necesidad de distinguir si es chino tradicional o simplificado, esto permite tener la libertad de aceptar más tipos de caracteres frente a los otros servicios donde necesitamos especificar si es chino tradicional o simplificado. Adicionalmente a que los usuarios son libres de elegir el input sea tradicional o simplificado y de esta forma evitaremos la precondición de conocerlo para especificarlo.

Las primeras 500 peticiones del día son gratuitas, y dado el volumen de tráfico que se espera podría ser suficiente.

Ofrece una gran fiabilidad con respecto a otras API en que proporcionan un servicio similar, obteniendo resultados bastantes aceptables para todos los estilos salvo el Zhuan.

Trabaja con imágenes codificadas en base64 que coincide con la codificación utilizada para la comunicación entre la página web y back-end.

La solución la ofrece Baidu que es una empresa especializada en servicios relacionados con internet e inteligencia artificial. Coloquialmente se la conoce como el "Google Chino" ya que el abanico de servicios y productos es estrechamente similar en ambos casos.

Para utilizar sus servicios, en primer lugar, hay que registrarse en la página de Baidu Inteligencie Cloud[11]. Una vez registrado y autenticadas las credenciales, se dispone de los permisos para acceder al panel principal de aplicaciones y servicios que tiene una interfaz bastante similar a la consola de desarrolladores de Google.

A continuación, se selecciona el servicio de reconocimiento de caracteres y se crea una solicitud para obtener las credenciales de *API key* y *Secret Key*. Éstas serán las claves para utilizar tanto para el acceso a la API como para renovar los tokens concedidos para la conexión.

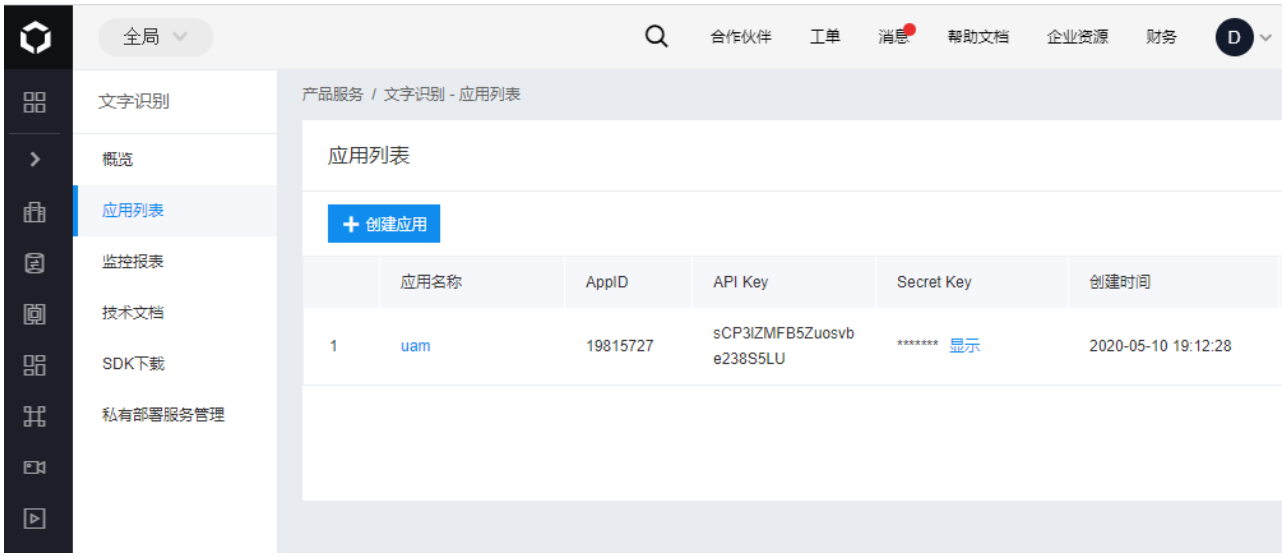

Figura Consola Baidu Cloud[12]

En caso de no disponer de un token temporal, o de que éste esté caducado, para acceder a la API de Baidu se deberá solicitar uno a través de la siguiente petición POST:

https://aip.baidubce.com/oauth/2.0/token

[11] https://cloud.baidu.com/
[12] Actualmente no dispone de interfaz en inglés

En caso de realizar una petición al servicio con un token caducado, este retornará un código de error 111. Se ha codificado una función auxiliar en el proyecto DRF para renovar este token de forma automática si se caduca.

Esta petición requiere 3 variables claves:

grant_type: Será rellenado con un valor fijo *client_credentials.*
client_id: Valor de *API Key* obtenido anteriormente en la creación de la aplicación.
client_secret: Valor de *Secret key* obtenido durante la creación de la aplicación.

En caso de reconocer el id del cliente y de que su clave secreta sea correcta, se concederá una clave de acceso de validez mensual con la siguiente respuesta:


```
{
  "refresh_token": "25.b55fe1d287227ca97aab219bb249b8ab.315360000.1798284651.282335-8574074",
  "expires_in": 2592000,
  "scope": "public wise_adapt",
  "session_key": "9mzdDZXu3dENdFZQurfg0Vz8slgSgvvOAUebNFzyzcpQ5EnbxbF+hfG9DQkpUVQdh4p6HbQcAiz5RmuBA
  "access_token": "24.6c5e1ff107f0e8bcef8c46d3424a0e78.2592000.1485516651.282335-8574074",
  "session_secret": "dfac94a3489fe9fca7c3221cbf7525ff"
}
```


Figura 30 Código de sesión Baidu

Una vez obtenido el token temporal, ya podemos utilizar el servicio de reconocimiento óptico. Baidu ofrece hasta 4 tipos de paquetes OCR para letras:

- Básico: reconoce alrededor de 10.000 caracteres.
- Básico posicionado: Adjunta información adicional sobre posiciones de los caracteres.
- Básico mejorado: hasta 20.000 caracteres.
- Básico mejorado posicionado: Añade la información de posiciones en término de posición en el plano de la imagen junto con la altura y anchura especifica reconocida.

La cuota gratuita de peticiones por día es inversamente proporcional a la calidad del resultado y de la información adicional en el resultado. Dado que la información adicional que podrían agregar estos paquetes no nos genera beneficio a la hora del modelo y que podría encarecer el proyecto, se opta por utilizar el servicio básico con los siguientes parámetros en el cuerpo de la petición POST:

- *Image*: Imagen en formato base64, admite un mínimo de 15px y máximo de 4096px.

- *Lenguaje_type*: *CHN_ENG*, chino e inglés mezclados.

- *detect_direction*: *False*. Permite detectar si la imagen está rotada 90, 180 o 270 grados. En caso de ser activada influiría en el tiempo de respuesta de manera negativa.

- *Probability*: *False*. Informa si se necesita conocer la probabilidad de la predicción sobre el carácter reconocido.

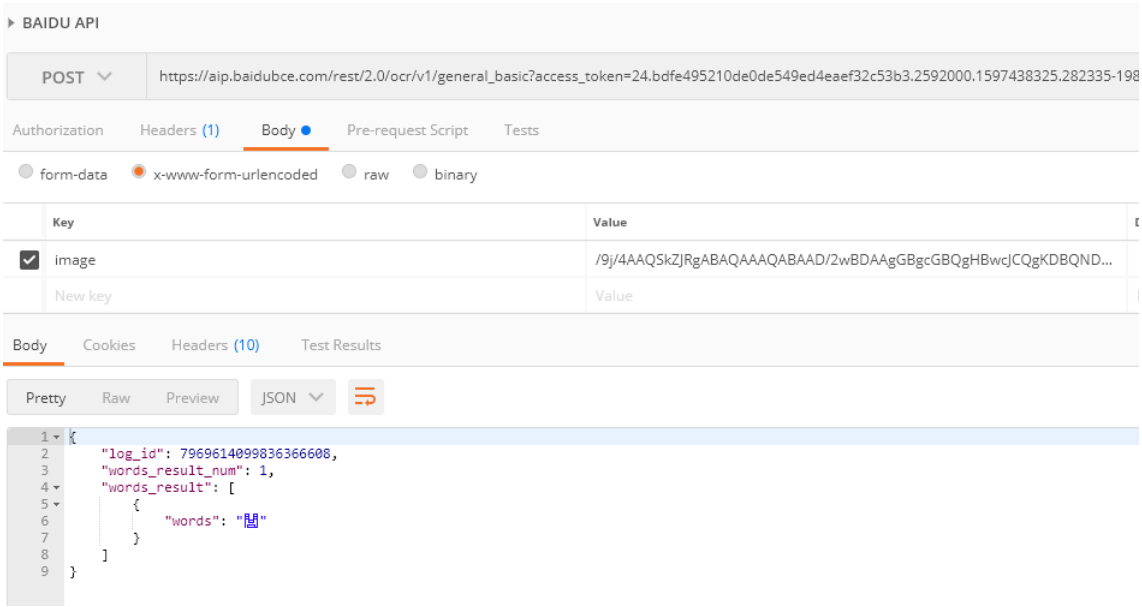

Figura 31 Resultado OCR de BAIDU

Finalmente, tras obtener permiso para realizar las llamadas a su API y conocer bien su funcionamiento, se procede a integrar la funcionalidad de OCR en nuestra API. Cabe recordar que se ha desarrollado junto con la opción de traducción a inglés mediante la biblioteca de traductor de Google. Este servicio es offline ya que se descarga el paquete de idioma directamente al servidor local permitiendo así no incrementar el tiempo de respuesta de las peticiones de los usuarios.

5.3.5. Sistema de fichero Linux

Para poder almacenar las nuevas imágenes proporcionadas por los usuarios y recopilar esta información de manera organizada en algún directorio, se ha empleado el sistema de ficheros Linux. De esta manera se guardan dichos ficheros para que más adelante se puedan aprovechar para mejorar la red neuronal artificial (proceso de refinamiento).

Todos los inputs adicionales se guardarán bajo una carpeta dentro del mismo proyecto. La carpeta tiene la siguiente estructura:

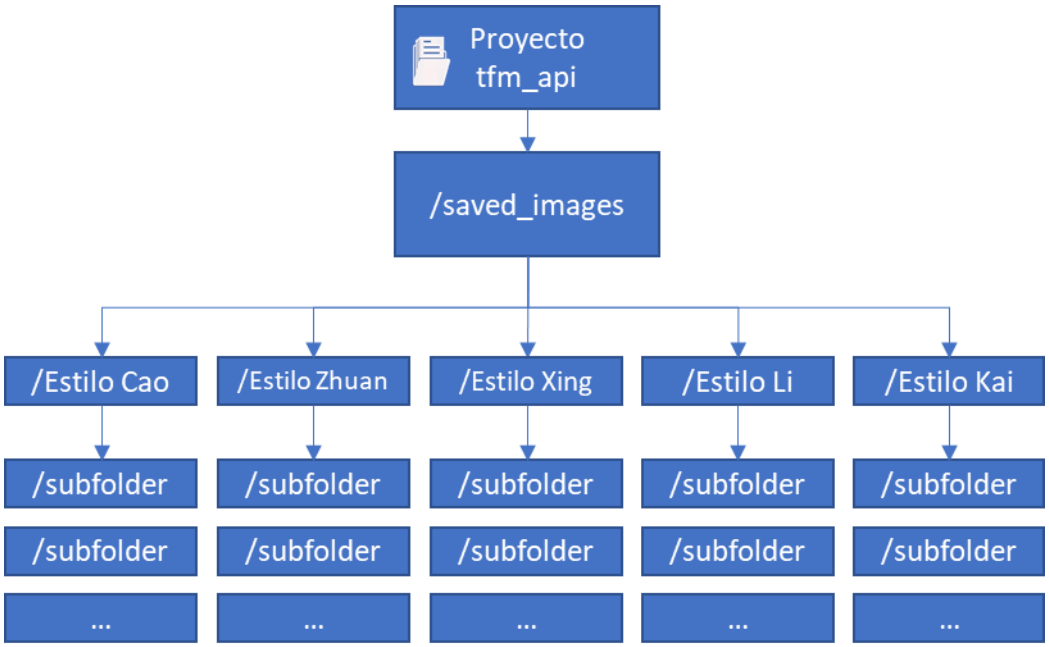

Figura 32 Estructura de guardado

Cada subcarpeta contendrá los distintos inputs que los usuarios envían al servidor para corregir los resultados erróneos de la predicción. Como vemos cada uno de ellos se guarda automáticamente en la subcarpeta que el usuario ha marcado como correcta. Estos inputs también ayudarán a la red neuronal a evolucionar hacia resultados más precisos ante muestras reales.
Cada subcarpeta tendrá el siguiente formato:

Un Timestamp[13] en formato YYYYMMDD_HH24MISS concatenado del email del usuario que es obligatorio de informar.

Una vez acumulado suficientes muestras nuevas, se procede manualmente a volver a entrenar el modelo con estos nuevos inputs para mejorar la precisión del modelo productivo. Por ello se emplea un proceso que recopila todas las imágenes de cada una de las subcarpetas, organizándose por las cinco categorías existentes y aplicando técnicas de *downsampling* o *upsampling* para equiparar en cantidades.

Para evitar prácticas de *adversarial machine learning*, es decir, la aplicación de técnicas para introducir inputs no reales o contaminadas para causar un mal funcionamiento del predictor, se revisará manualmente estos inputs para valorar si corresponde con las correcciones indicadas.

5.4. Front-end

Se ha intentado exponer la red neuronal convolucional artificial creada de una manera accesible para todos los públicos y que no se quede como un mero estudio de laboratorio. Para ello se ha diseñado una interfaz de usuario que permite acceder

[13] Marca temporal de tiempo utilizado en sistema informático

a la red neuronal entrenada mediante una interfaz web. Sin embargo, la finalidad de la API no es sólo para uso exclusivo de esta interfaz, sino que podrá utilizarse con otras aplicaciones. Esta es una forma más visual e intuitiva desde el punto de vista del usuario de interactuar con el modelo.

5.4.1. HTML5, CSS3 y JS

HTML (*HyperText MarkUp Lenguage*) [50] es un lenguaje de marcado destinado a la elaboración de páginas web. Se trata de un estándar que sirve de referencia para el software que conecta con la elaboración de páginas web en sus diferentes versiones. Define una estructura básica y un código (denominado código HTML) que construye los principales bloques de un sitio web. Este es un lenguaje sencillo, el cual utiliza "etiquetas", rodeadas por corchetes angulares (<, >, /) para construir un bloque. Ejemplos de etiquetas HTML son <p> para los párrafos e <img> para las imágenes.

En este proyecto se empleará la quinta versión, pues ofrece elementos y atributos que reflejan el uso típico de los sitios web modernos. Algunos de ellos son técnicamente similares a las etiquetas y , pero tienen un significado semántico, como por ejemplo <nav> (bloque de navegación del sitio web) y <footer>.

Además, a este lenguaje se le puede agregar estilo, el cual puede estar en el propio código, o en un fichero auxiliar, el cual se denomina hoja de estilo en cascada o **CSS** (*Cascading Style Sheets*). En dicho fichero de estilo se pueden definir cuestiones relativas a apariencia como: colores, fuentes, márgenes, tamaños, imágenes de fondo, posicionamientos, entre otros.

Javascript es un lenguaje de programación interpretado, orientado a objetos con un tipado débil y dinámico. Se utiliza principalmente en su forma del lado del cliente (client-side), fomentando la interactividad en el sitio web con los usuarios. Este script a menudo utiliza bibliotecas de terceros, por lo que puede aumentar la funcionalidad del sitio web de manera más que proporcional.

5.4.2. JQuery

jQuery es una librería JavaScript rápida, ligera y con muchas características incluidas en un único archivo de extensión *.js*. Se trata de una librería de código abierto que simplifica la tarea de programar funciones JavaScript y permite agregar interactividad al sitio web.

5.4.3. Boostrap

Bootstrap [51] es un marco de trabajo de código abierto y gratuito de CSS, dirigido al desarrollo de la web. Contiene plantillas de diseño basadas en CSS y en

JavaScript para la tipografía, los formularios, los botones, la navegación y otros elementos de diseño basados en HTML y CSS. Además de las funcionalidades ya comentadas Bootstrap también permite a los desarrolladores sacar más rendimiento a muchos plugins de JQuery personalizados. Esto ofrece aún más libertad para jugar con la interactividad, ofreciendo soluciones fáciles para ventanas de avisos, transiciones, carruseles de imágenes, entre muchas otras.

Se trata de un framework modular que consiste esencialmente en una serie de hojas de estilo LESS que implementan la variedad de componentes de la herramienta. Una hoja de estilo llamada *bootstrap.less* incluye los componentes de las hojas de estilo. Los desarrolladores pueden adaptar el mismo archivo de Bootstrap seleccionando los componentes que deseen usar en su proyecto.

Además, tiene a disposición un sistema de grid, con cuatro variaciones para hacer uso de distintas resoluciones y tipos de dispositivos: teléfonos móviles, formato de retrato y paisaje, tabletas y computadoras con baja y alta resolución (pantalla ancha). Esto permite ajustar el ancho de las columnas automáticamente y consiguiendo así que el diseño web sea adaptable (Responsive Design): Es decir, un diseño que se adapta a cualquier tipo de dispositivo, independientemente de su resolución y tamaño de la pantalla.

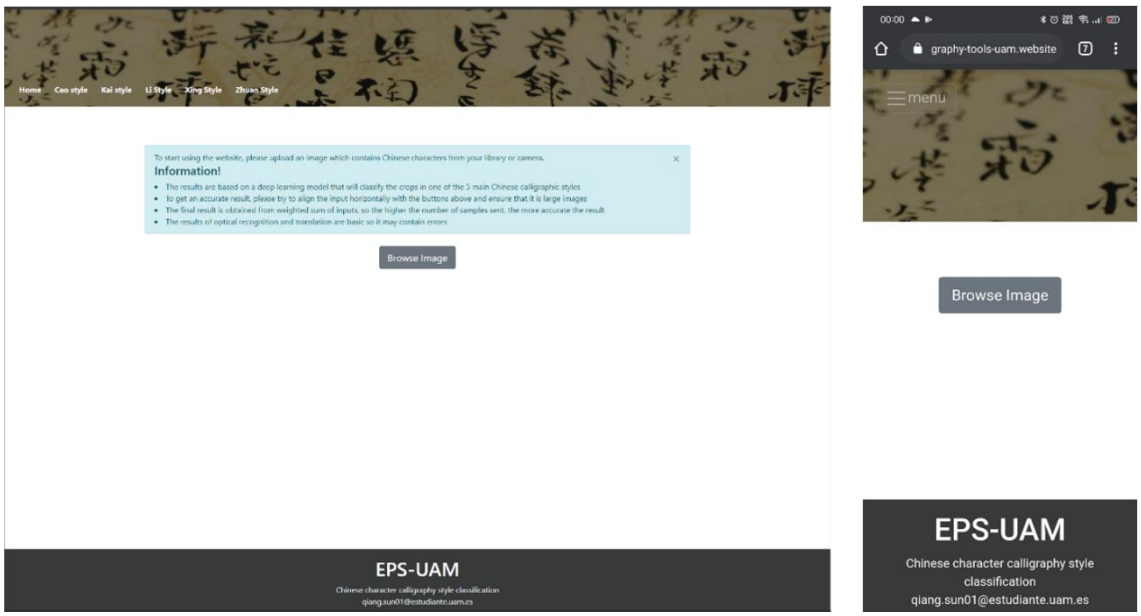

Figura 33 Boostrap Escritorio vs Móvil

5.4.4. Fetch API

Ofrece una definición genérica de los objetos de solicitud y respuesta (y otros elementos relacionados con las solicitudes de la red) de forma asíncrona, esto nos permite enviar las peticiones al servidor.

Fetch API utiliza promesas en sus llamadas; esto se traduce a que devuelve un objeto con dos métodos principales, un *then* donde está englobada la respuesta de la petición, y un *catch* que será invocado cuando se produzca un error.

Para utilizar este método asíncrono únicamente debemos configurar la petición con sus parámetros correspondientes, en este caso se trata de la URL del servidor con el servicio indicado junto con el cuerpo de la petición, compuesta por un array de imágenes codificadas en base64.

5.4.5. Otras librerías utilizadas

De representación:

Sweetalert2: librería que reemplaza las cajas de notificación del JavaScript tradicional, mostrando una bonita, personalizable y dinámica ventana emergente para interactuar con el usuario.

Amcharts: librería para la visualización de datos escrita en TypeScript. Se emplea básicamente para mostrar la gráfica de resultados tras la respuesta de predicción del estilo.

De funcionalidad:

Cropper, librería desarrollada por *FengYuan Chen*, está disponible en forma de código abierto en Github [52]. Esta potente biblioteca permite realizar recortes sobre una imagen y está disponible una vez el usuario sube su imagen para que pueda centrarse en unos caracteres concretos si así lo desea.

5.4.6. Interfaz

En el desarrollo de la página web no se ha utilizado ninguna plantilla prediseñada. Se compone de una página índice principal sencilla, donde se centra toda la lógica para la clasificación del estilo caligráfico, complementada con cinco páginas auxiliares, una por cada estilo caligráfico. En estas cinco páginas de apoyo se muestra una sencilla descripción de cada estilo detallando tanto sus principales características como la línea temporal histórica de dicha caligrafía china.

La estructura sigue la distribución común en cuanto al estilo de una página web: un encabezado coherente seguido del cuerpo y por último el pie de página también común en todas las páginas para promocionar la mayor facilidad de uso a los usuarios.

Escenario de uso:

Posteriormente se describe la secuencia de pasos que realiza un actor sobre la página web para reflejar el flujo exitoso más habitual y esperable de la página web desarrollada.

Realizar una clasificación de imágenes

Actor: Cualquier usuario

Precondición: El usuario tiene acceso a internet y la página web está funcionando correctamente.

Postcondición: El sistema muestra los resultados de la predicción en una gráfica como los posibles reconocimientos y traducciones

Flujo:

Pasos	Ejecución	Servicios
1	Acceder a la URL de la página web	
2	Subir una imagen desde tu sistema local	
3	Seleccionar si se trata de una imagen limpia o compleja	
4	Realizar recortes sobre la imagen seleccionada	
5	Enviar al servidor para la clasificación de las imágenes recortadas	/char_prediction
6	Dibujar la gráfica de resultado mediante la respuesta del servidor	
7	Dibujar la tabla de reconocimiento de caracteres como la traducción al inglés si existe	

Tabla 5-10 Escenario clasificación

Realizar una retroalimentación al modelo para futuro corrección

Actor: Cualquiera

Precondición: Haber realizado una clasificación previamente

Postcondición: Se guarda en el servidor la aportación del usuario

Flujo:

Pasos	Ejecución	Servicios
1	Introduce el email en el input box habilitado para ello	
2	Selecciona el estilo correcto	
3	Pulsa enviar	*/save_image*
4	Recibe feedback del servidor	

Tabla 5-11 Escenario guardar imágenes

6. Pruebas

Este apartado trata de detallar un conjunto de prácticas que tienen lugar durante el proceso de control de calidad para analizar el software desarrollado y verificar su correcto funcionamiento en el horizonte temporal.
A continuación, se listan las pruebas que se han utilizado para validar y verificar el correcto funcionamiento de la aplicación. Los puntos en los que se hace hincapié en este control de calidad son: la funcionalidad, el rendimiento, su compatibilidad con diferentes sistemas, su accesibilidad desde diferentes puntos y la usabilidad.

6.1. Funcionalidad

Pruebas de enlaces
Se comprueba que todos los enlaces de la página web funcionan correctamente. Esto engloba tanto la prueba de navegación a través de los menús como la carga de recursos de la página.

Pruebas de formulario
Existe un formulario donde se solicita el contacto del usuario por medio del correo electrónico que se guarda en la base de datos por si en un futuro necesitamos contactar con algún usuario concreto por su aportación al modelo entrenado. Concretamente se comprueba que este siempre es obligatorio y se valida también que debe contener el formato correcto de un correo electrónico, rechazando inputs no válidos.

6.2. Rendimiento

Se mide la latencia de respuesta media de varias peticiones con distintos números de inputs, todo ello con el objetivo de estudiar el rendimiento que tiene el back-end completo (con el modelo de red neuronal y Baidu API integradas). Por ello la prueba se realiza tanto desde un móvil con red 4G como desde un escritorio con fibra de 100 Mb.

Desde escritorio con fibra:

N.º de recortes enviado	Tiempo medio de respuesta*
1	1.80s
2	2.72s
3	4.10s
4	5.25s
5	6.50s

Tabla 6-1 Rendimiento servidor fibra

Desde un teléfono móvil con red 4G:

N.º de recortes enviado	Tiempo medio de respuesta*
1	4.92s
2	5.40s
3	6.81s
4	8.18s
5	9.50s

Tabla 6-2 Rendimiento servidor 4G

**Los resultados puede variar si durante la petición se necesita renovar la clave de API BAIDU*

Hasta el momento no se ha observado problema con el uso de Memoria o CPU de la máquina contratada. A corto plazo si se desliga más software en el servidor, se medirá los recursos utilizamos por DRF para estudiar si requiere una optimización o simplemente contratar un plan superior.

6.3. Disponibilidad

Esta métrica es el cociente entre el tiempo disponible efectivo y el tiempo total de parada. Esto es de vital importancia para tener disponibles tanto la página web como la API en cualquier momento.

A nivel de hardware del servidor físico, el acuerdo de nivel de servicio (SLA) contratado con DigitalOcean garantiza una disponibilidad del 99.99% [53]. Esto se traduce a que, durante un año completo, como máximo, el servidor estará caído durante 52,56 minutos.

A nivel de software, se ha probado de forma exhaustiva su correcto funcionamiento, controlando todos los posibles errores que pudieran producir una caída de la página por excepción no controladas.

6.4. Compatibilidad

Tal como se ha explicado en puntos anteriores, la API es totalmente independiente de las plataformas o del software ya que consiste únicamente en peticiones.
Por otra parte, la página web sí que depende de los navegadores existentes hoy en día por lo que se procede a hacer una prueba de compatibilidad en los principales navegadores actuales:

Navegador	Versión testeada
Google Chrome	84.0.4147.105 (64 bits)
Firefox	79.0
Safari	13
Microsoft Edge	44.19041.1.0
Internet Explorer	No soportada

Tabla 6-3 Navegadores soportados

6.5. Usabilidad

Se realizan unas pruebas de usuario en el círculo cercano para poder recopilar una serie de métricas que nos permitan evaluar si la página web resulta intuitiva de utilizar o si por el contrario necesitaría alguna mejora.

Las métricas sobre la usabilidad que se han utilizado son las siguientes:

Eficacia. El número de errores cometidos por los usuarios al utilizar la página web fue muy reducido y en todos los casos fueron reproducibles.

Tiempo. El tiempo medio requerido para realizar una petición completa es corta a pesar de que este depende de los números de recortes que quiera hacer el usuario. Se realiza de forma fácil y rápida.

Recuerdo. Basta con utilizar la página web una sola vez para recordar su funcionamiento.

Satisfacción. Casi dos tercios de los usuarios han mostrado que el resultado de la predicción encaja con la realidad. No obstante, existen casos puntuales en los que el input escrito a mano sigue siguiendo difícil de distinguir para el predictor. Este es un inconveniente que se puede mejorar y pulir, pero que nunca desaparecerá del todo ya que depende de la caligrafía del usuario en concreto.

El aporte adicional de este análisis es que permite obtener una visión de alguien que no ha estado trabajando directamente con la página (que no tiene por ello unas concepciones previas y que no conoce cómo funciona internamente) puede aportar alguna idea sencilla de aplicar pero que hubiera pasado inadvertida.

7. Conclusiones

Implementar un modelo de *Machine Learning* parece ser algo que se está convirtiendo esencial para cualquier tipo de programa o página web. Sus objetivos suelen ser la mejora de la experiencia de usuario y, de manera complementaria, aprovechar los recursos al máximo ya sea por categorizar o por facilitar la labor a los usuarios.

Con este mismo objetivo de ayudar a las personas a enfrentar el aprendizaje del chino, nace este proyecto de código libre que consiste en la creación de una página web que ayuda a clasificar a qué estilo caligráfico pertenecen diferentes textos. Esto permite a los aprendices o simplemente los interesados en la escritura de la cultura china conocer no sólo el estilo, sino también detalles acerca de la época histórica a la que pertenece el estilo y a los motivos que fueron motores de los cambios y refinamientos en los estilos. Como se ha explicado anteriormente, la caligrafía está altamente ligada a la evolución de la sociedad china y su estructura. Adicionalmente, gracias a la integración de API de terceros, también se ha logrado trasmitir a los usuarios una inmersión completada ofreciendo el reconocimiento de cada carácter individualmente como su traducción al inglés.

Para conseguir esto, se han probado tanto diversas configuraciones de redes neuronales como varias iteraciones de entrenamiento, realizando siempre los ajustes oportunos. Finalmente se ha conseguido archivar una red neuronal artificial con una precisión del 98.6% empleando la librería de Tensorflow y aprovechando la capacidad de una tarjeta gráfica dedicada. Para conseguir esta precisión se ha empleado en torno a un millón de imágenes extraídas desde fuentes de estilo de sistemas operativos.

7.1. Líneas de trabajo futuro

La página web actual con su back-end es totalmente funcional y válida para ser presentada como un proyecto completo. Sin embargo, existen varias líneas de trabajo que podrían servir para mejorar y/o perfeccionar tanto las funcionalidades como la experiencia de usuario.

Añadir la segmentación de caracteres de forma automática para los inputs complejos. Esto supone un gran reto ya que la escritura china antigua puede ir de forma vertical u horizontal. Adicionalmente se le puede sumar la complejidad de separar caracteres conexos si se trata de escritura manual de los estilos Cao o Xing. Con la segmentación automática se abre un mundo de posibilidades como, por ejemplo, aplicaciones de móviles utilizando bibliotecas de visión artificial y realidad aumentada.

Eliminar la dependencia con Baidu API. Al disponer de las fuentes originales de estilo de las que se han extraído las imágenes individuales se podría estudiar la posibilidad de realizar el reconocimiento de cada carácter de forma individual y limitarse en informar el estilo caligráfico.

Añadir a la página algún *feedback* adicional al usuario para que éste conozca cómo se ha obtenido el resultado final y todo sea más trasparente. De esta manera se incrementaría el interés del usuario, ya que tiene más conocimiento, fomentándose de esta manera el uso de la web.

Reforzar en términos de seguridad en las distintas capas del proyecto como, por ejemplo:

Depuración de inputs externos contra técnicas de *Adversarial Machine Learning*. Se trata básicamente de técnicas que tratan de engañar al modelo para ver cómo éste respondería. Se consigue por ejemplo a través de enviar inputs falsos (caracteres que asimilen caligrafía china sin serlo).

Añadir una capa de usuario a la API para que sólo la puedan utilizar los usuarios autorizados.

8. Bibliografía

[1] Y. Ge y Q. Huo, «A comparative study of several modeling approaches for large vocabulary offline recognition of handwritten Chinese characters,» de *Object recognition supported by user interaction for service robots*, Quebec, Canadá, 2002.

[2] Y.-S. Chen, G. Su y H. Li, «Machine Learning for Calligraphy Styles Recognition,» 2016.

[3] L. Jung y W. Liao, «Applying Deep Convolutional Neural Network to Cursive Chinese Calligraphy Recognition,» de *ew Trends in Computer Technologies and Applications. ICS 2018. Communications in Computer and Information Science*, Singapur, Springer, 2018, pp. 646-653.

[4] I. Jolliffe y J. Cadima, «Principal component analysis: a review and recent developments,» *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, vol. 374, 2016.

[5] «Python Programming,» [En línea]. Available: https://pythonprogramming.net/matplotlib-3d-scatterplot-tutorial/. [Último acceso: Abril 2020].

[6] «Wolfram Language & System,» [En línea]. Available: https://reference.wolfram.com/language/workflow/ChangeTheStyleOfPointsInA2DScatterPlot.html. [Último acceso: Abril 2020].

[7] O. Jena, S. Pradhan, P. Biswal y S. Nayak, «Implementation of Linear Discriminant Analysis for Odia Numeral Recognition,» de *2018 International Conference on Information Technology (ICIT)*, Bhubaneswar, India, 2018, 2018.

[8] A. Martínez y A. Kak, «PCA versus LDA,» *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, nº 2, pp. 228-233, Febrero 2001.

[9] X. He y Partha Niyogi, «Locality Preserving Projections (LPP),» *Advances in Neural Information Processing Systems*, nº 16, 2002.

[10] N. Ahmed, T. Natarajan y K. Rao, «Discrete Cosine Transform,» *IEEE Transactions on Computers* , vol. C23, nº 1, pp. 90-93, 1974.

[11] M. Faundez-Zanuy, «On-line signature recognition based onVQ-DTW,» *The Journal of the Pattern Recognition society*, vol. 40, pp. 981-992, 2007.

[12] P. Senin, Dynamic Time Warping Algorithm Review, Jour, 2009.

[13] L. Rabiner y B. Juang, «An introduction to hidden Markov models,» *IEEE ASSP Magazine*, vol. 3, nº 1, pp. 4-16, 1986.

[14] C. Lee y D. A. Landgrebe, «Feature Extraction Based On Decision Boundaries,» *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, nº 4, pp. 388-400, 1993.

[15] «Text Detection and Recognition in Imagery: A Survey,» *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, nº 7, pp. 1480-1500, July 2015.

[16] K. Hamad y M. Kaya, «A Detailed Analysis of Optical Character Recognition Technology,» *International Journal of Applied Mathematics, Electronics and Computers,* vol. 4, pp. 244-249, 2016.

[17] A. Shinde y D. Chougule, «Text Pre-processing and Text Segmentation for OCR,» *International Journal of Computer Science Engineering and Technology,* pp. 810-812, 2012.

[18] O. Sharma, M. Ghose y K. Shah , «An improved zone-based hybrid feature extraction model for handwritten alphabets recognition using Euler Number,» *International Journal of Soft Computing and Engineering,* vol. 2, nº 2, pp. 504-508, May 2012.

[19] R. Brunelli, Template Matching Techniques in Computer Vision: Theory and Practice, WILEY, 2009.

[20] I. I. Systems, «Support Vector Machines,» *Marti A. Hearst,* vol. 13, nº 4, 1992.

[21] «Xeridia,» [En línea]. Available: https://www.xeridia.com/blog/redes-neuronales-artificiales-que-son-y-como-se-entrenan-parte-i. [Último acceso: Abril 2020].

[22] N. Altman, An Introduction to Kernel and Nearest Neighbor, Ithaca: Cornell University, 1991.

[23] Y.-S. Chen, G. Su y H. Li, «Machine Learning for Calligraphy Styles Recognition,» 2016.

[24] I. Goodfellow y Y. Bengio, «Softmax Units for Multinoulli Output Distributions,» *Deep Learning. MIT Press.,* pp. 180-184, 2016.

[25] «Analytics Vidhya,» [En línea]. Available: https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/. [Último acceso: Abril 2020].

[26] L. Breiman, «Random Forests,» *Machine Language,* vol. 45, nº 1, 2001.

[27] «Analytics Vidhya,» [En línea]. Available: https://www.analyticsvidhya.com/blog/2020/05/decision-tree-vs-random-forest-algorithm/. [Último acceso: Agosto 2020].

[28] D. OPPENHEIMER, «DevOps,» [En línea]. Available: https://devops.com/five-challenges-of-machine-learning-devops/. [Último acceso: 06 06 2020].

[29] F. Roy Tomas , Architectural Styles and the Design of Network-based Software Architectures, Irvine: University of California, 2000.

[30] b. Otsu, «A Threshold Selection Method from Gray-Level Histograms,» *IEEE Transactions on Systems, Man, and Cybernetics,* vol. 9, nº 1, pp. 62-66, Enero 1979.

[31] unipython, «unipython,» [En línea]. Available: https://unipython.com/umbralizacion-una-imagen/. [Último acceso: 30 01 2021].

[32] K. b. Satoshi Suzuki, «Topological structural analysis of digitized binary images by border following,» *Computer Vision, Graphics, and Image Processing,* vol. 30, nº 1, pp. 32-46, 1985.

[33] F. Pukelsheim, «The Three Sigma Rule,» *American Statistician,* pp. 88-91, 1994.

[34] D. Kernler, «A visual representation of the Empirical (68-95-99.7) Rule based on the normal distribution».

[35] FontForge, «Fontforge,» [En línea]. Available: https://fontforge.org/en-US/project/. [Último acceso: 02 03 2019].

[36] Y. Wen, «Chinese calligraphy: Character style recognition based on Full-page document,» de *Trabajo fin de master*, Madrid, UAM, 2018, pp. 48-53.

[37] Y. Zhang, «Deep Convolutional Network for Handwritten Chinese Character Recognition,» 2015.

[38] T. P. C. S. Dean Rizki Hartawan, «Disaster Victims Detection System Using Convolutional Neural Network (CNN) Method,» *International Conference on Industry 4.0*, 2019.

[39] Uniqtech, «Understand the Softmax Function in Minutes,» [En línea]. Available: https://medium.com/data-science-bootcamp/understand-the-softmax-function-in-minutes-f3a59641e86d. [Último acceso: 15 05 2020].

[40] N. V. Chawla, K. W. Bowyer, L. O. Hall y W. P. Kegelmeyer, «SMOTE: Synthetic Minority Over-sampling Technique,» *Journal of Artificial Intelligence Research 16,* pp. 328-331, 2002.

[41] Keras, «Documentación sobre Keras, preprocesamiento de imágenes,» [En línea]. Available: https://keras.io/api/preprocessing/image/. [Último acceso: 06 03 2020].

[42] B. Li, «Convolution Neural Network for Traditional Chinese Calligraphy Recognition,» 2016.

[43] J. Zhang , L. Guo , S. Yang , X. Sun y X. Li , «Detecting Chinese calligraphy style consistency by deep learning and one-class SVM,» *IEEE*, 2017.

[44] P. Gao , G. Gu , J. Wu y B. Wei , «Chinese calligraphic style representation for recognition,» *Document Analysis and Recognition,* vol. 20, nº 1, 2017.

[45] DRF, «Guía Django Rest Framework,» [En línea]. Available: https://www.django-rest-framework.org/api-guide/. [Último acceso: 02 05 2020].

[46] Certbot, «Auto updating certbot and letsencrypt,» [En línea]. Available: https://certbot.eff.org/docs/contributing.html#updating-certbot-auto-and-letsencrypt-auto. [Último acceso: 30 04 2020].

[47] W3Techs, «Web Technology Surveys. Usage statistics of Apache,» [En línea]. Available: https://w3techs.com/technologies/details/ws-apache. [Último acceso: 11 05 2020].

[48] Ubuntu, «UFW - Uncomplicated Firewall,» [En línea]. Available: https://help.ubuntu.com/community/UFW. [Último acceso: 28 04 2020].

[49] Baidu Cloud, «Manual Baidu Cloud OCR,» [En línea]. Available: https://cloud.baidu.com/doc/OCR/s/dk3h7y5vr. [Último acceso: 26 06 2020].

[50] w3schools, «Documentación y ejemplos sobre el uso de HTML5, CSS y JS,» [En línea]. Available: https://www.w3schools.com/. [Último acceso: 30 04 2020].

[51] Boostrap4, «Documentación de instalación y configuración de Boostrap 4,» [En línea]. Available: https://getbootstrap.com/docs/4.5/getting-started/introduction/. [Último acceso: 30 01 2020].

[52] F. Cheng, «CropperJS,» [En línea]. Available: https://github.com/fengyuanchen/cropperjs/blob/master/README.md. [Último acceso: 20 09 2019].

[53] DigitalOcean, «Políticas Droplet DigitalOcean,» [En línea]. Available: https://www.digitalocean.com/docs/platform/droplet-policies/. [Último acceso: 05 03 2020].

[54] A. W. y. C. W. Danilo Sato, «martinfowler,» 19 Septiembre 2019. [En línea]. Available: https://martinfowler.com/articles/cd4ml.html. [Último acceso: 01 08 2020].

[55] C. Suen, «Character recognition by computer and applications,» de *Handbook of pattern recognition and image processing*, 1986, pp. 569-586.

[56] D. Kernler, «A visual representation of the Empirical (68-95-99.7) Rule based on the normal distribution.».

[57] C. Y. H. W. Y. W. S. Z. W. H. a. J. L. Xinyu Zhou, «EAST: An Efficient and Accurate Scene Text Detector,» *Computer Vision and Pattern Recognition*, 2017.

9. Apéndices

9.1. A - Configuración del PC para entrenamiento

Para entrenar el modelo expuesto en el punto 5.1, se ha empleado la siguiente configuración de máquina local:

Sistema: Windows 10
CPU: AMD RYZEN 3800XT 8 núcleos 16 hilos
GPU: MSI RTX 2070 Gaming Z (N.º núcleos CUDA 2304, Memoria: 8GB)
Almacenamiento principal: NVME m.2 SN750 WD 500GB

9.2. B - Aplicaciones

Aplicación empleada	Uso	Versión
FontForge	Extracción de input	Windows 2020-03-14
Anaconda	IDE para el desarrollo	1.9.2
WinSCP	Subir el modelo entrenado al servidor	5.17
Postman	Test de servicios web	5.5.5

Tabla 9-1 Aplicaciones utilizadas

Entorno instalado en Windows para el entrenamiento

Biblioteca instalada	Uso	Versión
python	Lengua principal utilizado	3.6.9
matplotlib	Imprimir gráficas de resultado	3.1.1
jupyter_client	IDE para el desarrollo	5.3.4
numpy	Tratamiento de funciones numéricos	1.17.4
Opencv	Visión artificial para tratamiento de imágenes	3.4.2
scikit-learn	Funciones de aprendizaje automático	0.23.1
Tensorflow-gpu	Machine learning	2.0.0
keras-gpu	Sub-librería de tensorflow para redes neuronales	2.2.4
cudatoolkit	Permite construir aplicaciones aceleradas en la GPU	10.0.130
cudnn	Librería para cudatoolkit asociado a redes neuronales	7.6.5

Tabla 9-2 Librerías en máquina de entrenamiento

Entorno instalado en entorno Ubuntu – servidor para los DRF:

Biblioteca instalada	Uso	Versión
python	Lengua principal utilizado	3.6.9
numpy	Tratamiento de funciones numéricos	1.17.4
Opencv	Visión artificial para tratamiento de imágenes	3.4.2
Tensorflow-gpu	Machine learning	2.0.0
keras-gpu	Sub-librería de tensorflow para redes neuronales	2.2.4
Django	Framework principal	2.2.3
Djangorestframework	REST sobre Django	3.11.0
Googletrans	Traducción al inglés	2.4.0
Requests	Tratamiento de peticiones	2.22.0

Tabla 9-3 Librerías empleadas en servidor

9.3. C - Servicios web

/char_prediction

Este *endpoint* recibe un array de imágenes en formato base 64 y retorna el resultado de la predicción en porcentajes, adicionalmente se intentará reconocer el contenido como traducirlo a inglés:

Por ejemplo, para el sinograma [人] "persona" tenemos la siguiente petición:

Imagen original: 

Convertido en base64:

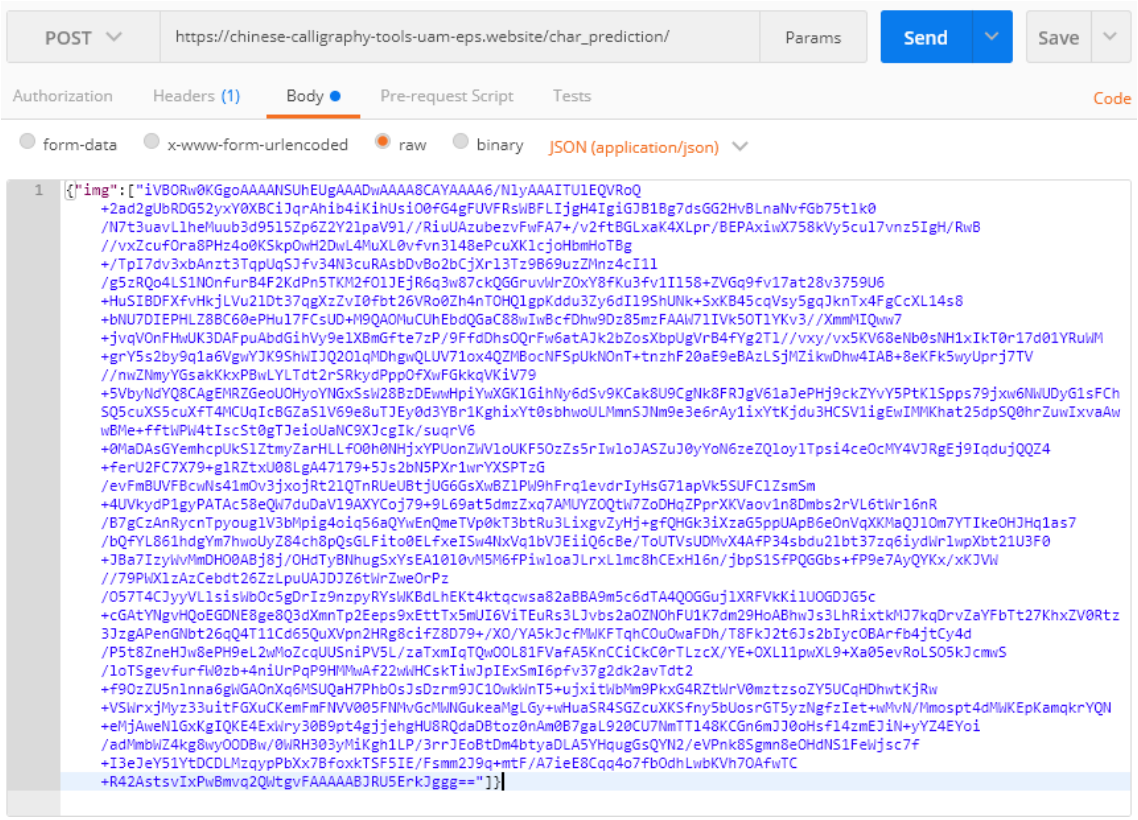

Figura 34 Codificación base64 petición

Resultado de la petición:


```
{
    "final_result": "Estilo_Kai",
    "details": {
        "Estilo_Cao": 0.0924,
        "Estilo_Kai": 0.5512,
        "Estilo_Li": 0.0445,
        "Estilo_Xing": 0.3081,
        "Estilo_Zhuan": 0.0038
    },
    "CN": [
        "人"
    ],
    "EN": [
        "people"
    ]
}
```


Figura 35 Resultado petición char_prediction

Se puede apreciar que, al tratarse de un carácter chino muy simple, la probabilidad de resultado no es alta en ninguno de los cinco estilos.

/save_image

El contenido de la petición es idéntico al paso anterior, salvo la excepción de dos campos adicionales que son el email del remitente y la corrección del estilo.

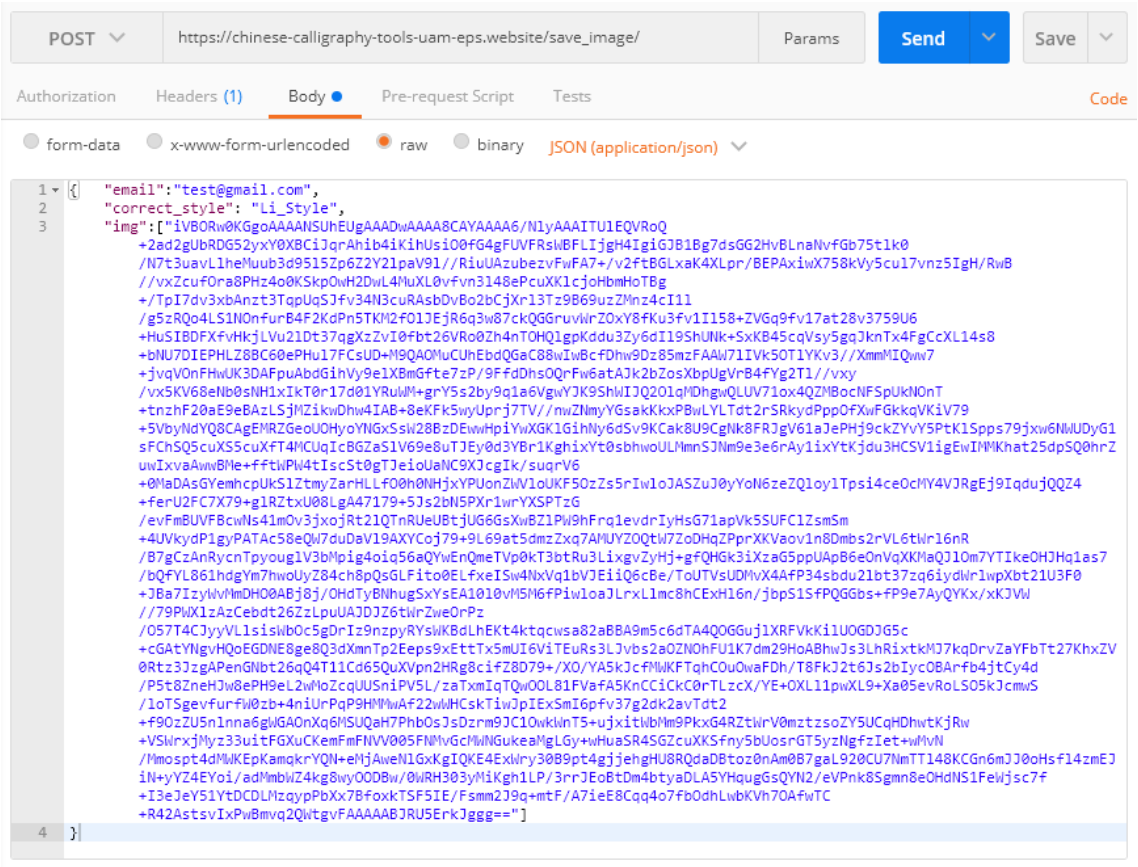

Figura 36 Guardar input adicional

Resultado de la petición:

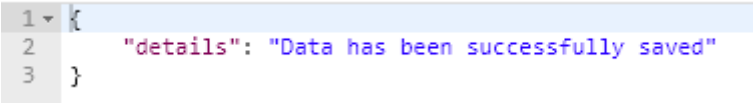

Figura 37 Resultado del guardado save_image

Por lado del servidor:

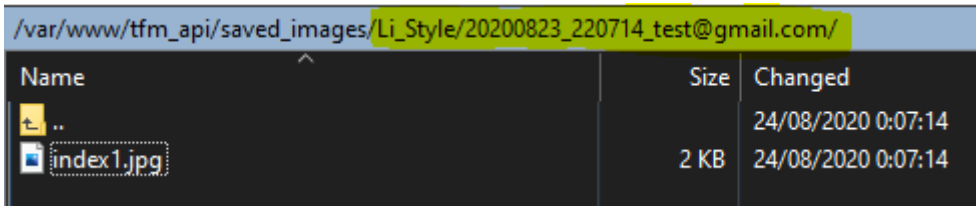

Figura 38 Fichero guardado en el servidor

9.4. D - Organización del proyecto.

Un aspecto destacable en cada una de las fases de este proyecto realizado son los roles y labores que existen en el proyecto. De esta forma, se han podido adquirir conocimientos en las labores que ejerce cada tipo de profesional ya que el ciclo completo del proyecto lo ha realizado únicamente el estudiante – Qiang Sun.

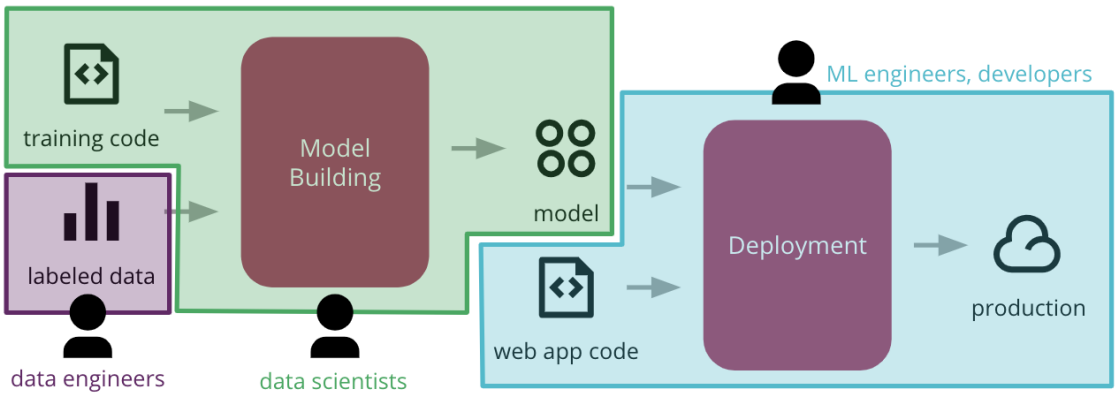

Figura 39 Roles & fases [54]

De la imagen anterior, se pueden extraer los roles que forman la estructura interna del proyecto y cuya visión se ha incorporado al proyecto. A continuación, se detallan brevemente estos roles con sus tareas asignadas:

Data engineers o ingenieros de datos: Responsables del diseño, desarrollo y mantenimiento de los sistemas de procesamiento de datos.

Data scientists o científicos de datos: Se encargan de la parte de entrenar y construir el modelo de la forma óptima posible.

Developers o desarrolladores: Programador con un perfil técnico con amplio conocimiento sobre la solución a desarrollar.

Cabe destacar que también se han ejercido labores de arquitecto informático en la definición de la solución completa *end-to-end*. Este rol se manifiesta, por ejemplo, en la toma de decisión de qué tipo de tecnología, medios y protocolos utilizar.