UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Diseño de un sistema de identificación de personas

Presentado por Víctor de Castro Hurtado
en Universidad de Burgos — 23 de mayo de 2018
Tutor: César Represa

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

D. César Represa, profesor del departamento de Ingeniería Electromecánica, área de Tecnología Electrónica.

Expone:

Que el alumno D. Víctor de Castro Hurtado, con DNI 71289378G, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado Diseño de un sistema de identificación de personas.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 23 de mayo de 2018

Vº. Bº. del Tutor:

D. César Represa

Resumen

Se pretende diseñar un sistema de **identificación de personas** a través de un programa de **reconocimiento facial** con rostros previamente incluidos en una base de datos. El sistema de desarrollará en **Python** y utilizando técnicas de **Machine Learning**.

Descriptores

python, inteligencia artificial, IA, reconocimiento facial, cámara, tiempo real, identificación, aprendizaje automático, aprendizaje profundo, entrenamiendo de redes, visión artificial

Abstract

The main goal is to develop a **people identification** system with a **facial recognition** program and faces included on a database. This system will be developed in **Python** using **Machine Learning** techniques.

Keywords

python, artifical intelligence, AI, facial recognition, camera, real time, identification, machine learning, deep learning, network training, computer vision

Índice general

Índice de figuras

Índice de tablas

Introducción

1.1. Resumen

El proyecto consiste en una aplicación que, dadas dos imágenes (una obtenida desde una cámara y la otra alojada localmente), sea capaz de distinguir si en ambas imágenes se encuentra la misma persona utilizando técnicas de machine-learning.

1.2. Repositorio

Se puede encontrar el proyecto entero en el repositorio [22] habilitado para ello.

1.3. Estructura

La estructura que sigue nuestro proyecto y que se puede encontrar en Github es la siguiente:

- Carpeta **Other**

  En esta carpeta se encuentra el material ajeno al proyecto en python, y que, por lo tanto, no afecta a la ejecución del programa. En dicha carpeta se encuentra:

- Carpeta **Images**

  Contiene diversas imágenes relacionadas con el desarrollo del proyecto, explicaciones sobre algunos aspectos cuya descripción con palabras podía no quedar demasiado clara, etc.

  La mayoría de estas imágenes se han realizado de forma 'casera' por nuestra cuenta, aunque pueden encontrarse también algunas imágenes obtenidas de internet (en cuyo caso se especificará su orígen).

  - Carpeta **ClassDiagram**

    Esta carpeta contiene los diagramas de clases, tanto los que representaban la estructura inicial del proyecto como el resultado final. En ellos se incluye también la estructura de paquetes, cómo se relacionan entre ellas, etc. La explicación sobre ello se puede encontrar en el Anexo C-Diseno del documento anexos.pdf.

  - Carpeta **flowCharts**

    Esta carpeta contiene los diagramas de flujo del programa, representando cuales son los pasos que sigue, posibles ramas y desenlaces que puede tener. Como en el caso de los diagramas de clases, se incluyen los diagramas con la idea original y el resultado final. La explicación sobre ello se puede encontrar en el Anexo C-Diseno del documento anexos.pdf.

  - Carpeta **theoric concepts**

    Esta carpeta contiene imágenes relacionadas con los conceptos teóricos de la sección 3.- Conceptos teóricos 2.2.

  - Carpeta **results**

    Esta carpeta contiene imágenes sobre la ejecución del programa, para poder observar cómo sería el resultado tanto si se ejecuta sin problemas, pero no identificando a la persona que tiene delante de la cámara, como si se ejecuta sin problemas

identificándola, incluso posibles fallos.

- Fichero **TFG-FacialRecognition-Memoria.pdf**

  Contiene esta memoria en formato .pdf.

- Fichero **TFG-FacialRecognition-Anexos.pdf**

  Contiene los anexos de la memoria, en formato .pdf.

- Carpeta comprimida **TFG-FacialRecognition.zip**

  Contiene esta memoria y los anexos, divididos en diferentes ficheros .tex.

- Carpeta **sources**

  En esta carpeta se puede encontrar material que afecta a la ejecución del programa, pero que no es código ejecutable como tal. En dicha carpeta encontramos:

  - Carpeta **dataset**

    En esta carpeta podemos encontrar las imágenes originales que guardamos al principio en nuestra base de datos. Como se puede observar, son imágenes de cuerpo entero la mayoría de ellas, por lo tanto no están ajustadas a la posición de la cara de la persona, por lo que necesitamos crear (interna y automáticamente) la siguiente carpeta:

  - Carpeta **facesDataset**

    Contiene las imágenes anteriores, pero exclusivamente con la cara que haya podido encontrar en la imágen, recortando dicha imágen para eliminar elementos ajenos al rostro que puedan dificultar el reconocimiento.

- Carpeta **xml**

  Contiene dos ficheros .xml que son los encargados de encontrar las partes identificables de un rostro en la imágen. Dichos ficheros se proporcionan gratuitamente con fines educativos en un repositorio [78].

- Carpeta **recognizer**

  Contiene ficheros que utilizamos durante la ejecución del programa para visualizar datos sobre la persona reocnocida o cargar la imágen correcta de la base de datos a partir de un diccionario.

  - Fichero **dictionary-ID-labels.txt**

    Este fichero se crea automáticamente tras entrenar la red, y contiene el diccionario que relaciona una ID automática a cada imágen de la que ha conseguido obtener un rostro. Se podría evitar usar este fichero si se utilizara directamente el diccionario en el programa, pero eso supondría tener que entrenar la red todas las ejecuciones, y si no se han añadido imágenes nuevas es tiempo y recursos perdidos, de modo que se ha optado por crear un fichero para que el diccionario persista a cada ejecución.

  - Fichero **info.txt**

    Este fichero se ha creado sólo para mostrar información de la persona que se haya reconocido durante la ejecución. Para mostrar esta información consultar el anexo E-Manual-usuario. Esta información se puede actualizar a mano actualmente, aunque en un futuro se podría incluir en la base de datos para evitar accesos no permitidos.

  - Fichero **trainedData.yml**

    Este fichero contiene las principales características de los rostros detectados cuando entrenamos la red. Es el resultado de dicho entrenamiento, y por lo tanto se crea automáticamente. Las características que obtiene son las mismas que las

que se pueden encontrar en los ficheros .xml de dicha carpeta.

- Fichero **default.png**

  Esta imágen es la que carga el programa por defecto cuando no se ha conseguido superar el umbral de reconocimiento de un rostro en una imágen. Normalmente se conseguirá superar dicho umbral, aunque el reconocimiento falle, pero en caso de que no se hayan podido encontrar rostros, no se haya entrenado la red, o se le pase un fichero que no sea una imágen, el programa cargará esta imágen por defecto.

- Carpeta **src**

  En esta carpeta se encuentra el código de nuestro programa. Está dividido en diferentes clases que vemos a continuación:

  - Fichero **Main.py**

    Contiene el método principal de la ejecución del programa. Es donde se le pregunta al usuario si quiere entrenar la red, seleccionar una imágen desde fichero o desde la cámara, llama a todos los demás métodos de otras clases para comparar las imágenes, generar la interfaz y mostrar resultados. Se pueden ver la estructura en el anexo D-Manual-programador, y el flujo de programa en el anexo C-Diseno.

  - Fichero **Util.py**

    Contiene todos los métodos de utilidad, desde pedir una simple cadena de texto por teclado hasta inicializar la cámara y guardar la imágen que queramos para compararla, pasando por cargar los diferentes ficheros .xml y .txt que usamos o mostrar los menús por pantalla. Se pueden ver todos los métodos en el anexo D-Manual-programador.

  - Fichero **GUI.py**

Contiene todo lo relacionado con la interfaz gráfica, desde el panel central, donde se muestra toda la información y las imágenes obtenidas, hasta el menú que nos permite seleccionar una imágen desde fichero en vez de desde la cámara. Dicha interfaz de ha realizado con tkinter. Se puede ver cada uno de los métodos y su funcionalidad en el anexo D-Manual-programador.

- Fichero **CompareImages.py**

  Realiza la comparación entre la imágen que acabamos de obtener (ya sea mediante captura con la cámara o desde fichero), con todas las de la base de datos. Se puede ver el funcionamiento exacto de esta función en el anexo D-Manual-programador.

- Fichero **trainer.py**

  Esta clase se encarga exclusivamente de entrenar la red a partir de unas imágenes fuente, que se encuentran en la carpeta faces-Dataset. La funcionalidad completa se puede observar en el anexo D-Manual-programador.

- Fichero **README.txt**

  Fichero con información general sobre el proyecto, instrucciones básicas y modo de empleo.

Objetivos del proyecto

2.1. Resumen

El proyecto pretende servir como forma de identificación de personas que, teniendo una imágen de la persona en cuestión previamente obtenida en la base de datos, y su información básica (nombre, edad, lugar de nacimiento y profesión), se puede identificar a dicha persona cuando se pone delante de la cámara utilizando técnicas de machine-learning y computer vision.

2.2. Aplicación real

Esto llevado a la práctica podría servir como sistema de **identificación de criminales** utilizando como medio las **cámaras** instaladas en **lugares públicos**.

Para ello, se deberían tener las imágenes de dichos criminales almacenadas en la base de datos, todas ellas con las mismas características (tomadas de un **registro oficial**, como pueden ser imágenes de DNI, registros de su paso por comisaría en otras detenciones, etc).

Teniendo las personas que queremos encontrar ya introducidas en nuestra base de datos, se procedería a ejecutar un **rastreo**, ejecutando el programa en un **servidor** con capacidad suficiente como para analizar cada rostro de todas las imágenes en tiempo real, sobre las cámaras de ayuntamientos, bancos, semáforos, comisarías, etc. de la ciudad.

Por supuesto, este sistema de reconocimiento a **gran escala** y en **tiem-**

po real (recordemos que está recopilando información de todas las cámaras disponibles en la ciudad, a la vez que está obteniendo las características de los rostros que encuentra en cada una de ellas, comparando dichas características con las obtenidas al entrenar previamente la red), tiene un **alto coste computacional** y de recursos, de manera que sería necesario ejecutar el programa principal en un servidor con unos **requisitos técnicos muy altos**, además de una infraestructura adecuada para poder transmitir toda esa información sin retrasos (lo que implica, de nuevo, unos altos requisitos técnicos).

Si alguno de estos criminales aparece en alguna de estas cámaras, el sistema le reconocería y mostraría una interfaz con la imágen capturada desde la cámara y la imágen de la base de datos con la que la relaciona, mostrando además la cámara con la que ha realizado el reconocimiento, para que sea más sencillo **localizar** a ese criminal **en la zona de la ciudad** que corresponda.

Estamos hablando de una aplicación que debería ser llevada a cabo **a nivel local** (en un municipio o ciudad no demasiado grande), porque, como hemos comentado, cuanto **mayor sea la zona a cubrir** (y por lo tanto el número de cámaras utilizadas), **mayores serán los requisitos técnicos necesarios**.

Otra posible aplicación que se podría obtener de este proyecto sería el **buscar personas desaparecidas o secuestradas**. Utilizando el mismo método, lo único que habría que variar sería la base de datos utilizada: de criminales a gente desaparecida.

Lo mismo que se podría adaptar a otros ámbitos simplemente teniendo una base de datos diferente para cada uno de los casos. En ese sentido, el **coste de adaptación** a la variedad de casos de usos sería **mínimo**, siendo el coste inicial el mayor de todos: un servidor principal que se encargue de ejecutar el programa, recopilando los datos obtenidos de todas las cámaras.

En la realización de este proyecto no se ha llegado a un nivel tan lejos, primero porque **no se disponía de los medios necesarios** para cumplir con los requisitos del hardware (el servidor principal con requisitos técnicos elevados que comentábamos anteriormente), y segundo porque **no se disponía de una base de datos suficientemente amplia** como para mostrar esta funcionalidad completa que se mencionaba en párrafos anteriores.

Por lo tanto, se ha optado por describir sus posibles y futuras implicaciones en esta memoria, mientras se implementaba una primera versión del proyecto, acorde al presupuesto, tiempo y recursos de los que se disponía.

En esta primera versión se tiene una base de datos con unas cuantas imágenes de personajes famosos (representaría en la aplicación final a los criminales), entre los que se incluye una foto nuestra para poder obtener un resultado positivo en el análisis en tiempo real.

Con esta base de datos improvisada, se entrena la red, identificando las características de las personas que tenemos, almacenando dichas características para evitar entrenar la red cada vez que se trate de identificar un rostro (ahorrando tiempo y recursos).

Tras esto, ejecutamos la parte del programa encargada del reconocimiento propiamente dicho, por lo que nos colocamos delante de la cámara y tomamos una captura (esta parte también se ha visto simplificada con respecto a la aplicación final en la que se tomarían dichas capturas con varias cámaras, distribuidas a lo largo de la ciudad).

Por último, el programa reconoce automáticamente las características del rostro de la imágen capturada, compara dichas características con las que teníamos guardadas de los famosos, y nos da una estimación de la persona a la que más nos parezcamos.

En una aproximación más real y 'comercial', habría que adaptar los parámetros de comparación y muestreo de los resultados para evitar mostrar los resultados que no superen un umbral, de tal manera que, si no coincide en un alto porcentaje con la persona que estamos buscando, no muestre nada, evitando de esta manera falsos positivos.

Conceptos teóricos

3.1. Conceptos

Algunos de los conceptos que vamos a necesitar comprender para entender el proyecto son los siguientes:

- **Inteligencia Artificial**

  La inteligencia artificial (IA) según searchdatacenter [92], es 'la simulación de procesos de inteligencia humana por parte de máquinas, especialmente sistemas informáticos. Estos procesos incluyen el aprendizaje (la adquisición de información y reglas para el uso de la información), el razonamiento (usando las reglas para llegar a conclusiones aproximadas o definitivas) y la autocorrección.'

- **Aprendizaje Automático**

  El aprendizaje automático (o más conocido por su nombre en inglés: Machine Learning). es la parte de la inteligencia artificial que consiste en conseguir que **un ordenador actúe de manera automática**, sin necesidad de programarlo para ello.

- **Analítica predictiva**

  La analítica predictiva es la ciencia que analiza los datos para intentar predecir datos futuros relacionados. En nuestro caso, a partir de unos datos originales, se analizan mediante el aprendizaje automático para crear nuevos modelos predictivos.

- **Aprendizaje profundo**

  El aprendizaje profundo (o más conocido por su nombre en inglés: Deep Learning), es una rama del aprendizaje automático que automatiza la analítica predictiva.

  En el siguiente artículo [84] se puede encontrar un ejemplo de una red convolucional desarrollada en python.

- **Red neuronal artificial**

  Las [119]**redes neuronales artificales** son un modelo computacional basado en nodos y enlaces, donde **cada nodo corresponde con una neurona**, y los enlaces corresponden a las relaciones entre dichos nodos. Cada nodo almacena información independiente del resto de nodos, de manera que nuestro sistema se divide en función de la información independiente que tenga.

  Dados unos **datos de entrada** (nodos 'input'), se forman relaciones entre ellos para establecer los aspectos comunes y las diferencias.

  De estas relaciones nos quedamos con los **nodos intermedios** de la red (nodos 'hidden'), que son los que no ve el usuario (programador), y que puede haber tantas capas de nodos 'hidden' como hagan falta.

  Por último, la última de estas capas se conforma de los **nodos de salida** que vamos a obtener como **resultado del entrenamiento** de la red (nodos 'output').

  Este proceso se puede observar en la imágen 3.1.

  El objetivo de la red neuronal es resolver los problemas de la misma manera que el cerebro humano, aunque de forma más abstracta y con variantes de entre miles a millones de neuronas (nodos).

  Algo importante cuando hablamos de redes neuronales, inteligencia artificial y machine learning, es que **no se puede garantizar nunca su grado de éxito** después de realizar el auto-aprendizaje. Para obtener unos resultados más o menos homogéneos, se necesitan miles y miles de ejecuciones.

- **Red neuronal convulacional**

  Las redes neuronales convolucionales [120] son un subtipo de las redes neuronales artificiales, que se basan en las versiones biológicas de los perceptrones multicapa (Multi Layer Perceptron en inglés) [118], y consisten en **varias capas** o 'layers', en cada una de las cuales existen una serie de **nodos con información**. Estos nodos se conectan con cada uno de los nodos de la capa siguiente (a diferencia de las redes neuronales normales, que simplemente crean un nodo de la capa siguiente a partir de dos de la capa anterior), quedando **todos los nodos interconectados entre una capa y otra**. Este tipo de redes son especialmente útiles en sistemas de computer vision y deep learning, ya que los nodos se retroalimentan entre si, dando lugar a mejores soluciones.

  Este proceso se puede observar en la siguiente imágen 3.2. Para obtener más información sobre este tipo de redes se puden consultar las siguientes referencias [94], [51].

- **Tipos de Entrenamiento**

  Los diferentes tipos de entrenamiento de una red son:

  - **Aprendizaje supervisado**

    En este tipo de aprendizaje los datos tienen etiquetas, las cuales se usan para obtener nuevos patrones relacionados con los datos de dichas etiquetas.

  - **Aprendizaje no supervisado**

    En este tipo de aprendizaje los datos no tienen etiquetas, de manera que los nuevos patrones se obtienen y clasifican en función de similitudes y/o diferencias con el resto de datos.

  - **Aprendizaje por refuerzo**

    En este sistema de aprendizaje, como en el no supervisado, no hay etiquetas, pero a diferencia de este, tras realizar un modelo o patrón, el sistema recibe retroalimentación para ajustar dicho patrón.

- **Visión Artificial**

  La visión artificial [117] (o más conocido por su nombre en inglés: Computer Vision) pretende **simular el comportamiento del ojo humano**, permitiendo adquirir, procesar, analizar y comprender las imágenes mediante la clasificación y segmentación de las mismas utilizando técnicas de machine learning.

  Las grandes empresas de tecnología están desarrollando su propio software para aprovechar al máximo sus aplicaciones, una de ellas, y de las que mejores resultados da, es Google [39].

- **Reconocimiento Facial**

  El reconocimiento facial [26], [34] es una de las principales aplicaciones de la visión artificial, y consiste en **localizar y reconocer rostros de gente** en imágenes.

  Este facial recognition tiene muchas aplicaciones, cada día más utilizadas, destacando las relacionadas con la **seguridad**, como la utilizada en los aeropuertos para llevar un control de los pasajeros, o en las cámaras públicas de las ciudades, para obtener información sobre atracos, ataques terroristas, etc.

  También destacan los usos en reconocimiento y seguimiento de clientes por parte de empresas y tiendas, así como el reciente "desbloqueo facial"de los teléfonos inteligentes.

  En el siguiente enlace [23] se pueden ver distintas aplicaciones desarrolladas por tensorflow, no sólo para el reconocimiento facial, sino para el reconocimiento de imágenes en general.

  El proceso de este reconocimiento se compone de dos partes (para la información completa consultar el siguiente artículo [35]):

  - **Detección de rostros**

    En esta primera parte del proceso se **identifican los pixeles de la imágen que pertenezcan a un rostro**. Para realizar dicha identificación, hay varios algoritmo, aunque uno de los más utilizados y extendidos es el llamado "Haar Cascade face detection"[3]. Todas las versiones de este algoritmo se pueden

encontrar en el siguiente repositorio [78].

Hay muchas aplicaciones [54], [53] que se aprovechan de esta detección de rostros sin llegar a usar el reconocimiento facial, ya que no supone tanto trabajo implementar esta primera parte como seguir con el proceso.

- **Reconocimiento de rostros**

  En esta segunda parte se utilizan diversas técnicas para **evitar los problemas derivados de la calidad de la imágen** (iluminación, posición de la persona, cambios en los rasgos faciales, etc) como pueden ser las funciones que realizan cortes o "thresholds"de nuestra imágen a partir de ciertos valores umbrales, y se obtiene como resultado una segunda **imágen más .estandarizada", la cual comparamos** (usando en nuestro caso los CascadeClassifier [1]) con las otras imágenes que teníamos preparadas para ser comparadas, y como resultado final obtenemos una de estas imágenes, que será la que contenga el rostro que queríamos reconocer en la primera imágen (o la que más se acerque).

- **Eigenvectores**

  Un eigenvector o valor propio de una imágen es, según wikipedia [121], ün vector no nulo que, cuando es transformado por el operador, da lugar a un múltiplo escalar de sí mismo, con lo que no cambia su dirección."

  Esta definición a nosotros no nos interesa mucho, aunque si nos interesa su aplicación, que viene a significar lo siguiente:

  De una imágen, obtenemos todos sus píxeles, y hacemos que cada uno de ellos represente un elemento de una matriz de tamaño MxN (tamaño de la imágen). **Empezamos por un píxel** (generalmente al azar), y realizamos la operación que hemos visto en la definición más arriba con cada uno de los píxeles en los alrededores.

  De esta manera estamos **comparando su dirección con la de cada uno de los píxeles cercanos**, para saber cuales de ellos hacen que cambie su dirección.

Si todos los píxeles en los alrededores **mantienen la dirección** del pixel actual, significa que todos ellos **pertenecen al mismo elemento dentro de la imágen**.

Si hay alguno que **hace cambiar de dirección el eigenvector del pixel actual**, significa que esos dos píxeles pertenecen a **elementos diferentes de la imágen**.

De esta manera se pueden obtener las diferentes características de una imágen a partir de los eigenvectores.

Hay multitud de aplicaciones de los eigenvectores relacionados con imágenes por internet, en la siguiente referencia se muestran algunas de ellas [44].

- **Algoritmos de Reconocimiento Facial**

  Los diferentes algoritmos que se barajaban para realizar el reconocimiento facial fueron:

  - **Eigenfaces**

    Las eigencaras o eigenfaces [127] es el nombre que se les da a los eigenvectores cuando son usados en el reconocimiento facial de las personas con computer vision.

    Es la forma más básica y una de las primeras técnicas conocidas para el reconocimiento facial, que se empezó a usar a partir de 1991.

  - **Fisherfaces**

    Las fisherfaces [65] se basan en el mismo modelo que las eigenfaces, aunque en este caso se tiene también en cuenta la luz y los espacios entre características del rostro, dando más importancia a las expresiones faciales que a los propios rostros.

    A partir de 1997 se le empezó a dar un poco más de importancia a este método, aunque nunca acabó de tener mucho éxito.

  - **Local binary patterns**

    Según Kelvin Salton [95], el algoritmo LBP (Local Binary Pattern) es un sistema simple pero eficaz, que consiste en etiqutar los píxeles de una imágen a partir del umbral que se le asigna

a cada uno de sus vecinos. Esto significa que le da un valor a cada uno de los píxeles en función de la intensidad de color de los píxeles cercanos [93]. Para ello es necesario tratar la imágen en **escala de grises**.

Si la intensidad de color de uno de los píxeles es mayor o igual, a ese pixel se le asigna un '1', en caso contrario un '0'. A continuación, se juntan los 8 bits obtenidos de los 8 píxeles vecinos al pixel que estemos comparando y se forma un número en binario (el órden en que se leen los bits es irrelevante, mientras se use el mismo órden en toda la imágen, aunque lo más común suele ser empezar por una esquina y seguir el sentido de las agujas del reloj, aunque no hay ningún standard), que corresponderá con la nueva **intensidad de color** del pixel (de 0 -negro- a 255 -blanco-).

Con esta nueva intensidad de color definida para todos los píxeles de la imágen, se puede obtener una **relación entre ellos**, saber qué pixeles están más relacionados entre ellos que con otros, etc. A partir de cada una de estas agrupaciones **se obtienen las características** que vamos a utilizar en nuestro sistema de reconocimiento facial.

Se puede observar todo el proceso en la imágen 3.3.

Se empieza a estudiar antes que las fisherfaces, aunque hasta 1996 no se pone de moda. A partir de entonces se ha convertido en el más utilizado por su sencillez y relativa eficacia.

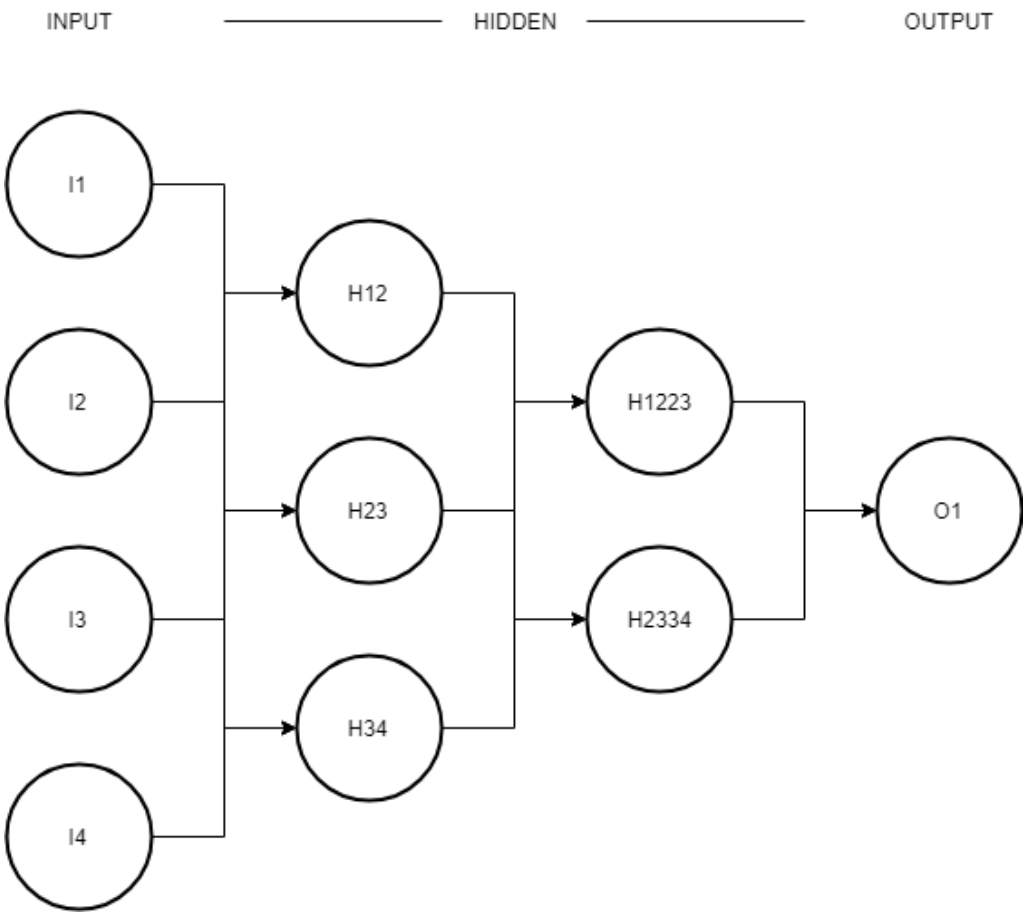

Figura 3.1: Red neuronal con las diferentes capas implicadas.

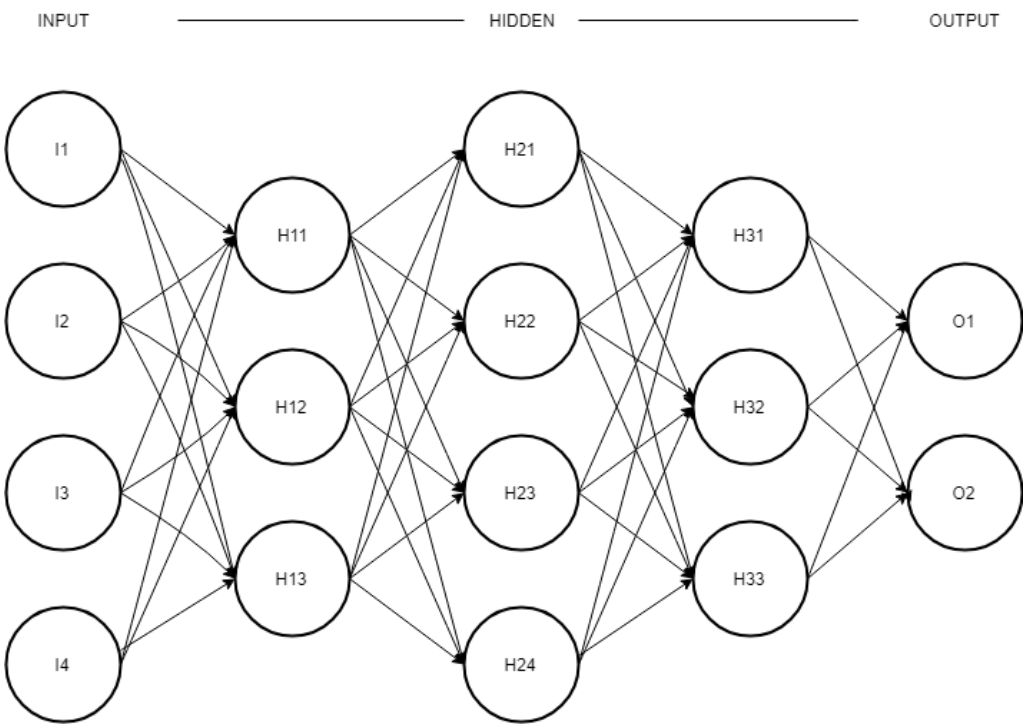

Figura 3.2: Red neuronal convolucional con las diferentes capas implicadas y cómo se relacionan los nodos de cada una entre si.

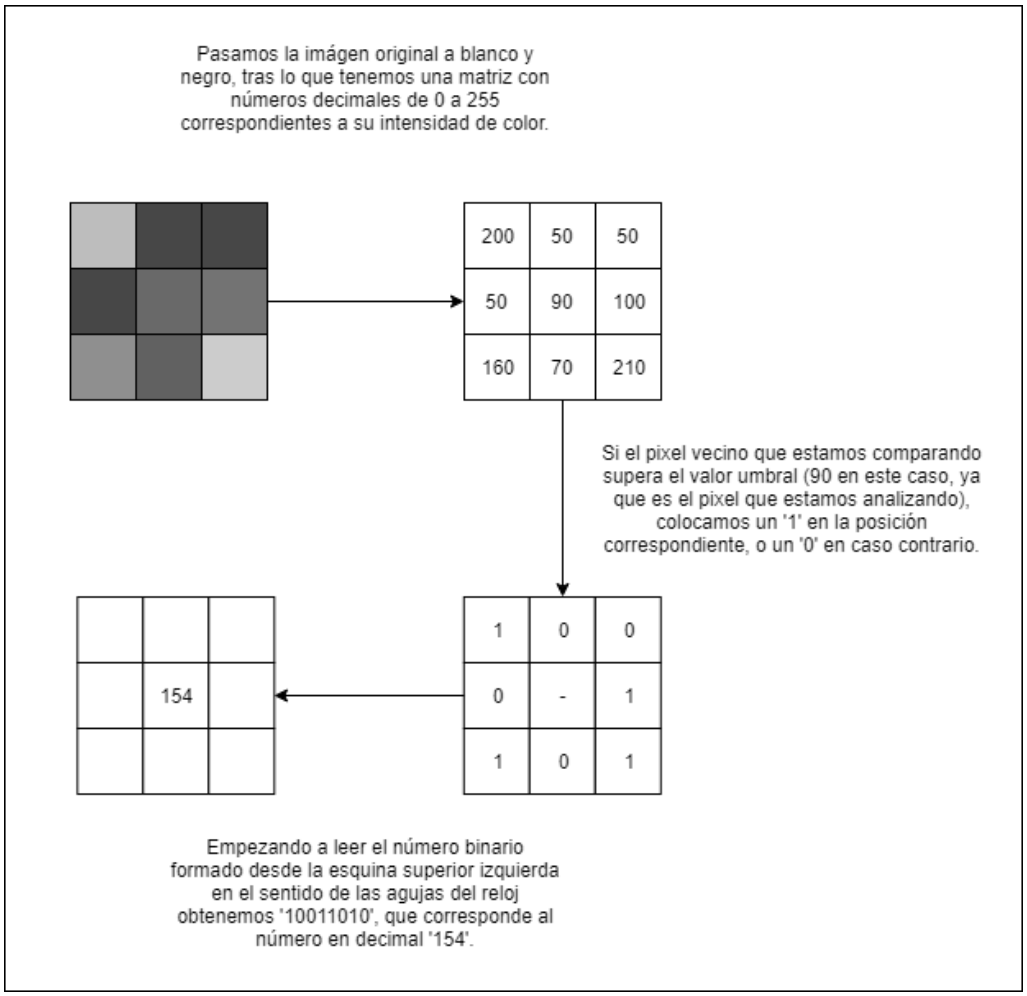

Figura 3.3: Proceso de agrupación e identificación de características en una imágen según el algoritmo LBPH.

Técnicas y herramientas

4.1. Fases de desarrollo

Las fases que se han seguido a la hora de desarrollar el proyecto han sido [100]:

- **Definir el proyecto**

  Al principio se tuvo que escoger el tipo de proyecto que se iba a realizar. En nuestro caso un proyecto medio, ya que no se tenían ni el tiempo ni los recursos necesarios para que fuera un proyecto demasiado grande, y tampoco podía ser un proyecto demasiado pequeño dado el objetivo del mismo.

  También se tuvo que definir la idea principal, que en nuestro caso iba a ser reconocer gente.

- **Identificar información, recursos y requisitos**

  A continuación, se definió qué tipo de información iba a tratar: íbamos a trabajar con imágenes en sus diferentes formatos, y con redes neuronales, las cuales íbamos a entrenar para que reconocieran a la gente almacenada en nuestra base de datos.

  Los recursos de los que íbamos a disponer eran:

  1. La imágen que obtuviéramos con la cámara.
  2. Otras imágenes de gente a la que necesitáramos reconocer (en nuestro caso gente desaparecida o en busca y captura). Dichas imágenes se obtendrían de una base de datos oficial de la policía/gobierno.

3. Información básica sobre cada una de las personas que tuviéramos almacenadas en la base de datos de nuestro programa.

En cuanto a los requisitos que necesitábamos cumplir, se establecieron los siguientes:

1. Obtener imágenes en tiempo real.
2. Entrenar una red neuronal con machine-learning o derivados (deep learning, computer vision, etc).
3. Tener una base de datos con imágenes que usaríamos para entrenar dicha red.
4. Realizar una comparación satisfactoria de la persona identificada en la imágen en tiempo real.
5. Mostrar resultados e información en una interfaz al usuario.

- **Fase de planificación**

  Cuando se empezó a desarrollar el proyecto, se sabía cual era el objetivo general del proyecto, aunque no el camino exacto que iba a seguir, de manera que se optó por seguir una serie de hitos principales, enumerados en el apartado anterior, y a partir de esos hitos ir creando otros más pequeños y asequibles a la hora del desarrollo.

  No se pretendía ir completando los hitos uno por uno y que no se pudiera empezar a desarrollar uno hasta que se acabara el anterior, de manera que el progreso se ha ido realizando sobre todos ellos de manera más o menos equivalente.

  Los hitos grandes no tenían fecha prevista de ser completados, sino que se completarían cuando no quedaran tareas pendientes relacionadas con dicho hito, de manera que era complicado planificar el desarrollo para que coincidiera con unos plazos concretos, así que se planificaron el resto de tareas más pequeñas, completando varias para cada reunión.

  Dichas reuniones se llevaban a cabo en función del trabajo planteado de una a otra, así que no eran todas las semanas, sino que se adaptaban un poco al trabajo pendiente, aunque si que es cierto que se han tenido reuniones cada (como máximo) dos semanas, para llevar un seguimiento más o menos regular del desarrollo y que no se quede

el mismo estancado en el mismo punto demasiado tiempo.

En las herramientas que se han barajado utilizar respecto a la planificación, se barajó utilizar trello [9], aunque se descartó ya que, sin otros miembros que aporten contenido, no tenía sentido tener una lista de tareas 'to do' o 'doing' para una sola persona.

También se pensó en utilizar zenhub [126] para la planificación de hitos, tareas y sprints, aunque se descartó la idea al no tener unos plazos fijos de entrega para cada una de ellas.

En la tabla 4.1 se pueden observar tanto los milestones principales del desarrollo del proyecto como las issues o tareas asociadas a cada uno.

- **Fase de investigación**

  A la vez que se realizaba la planificación de las tareas semanales, se iba realizando un proceso de investigación e información, tanto de conceptos y mecánicas como de seguimiento de tutoriales, documentación y diferentes referencias sobre los temas tratados en el proyecto (para más información consultar el apartado 2.2).

- **Desarrollo**

  En este apartado hemos ido realizando el desarrollo del propio proyecto, tanto la estructura de la fase de planificación como el propio código. Para ello se ha seguido un orden lógico de desarrollo, centrándonos primero en la funcionalidad básica de tomar una imágen de la cámara y compararla con las de la base de datos, para pasar más tarde a ampliar el número de muestras de nuestra red para mejorar el ratio de acierto de las predicciones, crear un interfaz gráfico para mostrar los resultados y que al usuario le resulte más sencilla su interpretación y crear un ejecutable para evitar tener que ejecutar nuestro código completo en un editor.

  Estas últimas modificaciones se toman para facilidad del usuario, intentando automatizar todo lo posible el proceso, aunque es cierto que se necesitan ciertos datos por parte del usuario, además de permitirle cierta libertad (por ejemplo, podríamos entrenar la red a cada ejecución, incluso si nuestras muestras no han variado y la red está correctamrnte entrenada, pero supondría una pérdida de tiempo y

Milestones	**Issues**	**Tipo de tarea**
1.- Install and Configure	Install environment	Develop
	Create class diagram	Develop
	Basic readme	Develop
2.- Image manipulation	Open image from file	Develop
	Save images	Develop
	Tensorflow Tutorials	Develop
	Meeting	Meeting
3.- Basic image recognition	Import OpenCV	Develop
	Capture image from camera	Develop
	Normalize illumination	Enhancement
	Create basic parameters	Develop
	Facial recognition (I)	Develop
	Meeting	Meeting
4.- Apply computer vision	Improve camera recording	Enhancement
	Change face location display	Enhancement
	Facial recognition (II)	Develop
	Meeting	Meeting
	Acquire camera	Enhancement
5.- Create database	Create local storage	Develop
	Get more images	Enhancement
6.- Interface	Basic interface	Develop
	Dynamic window	Enhancement
	Facial recognition (III)	Enhancement
	Camera view	Enhancement
	Result images	Develop
	Compare bar/percentage	Develop
	Meeting	Meeting
	Result information	Develop
	Executable	Develop
7.- Report	Basic report	Develop
	Meeting	Meeting
	Clean and order code	Enhancement
	Update readme	Enhancement
	Final report	Enhancement
	Video	Develop

Tabla 4.1: Distribución del desarrollo del proyecto según Milestones e Issues.

recursos en muchos casos, de manera que se opta por darle opción al usuario).

- **Revisión y control continuo**

  Además de las reuniones (semanales o bisemanales) mencionadas en la planificación, se ha ido llevando un control de errores por cuenta propia, que se iban resolviendo inmediatamente si eran urgentes, o después de realizar las tareas asignadas para esa semana si no afectaban al uso general del programa.

  Con el proyecto en github, ha sido sencillo seguir el progreso y desarrollo del mismo, sabiendo en todo momento qué tareas estábamos realizando, cuales nos quedaban por realizar, cuales eran los objetivos generales, etc.

4.2. Patrones

A la hora de desarrollar el proyecto, no se hizo pensando en utilizar unos u otros patrones de manera intencionada, sino que se fueron añadiendo en función de las necesidades.

Al principio se utilizaron más patrones [60], como los **Adaptadores ??**, que utilizaban clases intermedias para comunicar dos elementos diferentes del proyecto (la base de datos con la interfaz, por ejemplo).

O en el caso de obtener las imágenes que íbamos a analizar, en una primera versión se utilizaba el patrón **Abstract Factory** 4.5, ya que existía una interfaz 'CaptureImage' que instanciaba una clase concreta 'CaptureImageFromFile' o 'CaptureImageFromCamera' en función del orígen de la imágen.

Aunque posteriormente se unificaron varias clases, se hizo refactorización de muchos de los métodos anteriores, y se consiguió eliminar complejidad de código, referencias y clases, por lo que algunos de estos patrones ya no eran necesarios.

Al final, el patrón más utilizado y que más claramente podemos encontrar es el de **Singleton** 4.6, especialmente útil cuando queremos instanciar una nueva interfaz gráfica, que nos ayuda a asegurar que no se creen y sobreescriban varias interfaces (ya que la utilizamos de varias maneras diferentes, no nos interesa crear una interfaz para cada uno de estos usos, sino una común que vaya cambiando en función del uso que le queramos dar).

4.3. Herramientas

Algunas de las herramientas que hemos utilizado para el desarrollo del proyecto han sido los siguientes:

- El lenguaje de programación ha sido Python [30].
- Para realizar la instalación del entorno virtual utilizado durante el desarrollo del trabajo se ha utilizado Anaconda [8].
- Herramienta OpenCV [79]
- Se han obtenido respuestas y soluciones a varios de los problemas encontrados durante la realización del proyecto en stackoverflow [46].
- Para los diagramas de clases se ha usado la herramienta gratuita starUML [73].
- Para los flujogramas y otras imágenes de esquemas y diagramas se ha utilizado la herramienta draw.io [62].
- Para la memoria se ha utilizado la herramienta que permite la edición de ficheros en LaTex online sharelatex [98].
- A la hora de editar y formatear el fichero readme.md de github, se han utilizado herramientas y recursos varios [36].

- El editor de texto utilizado ha sido pycharm [48].

- Se ha utilizado la aplicación image.net [107] para la obtención de algunas imágenes de muestra.

Enlaces que resultaron útiles a la hora de desarrollar (y resolver problemas) del proyecto fueron:

- Relacionados con la instalación y configuración de python, sus complementos y similares:
  - How do I install pip on Windows? [80]
  - Installing packages with pip [29]
  - Installing modules with python [28]
  - Using wheels to install python dependences [38]
  - Library to solve compatibility problems between python 2.7 and 3.6 [85]
  - ERROR: problema a la hora de usar input()/raw_input() con las diferentes versiones de python [27]
  - Clases y paquetes en python [31]
  - ERROR: missing 1 required positional argument (Problema relacionado con el uso de clases y objetos en python [63])
  - Algunos problema con las llamadas a funciones desde el main [32] y desde otras clases [61]
  - Clases abstractas e interfaces en python [124], [58]

- Tratamiento de excepciones en python [82]
- Operadores ternarios [57]
- Operaciones con listas [90], [7]
- Contador de tiempo en python [86]
- Singleton simple en python [83]
- Operaciones con ficheros [105], [12], [25], [108], [123]
- Artículo [10] bastante interesante sobre cómo trabaja python con los redondeos y como solucionar posibles problemas que surgen de ello.

- Relacionados con el tratamiento de imágenes y Tensorflow:
  - Repositorio oficial de Tensorflow [110]
  - Página con buenos tutoriales sobre el uso general de Tensorflow [59]
  - Tutorial rápido de instalación de tensorflow [19]
  - ERROR: Tensorflow not found in pip [115]
  - Cómo entrenar un modelo de detección de rostros con Tensorflow [114]
  - Usando Tensorflow con la GPU en una red convolucional [75]
  - Pillow para pycharm [47]
  - DLib para reoconocimiento básico de rostros [56]

- Cargar, modificar y guardar imágenes (con la librería de cv) [2]
- Abrir imágenes desde fichero [16]

- Relacionados con la herramienta OpenCV utilizada para computer vision:
  - Se puede consultar las siguientes páginas para solucionar alguno de los problemas que pueden surgir de la instalación de OpenCV: [17], [45], [103], [113]
  - Activar cámara [74]
  - Detección de rostros con deep learning (Librería CV en nuestro caso [102]
  - Cómo funciona realmente el método predict() [13] de open cv.
  - Documentación [76] sobre FaceRecognition de Open CV.
  - ERROR: problema a la hora del tratamiento de typos de las imágenes [125], [24]
  - Pausar ejecución para ver resultados [11], [68]
  - ERROR: la cámara muestra imágen en negro - ajustar parámetros [116]
  - Cambiar tamaño de las imágenes obtenidas [112], [4]
  - Recortar imágenes para guardar en la base de datos sólo los rostros [33]
  - Imágen en b/n utilizando funciones de Threshold (para normalizar iluminación y elementos innecesarios) [77]

- Capturar imágenes con la cámara [52], [50], [111]

- Relacionados con la interfaz;
  - Resolución de pantalla con python (en nuestro caso con las librerías de tKinter y wx) [70]
  - Redimensionar automáticamente las ventanas de la interfaz en función de la resolución de nuestra pantalla [15]
  - Mostrar múltiples imágenes en una sola ventana [87] (con la librería de cv), [104] (con la librería de matplotlib)
  - Redimensionar imágenes de salida (con la librería de cv) [43]
  - Mostrar imágen capturada con la cámara en sentido correcto [5]
  - Sencillo tutorial para aprender a usar tKinter [14]
  - Juntando los resultados de OpenCV con la interfaz de tKinter [91]
  - Personalizando la interfaz con tKinter [72]
  - ERROR: botón para mostrar información adicional no responde [66]
  - ERROR: abrir dos ventanas conjuntas con tKinter [109], [99], [88], [42]
  - Barra de progreso (comparación) [21] en tKinter

- Otros enlaces de interés:
  - .gitignore de Python [37]

- Eliminar carpeta con contenido en github [55]
- Base de datos SQLite [106]
- Matplotlib para mostrar imágenes [49]
- Crear ejecutable [71] (Herramienta py2exe), [67] (Herramienta pyinstaller), [101] (Usar un .bat), [64] (diversas soluciones)

- Repositorios consultados:
  - Clasificación de imágenes usando Tensorflow [122]
  - Detección de rostros con la api de Tensorflow [89], [81]
  - Google cloud platform: Computer Vision [40]
  - Microsoft Cognitive Toolkit [69]
  - COCO: reconocimiento de objetos a gran escala [18]
  - Facenet: Face recognition using Tensorflow [20]
  - Face recognition: The world's simplest facial recognition api for Python and the command line [6]
  - Red neuronal desarrollada en python con numpy [41]
  - Tutoriales sobre computer vision [96]

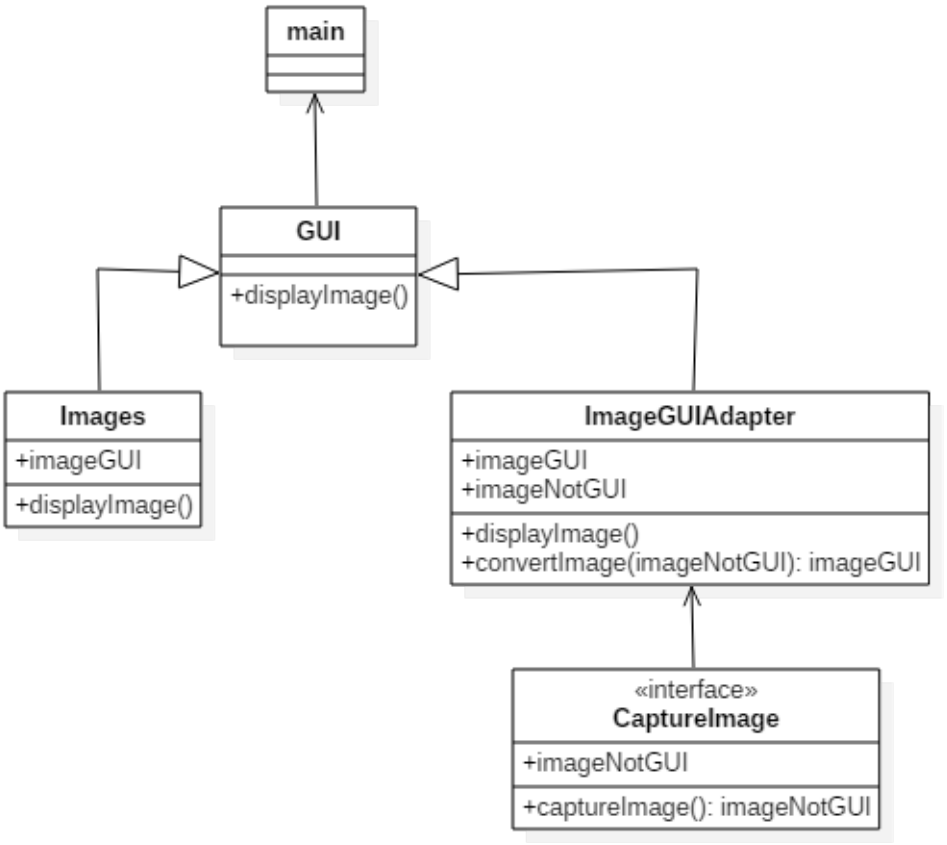

Figura 4.4: Ejemplo de patrón Adaptador utilizado al comienzo del desarrollo del proyecto.

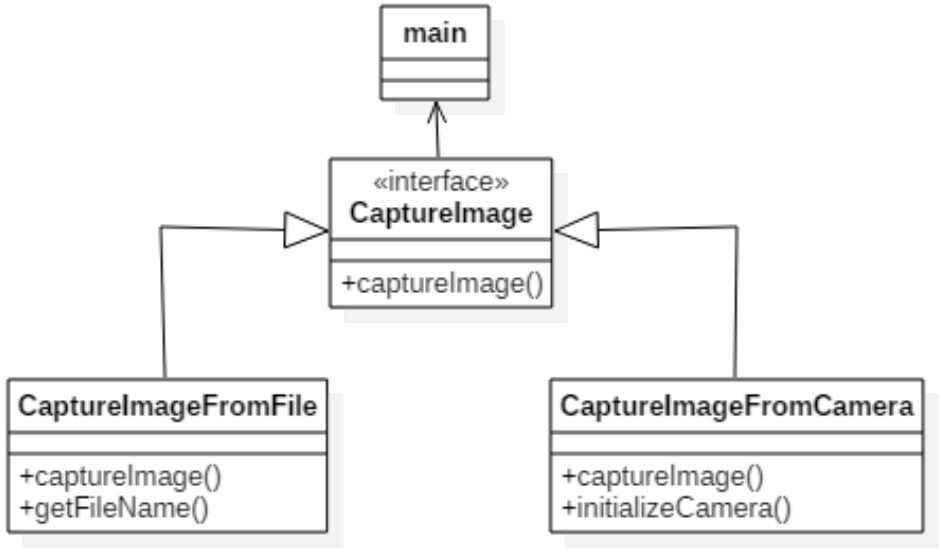

Figura 4.5: Ejemplo de patrón Abstract Factory utilizado al comienzo del desarrollo del proyecto.

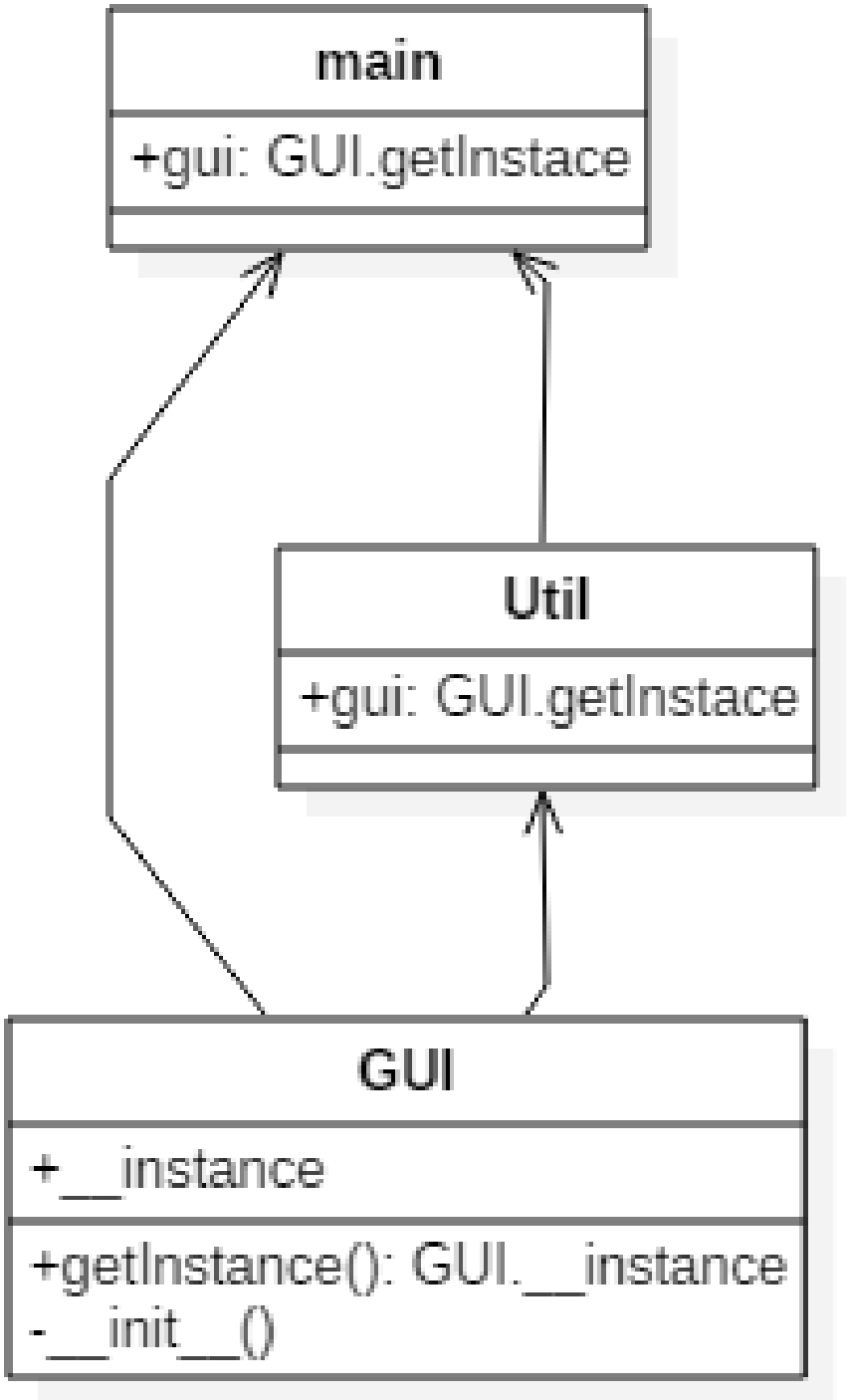

Figura 4.6: Ejemplo de patrón Singleton utilizado en la clase GUI.

Aspectos relevantes del desarrollo del proyecto

Trabajos relacionados

Sistema de Reconocimiento Facial [97]

Conclusiones y Líneas de trabajo futuras

Bibliografía

[1] OpenCV Version 2.4.13. Haar feature-based cascade classifier for object detection. https://docs.opencv.org/2.4/modules/objdetect/doc/cascade_classification.html, 2018.

[2] OpenCV Version 2.4.13. Load, modify, and save an image. https://docs.opencv.org/2.4/doc/tutorials/introduction/load_save_image/load_save_image.html, 2018.

[3] OpenCV Version 3.4.1. Face detection using haar cascades. https://docs.opencv.org/3.4/d7/d8b/tutorial_py_face_detection.html, 2018.

[4] achalddave. image.scale vs opencv resize. https://github.com/torch/image/issues/188, 2016.

[5] ADMIN. Flip image opencv python. https://scottontechnology.com/flip-image-opencv-python/, 2016.

[6] ageitgey. face_recognition. https://github.com/ageitgey/face_recognition, 2018.

[7] alvas ; Burhan Khalid. Concatenate item in list to strings. https://stackoverflow.com/questions/12453580/concatenate-item-in-list-to-strings, 2012.

[8] Inc Anaconda. Anaconda. https://anaconda.org/anaconda/python, 2018.

[9] Atlassian. Trello. https://trello.com.

[10] Danielle B. Rounding in python — when arithmetic isn't quite right. https://kfolds.com/rounding-in-python-when-arithmetic-isnt-quite-right-11a79a30390a, 2017.

[11] Antonio Serrano ; Martin Beckett. Make a pause between images display in opencv. https://stackoverflow.com/questions/46671348/make-a-pause-between-images-display-in-opencv, 2017.

[12] Python For Beginners. Reading and writing files in python. http://www.pythonforbeginners.com/files/reading-and-writing-files-in-python, 2013.

[13] MS23 ; berak. What is the label that 'predict' sends? http://answers.opencv.org/question/138933/what-is-the-label-that-predict-sends, 2017.

[14] Bodenseo Bernd Klein. Python tkinter course. https://www.python-course.eu/tkinter_labels.php, 2018.

[15] Tomha ; Marcin ; bodger and scraper. Tkinter resize background image to window size (python 3.4). https://stackoverflow.com/questions/24061099/tkinter-resize-background-image-to-window-size-python-3-4, 2014 - 2016.

[16] Casper Olsson ; BrtH. Open images? python. https://stackoverflow.com/questions/16387069/open-images-python, 2013.

[17] Andrei Cheremskoy. Opencv installation on linux and windows. https://gettocode.com/2017/02/07/opencv-installation-on-windows-and-maybe-linux, 2017.

[18] cocodataset. Coco: Common objects in context. http://cocodataset.org/#home, 2018.

[19] Melissa Dale. Python: Tensorflow on windows 10 in pycharm. https://www.youtube.com/watch?v=83vR1Nz3dHA, 2017.

[20] davidsandberg. facenet. https://github.com/davidsandberg/facenet, 2018.

[21] david_ p ; Noelkd ; BANZ111. Ttk.progressbar: How to change thickness of a horizontal bar. https://stackoverflow.com/questions/17912624/ttk-progressbar-how-to-change-thickness-of-a-horizontal-bar, 2013.

[22] Víctor de Castro Hurtado. Repositorio TFG - facial recognition. https://github.com/victorcas04/TFG-FacialRecognition.

[23] Google Developers. Image recognition. https://www.tensorflow.org/tutorials/image_recognition, 2018.

[24] M456 ; dF. Saving a numpy array as an image. https://stackoverflow.com/questions/902761/saving-a-numpy-array-as-an-image, 2009.

[25] duhhunjonn ; pycruft. How do i list all files of a directory? https://stackoverflow.com/questions/3207219/how-do-i-list-all-files-of-a-directory, 2013.

[26] Jan Fajfr. The basics of face recognition. https://blog.octo.com/en/basics-face-recognition, 2011.

[27] Jogofus ; FJSevilla. Duda con raw_input(). https://es.stackoverflow.com/questions/38288/duda-con-raw-input, 2017.

[28] Python Software Foundation. Building and installing python modules. https://docs.python.org/3/library/distutils.html#module-distutils, 2018.

[29] Python Software Foundation. Installing packages. https://packaging.python.org/tutorials/installing-packages/#requirements-for-installing-packages, 2018.

[30] Python Software Foundation. Python - version 2.7. https://www.python.org/downloads/release/python-2715, 2018.

[31] Python Software Foundation. Python packages. https://docs.python.org/2/tutorial/modules.html#packages, 2018.

[32] franka ; Amber. Python main call within class. https://stackoverflow.com/questions/7870869/python-main-call-within-class, 2011.

[33] Nolik ; Froyo. How to crop an image in opencv using python. https://stackoverflow.com/questions/15589517/how-to-crop-an-image-in-opencv-using-python, 2013.

[34] Adam Geitgey. Machine learning is fun! part 4: Modern face recognition with deep learning. https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learni 2016.

[35] Adam Geitgey. Machine learning is fun! part 4: Modern face recognition with deep learning. https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learni 2016.

[36] github. Basic writing and formatting syntax. https://help.github.com/articles/basic-writing-and-formatting-syntax/#styling-text, 2018.

[37] github. Python.gitignore. https://github.com/github/gitignore/blob/master/Python.gitignore, 2018.

[38] Christoph Gohlke. Unofficial windows binaries for python extension packages. https://www.lfd.uci.edu/~gohlke/pythonlibs/#pygame.

[39] Google. Google cloud vision api documentation. https://cloud.google.com/vision/docs.

[40] GoogleCloudPlatform. cloud-vision. https://github.com/GoogleCloudPlatform/cloud-vision, 2018.

[41] greydanus. pythonic_ocr. https://github.com/greydanus/pythonic_ocr, 2017.

[42] Tom ; guillegiraldo. How to disable window controls when a modal dialog box is active in tkinter? https://stackoverflow.com/questions/31892015/how-to-disable-window-controls-when-a-modal-dialog-box-is-active-in-tk 2015.

[43] Dr Dre ; Guyygarty. Resizing the output window of imshow function. http://answers.opencv.org/question/84985/resizing-the-output-window-of-imshow-function/, 2016.

[44] Bharath Hariharan. How are eigenvectors and eigenvalues used in image processing? https://www.quora.com/How-are-Eigenvectors-and-Eigenvalues-used-in-image-processing, 2013.

[45] Michael Hirsch. Install opencv 3.4 in python 3.6 / 2.7. https://www.scivision.co/install-opencv-python-windows/, 2018.

[46] Stack Exchange Inc. Stackoverflow. https://stackoverflow.com, 2018.

[47] LAUR IVAN. Install pillow from pycharm. https://www.laurivan.com/install-pillow-from-pycharm, 2010.

[48] JetBrains. Pycharm. https://www.jetbrains.com/pycharm, 2018.

[49] Eric Firing Michael Droettboom John Hunter, Darren Dale and the Matplotlib development team. Matplotlib - version 2.2.2. https://matplotlib.org, 2012 - 2018.

[50] Cerin ; Multimedia Mike ; jsbueno. How to programmatically capture a webcam photo. https://stackoverflow.com/questions/9711946/how-to-programmatically-capture-a-webcam-photo, 2012.

[51] Andrej Karpathy Justin Johnson. Convolutional neural networks for visual recognition. http://cs231n.github.io/convolutional-networks, 2018.

[52] Alexander Mordvintsev ; Abid K. Getting started with videos - capture video from camera. http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_gui/py_video_display/py_video_display.html, 2013.

[53] kaboomfox. Overlay a smaller image on a larger image python opencv. https://stackoverflow.com/questions/14063070/overlay-a-smaller-image-on-a-larger-image-python-opencv, 2012.

[54] KP Kaiser. Deal with it in python with face detection. https://dev.to/burningion/deal-with-it-in-python-with-face-detection-chi, 2017.

[55] Sahat Yalkabov ; karmakaze. How to remove a directory from git repository? https://stackoverflow.com/questions/6313126/how-to-remove-a-directory-from-git-repository, 2011.

[56] Davis King. Dlib 18.7 released: Make your own object detector in python! http://blog.dlib.net/2014/04/dlib-187-released-make-your-own-object.html, 2014.

[57] kip. Python - ejemplo del operador ternario en python. https://www.lawebdelprogramador.com/foros/Python/1517833-Ejemplo-del-operador-ternario-en-Python.html, 2016.

[58] Bernd Klein. Abstract classes. https://www.python-course.eu/python3_abstract_classes.php, 2018.

[59] LearningTensorFlow.com. Beginner-level tutorials for a powerful framework. https://learningtensorflow.com, 2016.

[60] Antonio Leiva. Patrones de diseño de software. https://devexperto.com/patrones-de-diseno-software/, 2016.

[61] levacjeep ; ShadowRanger. Python3 - typeerror: module.__init__() takes at most 2 arguments (3 given). https://stackoverflow.com/questions/35367340/python3-typeerror-module-init-takes-at-most-2-arguments-3-given/35367871, 2016.

[62] JGraph Ltd. Draw-io. https://www.draw.io, 2018.

[63] mahasamoot. trouble with the tutorial: Typeerror: step() missing 1 required positional argument: 'model'. https://github.com/projectmesa/mesa/issues/308, 2016.

[64] mamcx ; Caleb Hattingh ; Jason Baker ; Eli Bendersky. How to deploy python to windows users? https://stackoverflow.com/questions/1646326/how-to-deploy-python-to-windows-users, 2009.

[65] A. Martinez. Fisherfaces. *Scholarpedia*, 6(2):4282, 2011. revision #91266.

[66] Rikg09 ; Noshii ; mayure098. Tkinter command for button not working. https://stackoverflow.com/questions/45710162/tkinter-command-for-button-not-working, 2017.

[67] David Cortesi ; Giovanni Bajo ; William Caban ; Gordon McMillan. Pyinstaller manual - version 3.3.1. http://pyinstaller.readthedocs.io/en/v3.3.1/.

[68] md1hunox ; Abid Rahman K. Using other keys for the waitkey() function of opencv. https://stackoverflow.com/questions/14494101/using-other-keys-for-the-waitkey-function-of-opencv, 2017.

[69] Microsoft. Microsoft cognitive toolkit - cntk. https://github.com/Microsoft/CNTK, 2018.

[70] Mike. Getting your screen resolution with python. https://www.blog.pythonlibrary.org/2015/08/18/getting-your-screen-resolution-with-python/, 2015.

[71] MikeFox. python: py2exe tutorial. http://www.py2exe.org/index.cgi/Tutorial, 2014.

[72] Abkb ; Mark Mikofski. How to change font and size of buttons and frame in tkinter using python? https://stackoverflow.com/questions/20588417/how-to-change-font-and-size-of-buttons-and-frame-in-tkinter-using-python, 2013.

[73] Ltd MKLab Co. Staruml. http://staruml.io, 2018.

[74] Rodrigo ; John Montgomery. How do i access my webcam in python? https://stackoverflow.com/questions/604749/how-do-i-access-my-webcam-in-python, 2009.

[75] Cole Murray. Deep learning cnn's in tensorflow with gpus. https://hackernoon.com/deep-learning-cnns-in-tensorflow-with-gpus-cba6efe0acc2, 2017.

[76] OpeCV. cv::face::facerecognizer class reference. https://docs.opencv.org/trunk/dd/d65/classcv_1_1face_1_1FaceRecognizer.html, 2018.

[77] OpenCV. Image thresholding. https://docs.opencv.org/3.4.0/d7/d4d/tutorial_py_thresholding.html, 2017.

[78] OpenCV. haarcascades. https://github.com/opencv/opencv/blob/master/data/haarcascades, 2018.

[79] OpenCV. Opencv - releases. https://opencv.org/releases.html, 2018.

[80] Colonel Panic. How do i install pip on windows? https://stackoverflow.com/questions/4750806/how-do-i-install-pip-on-windows, 2017.

[81] Rohit Patil. How to use tensorflow object detection api on windows. https://medium.com/@rohitrpatil/how-to-use-tensorflow-object-detection-api-on-windows-102ec8097699, 2018.

[82] Joey Payne. Catching python exceptions – the try/except/else keywords. https://www.pythoncentral.io/catching-python-exceptions-the-try-except-else-keywords, 2013.

[83] pazdera. Singleton example in python. https://gist.github.com/pazdera/1098129, 2011.

[84] Christian S. Perone. Deep learning – convolutional neural networks and feature extraction with python. http://blog.christianperone.com/2015/08/convolutional-neural-networks-and-feature-extraction-with-python, 2015.

[85] Benjamin Peterson. Python 2 and 3 compatibility utilities. https://pypi.org/project/six/#description, 2018.

[86] Evan Fosmark ; pobk. How can i make a time delay in python? https://stackoverflow.com/questions/510348/how-can-i-make-a-time-delay-in-python, 2012.

[87] poljakov13 ; eshirima. How to display multiple images in one window? http://answers.opencv.org/question/175912/how-to-display-multiple-images-in-one-window/, 2017.

[88] Pythonspot. Tkinter message box. https://pythonspot.com/tk-message-box, 2017.

[89] qdraw. tensorflow-face-object-detector-tutorial. https://github.com/qdraw/tensorflow-face-object-detector-tutorial, 2017.

[90] Omid CompSCI ; Afloroaie Robert. How to get everything from the list except the first element using list slicing [duplicate]. https://stackoverflow.com/questions/40443331/how-to-get-everything-from-the-list-except-the-first-element-using-lis
2016.

[91] Adrian Rosebrock. Opencv with tkinter. https://www.pyimagesearch.com/2016/05/23/opencv-with-tkinter, 2016.

[92] Margaret Rouse. Inteligencia artificial, o ai. https://searchdatacenter.techtarget.com/es/definicion/Inteligencia-artificial-o-AI, 2017.

[93] Cesar Troya S. Lbp y ulbp – local binary patterns y uniform local binary patterns. https://cesartroyasherdek.wordpress.com/2016/02/26/deteccion-de-objetos-vi/, 2016.

[94] ANKIT SACHAN. Convolutional neural network (cnn). http://cv-tricks.com/tensorflow-tutorial/training-convolutional-neural-network-for-image-classificatio, 2017.

[95] Kelvin Salton. Face recognition: Understanding lbph algorithm. https://towardsdatascience.com/face-recognition-how-lbph-works-90ec258c3d6b, 2017.

[96] sankit1. cv-tricks.com. https://github.com/sankit1/cv-tricks.com, 2018.

[97] Germán Matías Scarel. Sistema de reconocimiento facial. http://pdi-fich.wdfiles.com/local--files/investigacion/PF_Scarel_SistemaReconocimientoFacial.pdf, 2010.

[98] ShareLaTeX. Sharelatex. https://es.sharelatex.com, 2018.

[99] Silmarillion101. Tkinter create image function error (pyimage1 does not exist). https://stackoverflow.com/questions/23224574/tkinter-create-image-function-error-pyimage1-does-not-exist, 2014.

[100] sinnaps. MetodologÍa de un proyecto. https://www.sinnaps.com/blog-gestion-proyectos/metodologia-de-un-proyecto, 2018.

[101] FrustratedWithFormsDesigner ; Alvin SIU. Bat file to open cmd in current directory. https://stackoverflow.com/questions/4451668/bat-file-to-open-cmd-in-current-directory/23633328, 2010.

[102] Peter Skvarenina. Detecting facial features using deep learning. https://towardsdatascience.com/detecting-facial-features-using-deep-learning-2e23c8660a7a, 2017.

[103] Sol. Install opencv 3 with python 3 on windows. https://solarianprogrammer.com/2016/09/17/install-opencv-3-with-python-3-on-windows/, 2016.

[104] soply. disp_multiple_images.py. https://gist.github.com/soply/f3eec2e79c165e39c9d540e916142ae1, 2018.

[105] spence91 ; rslite. How to check whether a file exists? https://stackoverflow.com/questions/82831/how-to-check-whether-a-file-exists, 2008.

[106] sqlitebrowser. Db browser for sqlite. http://sqlitebrowser.org/, 2017.

[107] Princeton University Stanford Vision Lab, Stanford University. Imagenet. http://www.image-net.org/index, 2016.

[108] UnkwnTech ; Nick Stinemates. How to delete the contents of a folder in python? https://stackoverflow.com/questions/185936/how-to-delete-the-contents-of-a-folder-in-python, 2008.

[109] Tom Fuller ; PM 2Ring ; Steven Summers. I can't seem to open two windows with python, tkinter. https://stackoverflow.com/questions/39039123/i-cant-seem-to-open-two-windows-with-python-tkinter, 2016.

[110] tensorflow. tensorflow. https://github.com/tensorflow/tensorflow, 2018.

[111] Matthew G ; thebjorn ; Froyo. Capturing a single image from my webcam in java or python. https://stackoverflow.com/questions/11094481/capturing-a-single-image-from-my-webcam-in-java-or-python, 2012.

[112] Tanmay Bhatnagar ; thewaywewere. Resize an image without distortion opencv. https://github.com/torch/image/issues/188, 2017.

[113] user2971844 ; Breeze. Opencv - cannot find module cv2. https://stackoverflow.com/questions/19876079/opencv-cannot-find-module-cv2, 2013 - 2017.

[114] Dion van Velde. How to train a tensorflow face object detection model. https://towardsdatascience.com/how-to-train-a-tensorflow-face-object-detection-model-3599dcd0c26f, 2017.

[115] Johshi ; Yash Kumar Verma. tensorflow not found in pip. https://stackoverflow.com/questions/38896424/tensorflow-not-found-in-pip, 2016.

[116] victor1234 ; elzbth. How to set camera fps in opencv? cv_cap_prop_fps is a fake. https://stackoverflow.com/questions/7039575/how-to-set-camera-fps-in-opencv-cv-cap-prop-fps-is-a-fake, 2011-2014.

[117] Wikipedia. Computer vision. https://en.wikipedia.org/wiki/Computer_vision, 2017.

[118] Wikipedia. Multilayer perceptron. https://en.wikipedia.org/wiki/Multilayer_perceptron, 2018.

[119] Wikipedia. Red neuronal artificial. https://es.wikipedia.org/wiki/Red_neuronal_artificial, 2018.

[120] Wikipedia. Redes neuronales convolucionales. https://en.wikipedia.org/wiki/Redes_neuronales_convolucionales, 2018.

[121] Wikipedia. Vector propio y valor propio. https://es.wikipedia.org/wiki/Vector_propio_y_valor_propio, 2018.

[122] wolfib. Python.gitignore. https://github.com/wolfib/image-classification-CIFAR10-tf, 2017.

[123] Yasoob. The open function explained. https://pythontips.com/2014/01/15/the-open-function-explained/#more-416, 2014.

[124] Zaiste. Abstract classes in python. https://zaiste.net/abstract_classes_in_python/, 2010.

[125] Martin Beckett ; zarthur. Convert numpy array to cvmat cv2. https://stackoverflow.com/questions/9913392/convert-numpy-array-to-cvmat-cv2, 2012.

[126] ZenHub. Zenhub. https://www.zenhub.com.

[127] S. Zhang and M. Turk. Eigenfaces. *Scholarpedia*, 3(9):4244, 2008. revision #128015.