Reconeixement de cares

Jordi Martín

Resum— Aquest projecte presenta una app per identificar a una persona a partir d'una imatge o vídeo com a paràmetre i en quin moment s'ha identificat. Aquesta seria molt útil per qüestions de seguretat com per exemple: identificar a una persona amb una càmera de seguretat dins d'un edifici on es necessita autorització per ser-hi i identificar-la per saber si aquella persona és del personal autoritzat de l'edifici o no; o bé, es podria utilitzar en el control d'assistència d'una sessió de classe fent servir una càmera. Per fer tot això, s'ha decidit fer servir el software de Visual Studio Code per programar i actualitzar la carpeta de treball que es penja a Github; dins de Visual Studio Code fer servir els llenguatges de programació Python pel back-end i HTML5, CSS i Javascript pel front-end. S'aplica una 'Haar-Cascade' per trobar el contorn de la cara, Opencv per processar les imatges, Flask per llançar l'aplicació al núvol, Mongodb per crear el dataset i guardar les dades dels usuaris i llibreries d'intel·ligència artificial per entrenar un model i per predir la persona de la imatge. S'ha creat un model de Matching Learning per classificar les cares amb una interfície còmode per l'usuari.

Paraules clau—Face recognition, Flask, Haar-Cascade, Opencv.

Abstract— This project features an app to identify a person based on an image or video as a parameter and when they are identified. This would be very useful for security reasons such as: identifying a person with a security camera inside a building where permission is needed to be there and identifying it to know if that person is authorized personnel of the building or not; or it could be used in attendance control of a class session using a camera. To do all this, we decided to use the Visual Studio Code software to program and update the workbook that is uploaded to Github. Within Visual Studio Code use the Python programming languages for the back-end and HTML5, CSS and Javascript for the front-end. A 'Haar-Cascade' is applied to find the outline of the face, Opencv to process the images, Flask to launch the application to the cloud, mongodb to create the dataset and save user data and intelligence libraries artificial to train a model and to predict the person in the image. A Matching Learning model has been created to classify faces with a user-friendly interface.

Index Terms—Face recognition, Flask, Haar-Cascade, Opencv.

1 Introducció - Context del treball

Actualment, dos conceptes bastant recents i molt demandats són el concepte de *Big Data* i d'intel·ligència artificial (IA). Aplicant aquests conceptes més els que s'han après durant el període d'aprenentatge del grau d'Enginyeria de Dades, va sorgir la idea de llançar una aplicació per poder reconèixer la cara d'una persona.

Un concepte que ara mateix amb certs telèfons està implementat és el reconeixement del contorn de la cara però que podria tenir més potencial si no només fos per càmeres de mòbils i que, a part de trobar la cara, et digues qui és la persona i en quin moment l'ha reconegut. És molt interesant el coneixement de les diferents eines que hi ha darrera de tot aquest processament i les diferents tasques i problemàtiques reals de més importància que es poden resoldre amb aquestes eines.

La primera cosa que es pensa sobre reconèixer cares és que és una cosa que no és innovadora ja que ho fan els mòbils. Però, no s'aprofita tot el seu potencial com per exemple: per seguretat, una càmera en un edifici en la qual només certes persones estan autoritzades per accedir, amb el reconeixement de cares es podria identificar quines persones estan autoritzades i quina és la que està accedint. També es pot aplicar a la universitat, hi ha moltes assignatures que et compten nota per l'assistència però el professorat no es posarà a passar llista amb una classe de cinquanta persones; llavors, si es posa una càmera davant de la classe on es fa l'assignatura, cada alumne hauria de creuar la porta i amb això ja es reconeixeria la persona que està entrant i a quina hora, per saber també si ha sigut puntual o no. Una altra bona aplicació seria en qualsevol lloc on s'ha de fer un registre d'una persona de manera ràpida i eficaç, sense demanar les dades a l'usuari ja que es rebran a partir d'aquella imatge.

El reconeixement de cares és una aplicació bastant demandada i amb molt potencial, i també, és perfecte per demostrar tot el que s'ha après durant tot el grau d'Enginyeria de Dades.

- *E-mail de contacte: jordi.martinro@autonoma.cat*
- *Treball tutoritzat per: Marta Prim Sabrià (Dep. Microelectrònica i Sistemes electrònics)*
- *Curs 2021/22*

Aquest article s'estructura primer amb aquesta breu introducció, seguidament de l'apartat d'Objectius marcats, després la motivació que hi d'aquest projecte, seguit de l'explicació l'estat de l'art. A continuació, es defineixen les tasques imposades pel treball i la seva metodologia, un cop fet això es parlarà del desenvolupament de cada tasca, després es mostraran els resultats adquirits, acabant amb les conclusions extretes d'aquest projecte.

2 OBJECTIUS

S'han marcat dos objectius:

1. L'objectiu principal d'aquest TFG és que el model de 'Matchine Learning' d'intel·ligència artificial que es faci servir tingui una *accuracy* més gran o igual d'un 50% com per reconèixer la cara en una imatge.

2. Adquirir i millorar els coneixements de desenvolupament de conceptes de programació tant de Back-end com de Front-end.

3 MOTIVACIÓ

He decidit fer aquest tema "Reconeixement de cares" com a TFG, ja que reconèixer cares és un tema que sempre m'ha cridat l'atenció per les pel·lícules de ciència ficció i era un concepte que sempre he volgut saber-ne més. Ara tinc l'oportunitat de fer-ho pel què he après a la carrera sobre Intel·ligència Artificial.

Un altre motiu per qual em motiva és que tinc l'oportunitat de crear una aplicació més de cara al públic. Fer les pràctiques externes em va inspirar a fer-ho, ja que desenvolupaven una xarxa social per conèixer a gent dins del transport públic.

4 ESTAT DE L'ART

En aquest apartat es fa menció a les tecnologies que han servit de referència per avançar amb el reconeixement de cares dins d'aquest treball.

4.1 Detecció de cares

La detecció facial és una de les tasques plantejades a resoldre amb intel·ligència artificial per fer autoaprenentatge i optimitzar lo màxim possible. Avui en dia la millor tecnologia és una xarxa neuronal com «*Tensorflow*» ja que són més precises que un mètode *Matchine Learning* com per exemple «*OpenCV*», tot i que gasten més recursos de còmput i es necessita GPUs.

4.2 Reconeixement de cares

El reconeixement facial és un altre de les tasques plantejades i va a la par amb la detecció facial ja que no pots reconèixer una cara sense primer trobar-la. Avui en dia, la millor tecnologia també és una xarxa neuronal com, en aquest cas, una *Convolutional Neural Network* (CNN) com *DeepFace* pel mateix motiu que el detector.

A dia d'avui, hi ha dos metodologies per resoldre aquests tipus de problemes: *Matchine Learning* i *Deep Learning*, dos tipus que cadascun d'ells tenen les seves avantatges i desavantatges, que depenen de la tasca funcionarà un millor que l'altre, però, sobretot és que continuen apareixent eines del tipus de metodologies per resoldre qualsevol cosa. Per aquest projecte, he fet servir la metodologia de *Matchine Learning*, concretament, el reconeixedor de cares de Fisher de «*OpenCV*» per limitacions de *Hardware*.

5 TASQUES MARCADES PEL TFG

Per al TFG, s'ha decidit aplicar metodologia *SCRUM for one person* per aquest projecte, ja que és una metodologia amb la qual s'ha treballat amb ella en més d'una ocasió i permet fer un seguiment de les tasques a fer i objectius a complir. Les tasques a fer són:

- **Reconèixer els contorns d'una cara:** Implementar un script de Python amb la finalitat de trobar el contorn d'una cara en una imatge o vídeo passat com a paràmetre o la mateixa càmera. Es farà servir la llibreria «*Opencv*» pel processament i el directori de «*Haar-Cascade*» per trobar el contorn (3 setmanes).

- **Interfície:** Implementar scripts d'*HTML*, de *CSS* i de Javascript amb la finalitat de crear una interfície gràfica còmode per l'usuari (2 mesos).

- **Flask:** Implementar un script de Python amb la finalitat de llançar l'aplicació al núvol. Es farà servir la llibreria «Flask» per llançar-la i crear les diferents rutes de l'app (1 setmana).

- **Dataset:** Implementar un script de Python amb la finalitat de manipular un dataset per als usuaris, ja sigui per consultar com per afegir de nous. Es farà servir la llibreria «*Pymongo*» per fer els diferents algorismes (3 setmanes).

- **Model d'Intel·ligència artificial:** Implementar un script amb la finalitat d'entrenar un model d'intel·ligència artificial per entrenar-la amb les imatges de la cara i identificar a la persona de la cara. Es farà servir un reconeixedor de la llibreria de «*OpenCV*» amb un arxiu tipus *pickle* que són els *labels* de cada persona a reconèixer i un arxiu tipus *yml* on es guardarà tota la informació de l'entrenament del model de *Matchine Learning* per després fer-lo servir per reconèixer un usuari (3 setmanes).

- **Pujar imatges al dataset:** Implementar un *script* d'*HTML* i Python per pujar imatges i guardar-les en el dataset. Es farà servir la llibreria «*Gridfs*» per pujar las imatges al dataset de MongoDB i tornar-les a recuperar per fer-les servir. També es farà servir la

llibreria «*Base64*» i «*ByteIO*» per codificar les imatges per poder guardar-les i descodificar per poder fer-les servir (2 setmanes).

- **Visualització d'imatges en l'aplicació:** Implementar un *script* d'*HTML* i Python per poder visualitzar tant les diferents imatges que s'han pujat en el dataset com la imatge resultant de fer una predicció, amb un requadre emmarcant la cara de la persona i un text a sobre amb el seu nom. Es farà servir la llibreria «*response*» per poder visualitzar la imatge desitjada dins de l'aplicació (2 setmanes).

- **Enviar email de verificació d'usuari:** Implementar un script d'*HTML* i de Python per, un cop registrat un usuari, enviar un email amb un codi de quatre dígits per comprovar si les dades són correctes. Es farà servir la llibreria «*random*» per generar quatre dígits de manera aleatòria i guardar-les en una variable, per després comprovar si el codi posat per l'usuari coincideix amb el codi demanat. Una altra llibreria que es farà servir és la de «*smtplib*» que servirà per definir el servidor d'email que es farà servir i les diferents dades que demanen per enviar un email. També es farà servir la llibreria «*email*» per definir el text de l'email.

6 DESENVOLUPAMENT

En aquest apartat s'explica el desenvolupament de cada tasca marcada durant les sessions del TFG.

6.1 Reconèixer els contorns d'una cara

L'objectiu era crear un *script* per trobar el contorn de les cares i guardar-ho. S'ha implementat dins dels *scripts* d'entrenament del model d'IA i dins de l'*script* de predicció ja que tots dos scripts necessiten trobar una cara i dibuixar el seu contorn, un cop fet això, processar només aquella part ja que és la part desitjada. Criden el directori de Haar-Cascade per trobar el contorn d'una cara i es fa servir la llibreria «*Opencv*» per llegir els paràmetres i processar les imatges, es passa a una escala de grisos, es detecten els contorns i es guarden en el dataset. S'ha complert satisfactòriament en el temps imposat (Fig. 1). També es pot veure part de l'arxiu *xml* d'una cascada (Fig. 2).

Fig. 1: Resultat d'una imatge reconeguda d'Elon Musk.

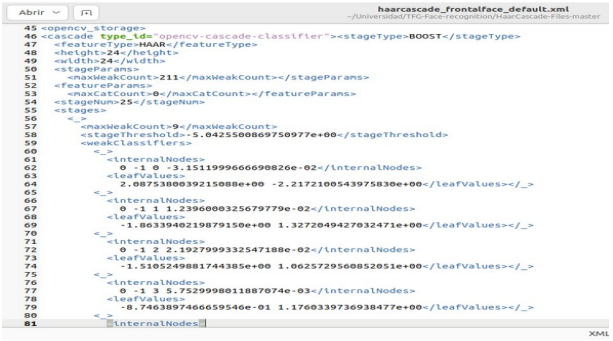


Fig. 2: Part de l'script de «Haar Cascade Frontalface Default».

6.2 Dataset

L'objectiu era crear un script per manipular el dataset dels usuaris. S'ha creat un dataset en Mongodb amb una «col·lecció» dels usuaris que es diu *users* i un altre per guardar arxius que es diu *fs*. Dins de la col·lecció d'usuaris, té com a camps l'identificador que es crearà de manera aleatòria, un string que representa el nom de l'usuari, un altre string que representa l'email de l'usuari, un altre string que representa la contrasenya de l'usuari hasheat per respectar la privacitat de les dades de l'usuari, un altre string que representa el rol de l'usuari per tenir diferents permisos d'accés a certes parts de l'app, un array amb les ids de les imatges que els usuaris pengin, un altre array amb els labels de cada imatge que posteriorment servirà per entrenar el reconeixedor i un boolea que representa si l'usuari s'ha verificat o no. També s'han afegit els mètodes de consulta del dataset que serien els de buscar un usuari per saber qui és l'usuari connectat actualment, buscar un arxiu per recuperar-ho i fer-ho servir, inserir un usuari a l'hora de registrar-se, inserir un arxiu a l'hora de pujar imatges i entrenar el reconeixedor i actualitzar un usuari per canviar el camp de verificació i actualitzar les dues arrays d'imatges. Aquest objectiu no s'ha complert, s'ha creat el dataset i es té una idea de com fer les diferents operacions. En la figura tres (Fig. 3) podem veure un exemple de document d'usuari i en la quatre (Fig. 4) un exemple d'un document arxiu.

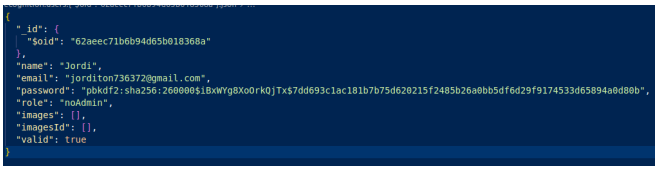


Fig. 3: Exemple d'usuari creat.

Fig. 4: Exemple d'imatge penjada al dataset.

6.3 Model d'Intel·ligència artificial

L'objectiu era buscar un model d'intel·ligència artificial per passar les cares per entrenar i poder predir la persona. La idea principal és separar les imatges de training i test per comprovar l'eficiència del model. S'ha trobat diferents models de reconeixedors de cares de «Matchine Learning» amb la llibreria d'«*OpenCV*». Consisteix en passar las imatges d'entrenament amb carpetes amb els noms de les persones, s'entrena guardant un arxiu *pickle* amb un *hash* amb els identificadors de cada persona i un arxiu *yml* amb una llista X_train que conté les imatges en blanc i negre retallades només per la part de la cara, i una llista Y_labels que conté un valor numèric que representa un usuari, depèn del valor numèric que conté el diccionari de l'arxiu *pickle*. Totes dues llistes es passen com a paràmetre. Un cop fet això, se li passa l'arxiu *yml* al reconeixedor i amb l'arxiu *pickle* s'identifica a la persona. S'ha tingut en compte l'*accuracy* de la Haar-Cascade a l'hora de reconèixer una cara i del mateix reconeixedor. Es pot veure part de l'arxiu *yml* a continuació (Fig. 5).

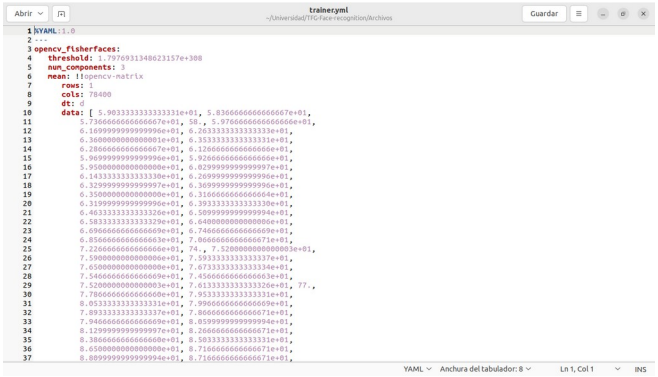


Fig. 5: Part de l'script de «Trainer.yml».

6.4 Directori temporal

L'objectiu era solucionar el problema que hi havia per processar els diferents tipus d'arxius un cop recuperats de la col·lecció *fs* o demanats per l'usuari. La solució trobada ha sigut fer servir un directori temporal. Un directori que es crea al començament de l'execució de l'aplicació, es pugen els arxius que es faran servir per processar-los i un cop es deixa d'executar l'aplicació s'esborra el directori. Aquest problema va sorgir per la part de fer servir les imatges per entrenar el model de *Matchine Learning* per fer servir tant els arxius *pickle* i *yml* com la imatge a predir amb el model de *Matchine Learning*, i per fer servir les imatges per visualitzar-les dins de la galeria. En la figura 6 (Fig. 6) es pot veure el directori temporal.

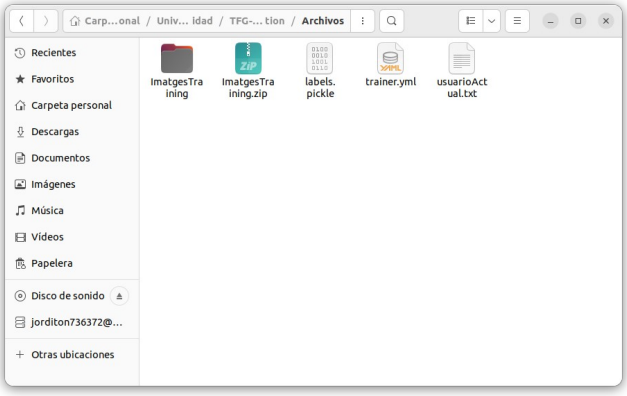


Fig. 6: Directori temporal.

6.5 Email de verificació

L'objectiu era enviar un email amb un número aleatori de quatre dígits per verificar un usuari. S'ha completat de manera satisfactòria. Es fa servir el servidor de gmail amb las dades del meu email personal per enviar un email al email que s'ha fet servir per registrar el nou usuari. Es pot veure un exemple de mail (Fig. 7).

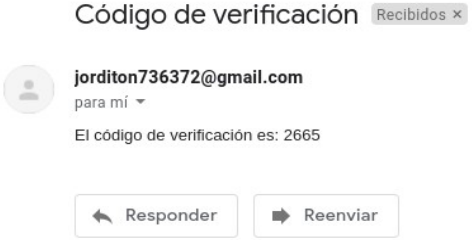


Fig. 7: Email de verificació amb codi.

6.6 Interfície gràfica

L'objectiu era crear una interfície gràfica còmode per l'usuari. Aquesta tasca està dividida en diferents subapartats ja que cadascuna és una plantilla *HTML* implementada de manera diferent. En l'apèndix, es pot

veure un diagrama de flux amb les rutes de l'aplicació (A.1).

Interfície d'iniciar sessió

És una tasca específica dins del desenvolupament de la interfície. Consistia en crear la interfície per iniciar sessió d'un usuari. S'ha complert satisfactòriament. La plantilla està composada per un gradient dins d'un arxiu *css* que serveix per estilitzar la plantilla. Dins d'això, hi ha un requadre amb tres camps obligatoris d'omplir que són els tres camps de l'usuari per iniciar sessió. A sota hi ha un botó per passar a la següent si s'han omplert tots els camps i les dades són correctes. Si l'usuari està verificat passa al menú principal, sinó a la plantilla de verificació. Finalment, a sota del tot hi ha un enllaç que porta a la plantilla de registre en cas de no tenir un usuari registrat. També s'ha afegit un petit detall que és el de fer servir una icona d'un ull que quan li dones, amb una funció de JavaScript, et mostra la contrasenya o l'oculta depenent de com estigui el text o no (la icona o opció que es fa servir a l'hora d'iniciar sessió o registrar-se per poder veure que s'està escrivint com a contrasenya o ocultar el text). Es pot veure a continuació una captura de pantalla mostrant la plantilla (Fig. 8).

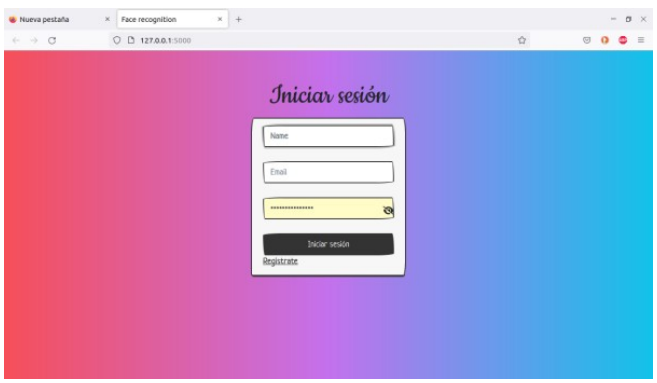


Fig. 8: Interfície gràfica part d'iniciar sessió.

Interfície registre d'usuari: És una tasca específica dins del desenvolupament de la interfície. Consistia en crear la interfície per registrar un usuari. S'ha complert satisfactòriament. La plantilla està composada exactament igual que la plantilla d'iniciar sessió. Es té en compte si un usuari està registrar o no amb la consulta de la col·lecció de *users* ja que un usuari s'ha de registrar prèviament. Però a part de fer servir un mètode de buscar de MongoDB, també fa servir el d'inserir. Un cop inserit, s'envia un email de verificació a l'usuari amb un número aleatori de quatre dígits per acabar de verificar l'usuari. Es pot veure a continuació una captura de pantalla mostrant la plantilla (Fig. 9).

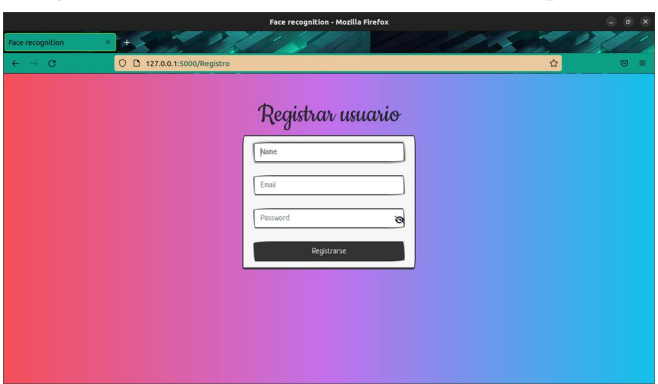


Fig. 9: Interfície gràfica part de registrar usuari.

Interfície verificació d'usuari: És una tasca específica dins del desenvolupament de la interfície. Consistia en crear la interfície per verificar un usuari. S'ha complert satisfactòriament. La plantilla està composada d'un arxiu *css* que es fa servir en totes las plantilles menys en la d'iniciar sessió i verificar usuari. Té una graella per omplir el codi de quatre dígits i després enviar. Si l'usuari posa un número incorrecte, se li envia a una plantilla amb un text d'error i un enllaç per tornar enrere. En cas contrari, s'actualitza el camp de vàlid dins de la col·lecció de MongoDB a true i se li envia a una plantilla amb opció d'anar al menú principal.

Interfície menú principal: L'objectiu era fer la part del menú principal una vegada un usuari entra i poder pujar imatges per poder entrenar el reconeixedor. S'ha pogut completar ja que en el menú només s'ha copiat la plantilla d'*HTML* de la part de registrar un usuari però amb un *css* diferent.

Interfície uploader: L'objectiu era fer la part del menú principal una vegada un usuari entra i poder pujar imatges per poder entrenar el reconeixedor. S'ha pogut completar. La plantilla està composada per un input de tipus file per pujar un arxiu *zip* per descomprimir-lo i pujar múltiples imatges d'una sola vegada. Un cop fet això, transformar-les a base64 per guardar-les al dataset per després tornar-les a passar a una *array* (una estructura de dades que és com una llista però amb dimensions fixes i contingut del mateix tipus de variable) i passar-les al reconeixedor.

Interfície entrenament: L'objectiu era implementar la part d'interfície gràfica d'entrenament del model de *Matchine Learning*. S'ha completat de manera satisfactòria. A la ruta de l'app que li correspon se li crida l'script d'entrenament passant-li com a paràmetres la llista de noms de las imatges de l'usuari i la col·lecció *fs* que conté la informació de les imatges, dins de l'script d'entrenament recuperen les imatges del *fs* i les fem servir per entrenar el model. Un cop entrenat el model, es guarda l'arxiu *pickle* i l'arxiu *yml* dins de la col·lecció *fs*.

Interfície predicció: L'objectiu era implementar la part d'interfície gràfica de predicció del model de *Matchine Learning*. S'ha completat de manera satisfactòria. A la ruta de l'app que li correspon se li demana de pujar una imatge a predir, un cop fet això cridem l'*script* de Python de predicció passant com a paràmetre la col·lecció fs, dins d'aquell *script* busquem els arxius *pickle* i *yml* i fem la predicció de la imatge. Un cop fet això, fem un *response* per poder fer la visualització de la imatge resultant amb el requadre i el nom. A continuació es pot veure el resultat

d'una predicció (Fig. 10)

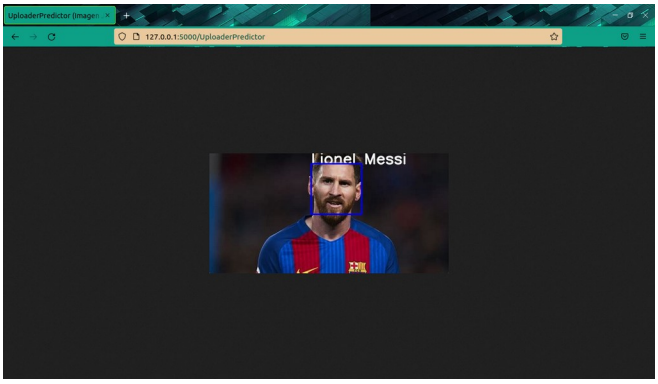


Fig. 10: Interfície gràfica part de predicció amb una foto de Lionel Messi.

Interfície galeria: L'objectiu era implementar la part d'interfície gràfica de galeria per poder visualitzar les imatges que s'han pujat en el dataset que serveixen per entrenar el model de *Matchine Learning*. S'ha completat però es pot millorar. A la ruta de l'app que li correspon imprimeix una llista per pantalla del nom de les imatges que s'han pujat i una graella amb el nom de la imatge que es vol visualitzar. Un cop es posa el nom, es visualitza la imatge. Cal mencionar que aquesta ruta es podria millorar per mostrar directament les imatges. A continuació es mostra una imatge de la galeria (Fig. 11).

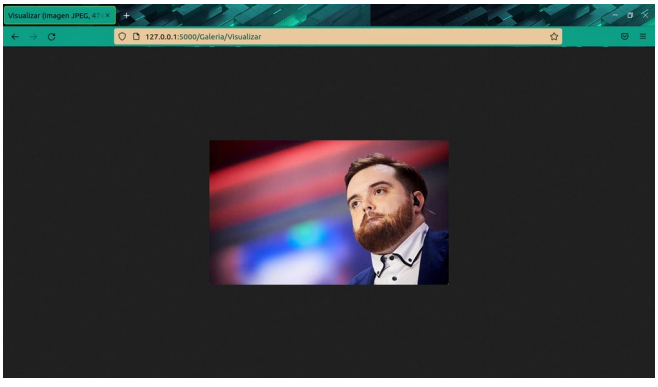


Fig. 11: Prova de la galeria amb una imatge d'Ibai Llanos.

6.7 Reestructuració de la carpeta de treball: L'objectiu era reestructurar millor la carpeta de treball i modificar els scripts perquè quedin el millor comentats i ordenats possible. S'ha completat l'objectiu de manera satisfactòria. S'ha afegit un arxiu readme.txt amb explicacions i instruccions de cada carpeta i arxiu dins de la carpeta de treball. També s'ha afegit un arxiu requirements.txt que serveix per instal·lar totes les llibreries de Python. Finalment, s'han creat més scripts de Python per repartir millor les tasques de cadascuna.

6.8 Test d'errors i solucionar-los: L'objectiu era solucionar *bugs* genèrics de l'aplicació perquè funcionés de manera correcte i estigués presentable. S'han pogut arreglar tots els *bugs* que hi havia. Las rutes connecten de manera correcta entre elles, cadascuna té un enllaç per tornar a la ruta anterior. En cas de les rutes que requereixen un ordre d'execució, com les del menú principal, se'ls hi ha posat una plantilla d'*HTML* de control d'error cadascuna, avisant a l'usuari de que s'ha de fer alguna cosa prèvia. També s'ha arreglat la part de visualització d'imatges, ja que abans no es visualitzaven i no era un error de sintaxis o d'escriure malament la ruta de la imatge, sinó de mètodes de la ruta, amb el *response* s'ha pogut solucionar el problema i es poden visualitzar les imatges. Una altra cosa que s'ha arreglat és guardar l'usuari actual, quan finalitzava d'executar la ruta de pujar les imatges, es reiniciaven totes les variables i es buidaven; per solucionar el problema, s'ha guardat l'usuari actual en un arxiu *txt* del qual es pot llegir en qualsevol moment, solucionant el problema.

7 Resultats

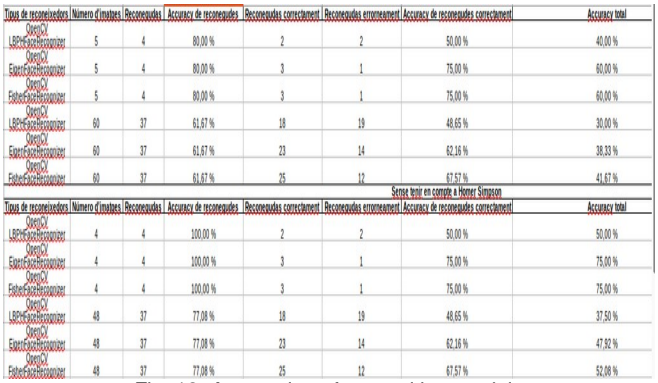
En aquesta part s'explica els resultats de comparar diferents *cascades* i reconeixedors d'OpenCV.

Com es pot veure en la figura 12, provant totes las cascades trobades, podem arribar a la conclusió de que la millor cascada és la de Frontalface_default amb més d'un 50% d'*accuracy*, tots amb els mateixos paràmetres. També cal destacar que es va provar amb un personatge de dibuixos animats i es va arribar a la conclusió de que el no diferenciar-se bé el contorn del cos, la cara no es detecta.

Tipus de cascade	Nom	Número d'imatges	Caras trobadas	Accuracy	Comentaris
HaarCascade	frontalcatface_extended	18	1	5,56 %	
HaarCascade	frontalcatface	18	1	5,56 %	
HaarCascade	frontalface_alt_tree	18	6	33,33 %	
HaarCascade	frontalface_alt	18	11	61,11 %	
HaarCascade	frontalface_alt2	18	12	66,67 %	
HaarCascade	frontalface_default	18	13	72,22 %	
HaarCascade	frontalcatface_extended	24	1	4,17 %	
HaarCascade	frontalcatface	24	1	4,17 %	
HaarCascade	frontalface_alt_tree	24	6	25,00 %	
HaarCascade	frontalface_alt	24	16	66,67 %	
HaarCascade	frontalface_alt2	24	17	70,83 %	
HaarCascade	frontalface_default	24	18	75,00 %	
LBPCascade	frontalface	24	12	50,00 %	
HaarCascade	frontalcatface_extended	48	1	2,08 %	
HaarCascade	frontalcatface	48	1	2,08 %	
HaarCascade	frontalface_alt_tree	48	6	12,50 %	
HaarCascade	frontalface_alt	48	27	56,25 %	
HaarCascade	frontalface_alt2	48	29	60,42 %	
HaarCascade	frontalface_default	48	30	62,50 %	
LBPCascade	frontalface	48	22	45,83 %	

Fig. 12: *Accuracies of cascades*

Com es pot veure en la figura 13. Els dos millors reconeixedors són els de Eigen faces i Fisher faces. Els dos tenen millor accuracy a l'hora de trobar cares i a l'hora de reconèixer a la persona. Tot i que aquí depèn també de quantes cares troba la cascada, s'ha fet servir en tots casos la de Frontal-face-default.



Tipus de reconeixedors	Número d'imatges	Reconegudes	Accuracy de reconegudes	Reconegudes correctament	Reconegudes erroneament	Accuracy de reconegudes correctament	Accuracy total
OpenCV LBPHFaceRecognizer	5	4	80,00 %	2	2	50,00 %	40,00 %
OpenCV EigenFaceRecognizer	5	4	80,00 %	3	1	75,00 %	60,00 %
OpenCV FisherFaceRecognizer	5	4	80,00 %	3	1	75,00 %	60,00 %
OpenCV LBPHFaceRecognizer	60	37	61,67 %	18	19	48,65 %	30,00 %
OpenCV EigenFaceRecognizer	60	37	61,67 %	23	14	62,16 %	38,33 %
OpenCV FisherFaceRecognizer	60	37	61,67 %	25	12	67,57 %	41,67 %

Sense tenir en compte a Homer Simpson

Tipus de reconeixedors	Número d'imatges	Reconegudes	Accuracy de reconegudes	Reconegudes correctament	Reconegudes erroneament	Accuracy de reconegudes correctament	Accuracy total
OpenCV LBPHFaceRecognizer	4	4	100,00 %	2	2	50,00 %	50,00 %
OpenCV EigenFaceRecognizer	4	4	100,00 %	3	1	75,00 %	75,00 %
OpenCV FisherFaceRecognizer	4	4	100,00 %	3	1	75,00 %	75,00 %
OpenCV LBPHFaceRecognizer	48	37	77,08 %	18	19	48,65 %	37,50 %
OpenCV EigenFaceRecognizer	48	37	77,08 %	23	14	62,16 %	47,92 %
OpenCV FisherFaceRecognizer	48	37	77,08 %	25	12	67,57 %	52,08 %

Fig. 13: Accuracies of recognition models

8 Conclusions

Ara, amb el treball finalitzat, ja es pot arribar a una conclusió definitiva.

Es pot arribar a la conclusió de que en aquest treball s'han tocat més conceptes dels esperats, no només la part del processament d'imatges, sinó també la part de *data managment* i *cloud computing* ja que la idea principal era fer només una aplicació per fer només el reconeixement de cares però ha acabat sent més complerta com, la part de guardar les dades d'usuaris i les diferents imatges o aprendre *HTML* per crear les diferents plantilles que es fan servir per la interfície gràfica de l'aplicació. Ha sigut un treball del qual estic bastant satisfet per la meva feina i per haver fet i après més del que esperava.

Tot i que m'emporto coses bones d'aquest treball, també m'emporto autocrítiques. La que es veu a simple vista és la de executar l'aplicació de manera local, l'aplicació funciona de manera correcta però potser en cas d'executar-se de manera que no local sorgeixen nous errors d'execució i sincerament, si es volgués comercialitzar aquesta aplicació seria millor que no fos per excutar localment. Una altra cosa a millorar és a l'hora de visualitzar les imatges de la galeria, una galeria normal es visualitzen directament totes les imatges, no es demana la imatge que es vol visualitzar, no es va poder trobar la manera de visualitzar totes les imatges d'una sola vegada. També una altra cosa a millorar és el fet de fer servir un directori temporal pel processament d'arxius, va ser una solució una mica simple. Finalment, Cal indicar que ha faltat temps per implementar la predicció amb vídeo i càmera dins de la interfície gràfica.

Amb tot això dit, vull transmetre que estic molt orgullós del meu treball i espero que en un futur sigui d'utilitat i continuï treballant amb ell.

Agraïments

Agrair a la meva coordinadora del grau Débora per deixar-me fer el TFG sobre el tema que jo mateix he proposat, a la meva tutora del TFG per portar el meu TFG i guiar-me per corregir els meus errors i millorar el meu treball, i al meu entorn d'amics i familiars per recolzar-me durant el procediment d'aquest treball.

Bibliografia

[1] Alalek, haar-cascade-files, (3 de gener del 2018). Distribuit per Github. https://github.com/opencv/opencv/tree/master/data/haarcascades

[2] Ashishprashar222, Flask_Mail, (25 de febrer del 2020). Distribuit per Github. https://github.com/Ashishprashar222/Flask_Mail

[3] Fazt Code, «Python Flask, React Hooks & MongoDB CRUD» (Abril 14, 2020). Accessed: Febrer 22, 2022, [Online video] Available: https://www.youtube.com/watch?v=D1W8H4Rkb9A&t=3194s

[4] Tips & Tricks, «Flask Mail - Email Verification - OTP Verification Using Flask - Send Mail Using Flask - Python» (Agost 18, 2020). Accessed: Abril 5, 2022, [Online video] Available: https://www.youtube.com/watch?v=ByoCkmilHg0&t=380s

[5] makeCode, «Mostrar Y Ocultar Contraseña | Vanilla JavaScript | makeCode» (Març 1, 2021). Accessed: Abril 6, 2022, [Online video] Available: https://www.youtube.com/watch?v=D8B7asBrJ3U&t=1s

[6] Aula Formativa, «Validar Formularios con HTML5» (Juny 11, 2014). Accessed: Abril 3, 2022, [Online video] Available: https://www.youtube.com/watch?v=MNf7gCVmY90&t=322s

[7] Ghost, «uiGradients» (Juliol 10, 2019). Accessed: Març 2, 2022, [Online] Available: https://uigradients.com/#Disco

[8] Robmadole, «Font awesome» (Febrer 6, 2018). Accessed: Abril 1, 2022, [Online] Available: https://fontawesome.com/

[9] Robmadole, «Font awesome libraries» (Febrer 6, 2018). Accessed: Abril 1, 2022, [Online] Available: https://cdnjs.com/libraries/font-awesome

[10] Xbpcb, «BootstrapCDN» (Septembre 22, 2021). Accessed: Març 2, 2022, [Online] Available: https://www.bootstrapcdn.com/

[11] Google, «Google fonts» (Gener 27, 2011). Accessed: Març 2, 2022, [Online] Available: https://fonts.google.com

[12] Alejandro Aldana, «Como subir imágenes y videos a MongoDB con Python y GridsFS» (Desembre 5, 2020). Accessed: Maig 10, 2022, [Online] Available: https://alejandroaldana99.medium.com/como-subir-im%C3%A1genes-y-videos-a-mongodb-con-python-y-gridfs-cff771a8a996

[13] Mi diario Python "Python y Flask: Subir archivos al servidor" (Gener 23, 2019). Accessed: Maig 15, 2022, [Online Video] Avaiable: https://youtu.be/LHXsBFWSdPo

[14] Mi diario Python "Python y Flask: Subir archivos al servidor" (Desembre 3, 2020). Accessed: Maig 17, 2022, [Online Video] Avaiable: https://youtu.be/Y95T_YE2ENg

[15] Mauro, «StackOverflow» (Agost 18, 2018). Accessed: Juny 17,

2022, [Online] Available: https://es.stackoverflow.com/questions/203662/recorrer-diccionario-en-la-plantilla

[16] Programador clic «Flask carga una imagen local y la muestra en la página», [En línia]. Disponible a: https://programmerclick.com/article/9321511456/

[17] J2LOGO «Tutorial Flask – Lección 16: Processar ficheros en Flask», [En línia]. Disponible a: https://j2logo.com/tutorial-flask-leccion-16-procesar-ficheros-en-flask/

[18] NoRelaX «Imágenes en bucle HTML», Gen. 2022. [En línia]. Disponible a: https://www.mediavida.com/foro/dev/imagenes-en-bucle-html-500374

[19] Decodigo.com «Python – Borrar un archivo», [En línia]. Disponible a: https://decodigo.com/python-borar-archivo

[20] Ricardo A, «StackOverflow» (Juny 1, 2016). Accessed: Juny 10, 2022, [Online] Available: https://es.stackoverflow.com/questions/15612/problema-al-insertar-imagen-en-html

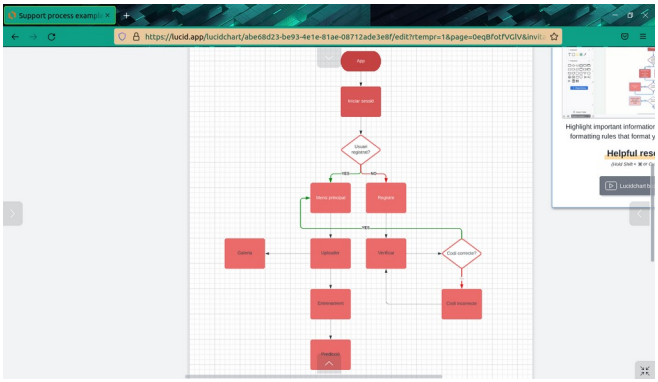
[21] P Raju, «StackOverflow» (Octubre 13, 2016). Accessed: Juny 4, 2022, [Online] Available: https://es.stackoverflow.com/questions/40015103/upload-file-size-16mb-to-mongodb

[22] Cut The Chaos, «How to display Multiple Image in Flask» (Juny 29, 2020). Accessed: Juny 13, 2022, [Online video] Available: https://www.youtu.be/fJMA9ac1IxE

[23] Codigofacilito, «Encriptar contraseñas en Python - Bytes» (Maig 11, 2017). Accessed: Juny 11, 2022, [Online video] Available: https://www.youtu.be/rVABbWlRuz8

APÈNDIX

A.1 DIAGRAMA DE FLUX



A.2 TAULA DE TASQUES

Tasca	Descripció	Durada	Completa %
Reconèixer els contorns d'una cara	Implementar un script de Python amb la finalitat de trobar el contorn d'una cara en una imatge o vídeo passat com a paràmetre o la mateixa càmera. Es farà servir la llibreria Opencv pel processament i el directori de Haar-Cascade per trobar el contorn	3 setmanes	100%
Interfície	Implementar scripts d'HTML, de CSS i de Javascript amb la finalitat de crear una interfície gràfica còmode per l'usuari	2 mesos	100%
Flask	Implementar un script de Python amb la finalitat de llançar l'aplicació al núvol. Es farà servir la llibreria Flask per llançar-la i crear les diferents rutes de l'app	1 setmana	100%
Dataset	Implementar un script de python amb la finalitat de manipular un dataset pels usuaris, ja sigui per consultar com per afegir de nous. Es farà servir la llibreria pymongo per fer els diferents algorismes	3 setmanes	100%
Model d'IA	Implementar un script amb la finalitat d'entrenar un model d'IA per entrenar-la amb les imatges de la cara i identificar a la persona de la cara. Es farà servir un reconeixedor de la llibreria d'OpenCV amb un arxiu tipus pickle que son els labels de cada persona a reconèixer i un arxiu tipus yml on és guardarà tota la informació de l'entrenament del model de «Matchine Learning» per després fer-lo servir per reconèixer un usuari.	3 setmanes	100%
Pujar imatges al dataset	Implementar un script d'HTML i python per pujar imatges i guardar-les en el dataset. Es farà servir la llibreria Gridfs per pujar las imatges al dataset de MongoDB i tornar-les a recuperar per fer-les servir. També es farà servir la llibreria Base64 i ByteIO per codificar les	2 setmanes	100%

	imatges per poder guardar-les i descodificar per poder fer-les servir.		
Visualització d'imatges en l'aplicació	Implementar un script d'HTML i python per poguer visualitzar tant les diferents imatges que s'han pujat en el dataset com la imatge resultant de fer una predicció, amb un requadre enmarcant la cara de la persona i un texte a sobre amb el seu nom. Es farà servir la llibreria response per poguer visualitzar la imatge desitjada dins de l'aplicació	2 setmanes	100%
Enviar email de verificació d'usuari	Implementar un script d'HTML i de python per, un cop registrat un usuari, enviar un email amb un codi de quatre dígits per comprovar si les dades son correctes. Es farà servir la llibreria random per generar quatre dígits de manera aleatòria i guardar-les en una variable per després comprovar si el codi posat per l'usuari coincideix amb el codi demanat. Un altre llibreria que es farà servir és la de smtplib que servirà per definir el servidor de mail que es farà servir i les diferents dades que demanen per enviar un mail. També es farà servir la llibreria correu electrònic per definir el texte del mail.	1 setmana	100%
Grau de maduresa			100%