UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DSIC DEPARTAMENT DE SISTEMES INFORMÀTICS I COMPUTACIÓ

MIARFID Máster en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital

Departament de Sistemes Informàtics i Computació
Universitat Politècnica de València

Ús d'emocions socials per a la millora de la interacció humà-agent

TREBALL FI DE MÀSTER

Màster Universitari en Intel·ligència Artificial, Reconeixement de Formes i Imatge Digital

Autor: Christian Verdú Mateu

Tutor: Vicente Julián Inglada
Carlos Carrascosa Casamayor

Curs 2017-2018

Resum

Actualment, l'habilitat per reconèixer les persones en les fotografies, així com, el reconeixement d'emocions es trivial per als éssers humans. Per aquest motiu, un sistema automàtic amb aquesta capacitat seria un gran pas en el camí cap a la interacció persona-màquina. En aquest treball es descriu una aproximació per a intentar resoldre aquest problema. A més a més, el sistema deu poder-se utilitzar en plataformes del núvol, ha de ser desplegable automàticament, oferir una plataforma web per a la gestió interna i enregistrar les diferents emocions per les quals transita una persona. Per tant, obtindrem un producte destinat a identificar les emocions i les persones per tal de formalitzar l'estat emocional d'un grup. Aquest estat, pot ser usat per un humà per identificar possibles estímuls negatius que afecten a les persones i proporcionar-li dades per prendre decisions a l'hora d'homogeneïtzar l'estat emocional d'un grup de persones.

Paraules clau: aprenentatge, automàtic, reconeixement, expressió, facial, processament, imatges, svm, ann, mlp, mas, multiagent, model, social, pad

Resumen

Actualmente, la habilidad para reconocer las personas en las fotografías, así como, el reconocimiento de emociones es trivial para los seres humanos. Por este motivo, un sistema automático con esta capacidad sería un gran paso en el camino hacia la interacción persona-máquina. En este trabajo se describe una aproximación para intentar resolver este problema. Además, el sistema debe de poderse utilizar en plataformas de la nube, tiene que ser desplegable automáticamente, ofrecer una plataforma web para la gestión interna y grabar las diferentes emociones por las cuales transita una persona. Por lo tanto, obtendremos un producto destinado a identificar las emociones y las personas para formalizar el estado emocional de un grupo. Este estado, puede ser usado por un humano para identificar posibles estímulos negativos que afectan a las personas y proporcionarle datos para tomar decisiones a la hora de homogeneizar el estado emocional de un grupo de personas.

Palabras clave: aprendizaje, automático, reconocimiento, expresión, facial, procesamiento, imágenes, svm, ann, mlp, mas, multiagente, modelo, social, pad

Abstract

Currently, the ability to recognize people in photographs, as well as, the recognition of emotions is trivial for humans. For this reason, an automatic system with this capability would be a great step on the way to human-machine interaction. This paper describes an approach to try to solve this problem. In addition, the system must be able to be used in cloud platforms, it must be automatically deployable, it must offer a web platform for internal management and it must record the different emotions through which a person transits. Therefore, we will obtain a product designed to identify emotions and people to formalize the emotional state of a group. This state can be used by a human to identify possible negative stimuli that affects people and it provides data to make decisions when homogenizing the emotional state of a group of people.

Key words: machine, learning, emotion, face, expression, recognition, image, processing, svm, ann, mlp, mas, multi-agent, social, model, pad

Índex

Índex de figures

Índex de taules

CAPÍTOL 1
Introducció

Actualment, la forma d'interactuar amb les màquines està canviant de forma dràstica. Per exemple, a les videoconsoles, s'està abandonant el comandament físic per càmeres que reconeixen els gestos i moviments per permetre altra forma d'interactuar amb el joc. També existeixen moltes aplicacions, capaces de reconèixer objectes i cares, que permeten una interacció més humana amb el maquinari. Si ens parem a pensar, la comunicació humana no és sols la comunicació verbal i l'escrita; la comunicació gestual és la que més ens descriu, la que més importància aporta a la comunicació. No podem oblidar que, si existeix un llenguatge universal, aquest és el de les emocions. Es per això que, en aquest projecte, tractarem el tema de com implementar un sistema multiagent capaç de reconèixer emocions individuals i transformar-les a un espai grupal. Aquesta emoció grupal ens proporcionarà informació per a prendre decisions per tal de proporcionar-li dades a un sistema qualsevol i que aquest puga prendre les decisions oportunes depenent del domini en el qual s'estiga desenvolupant.

1.1 Motivació

Amb aquesta contribució, s'intenta ampliar el Treball Fi de Grau (**TFG**), és a dir, aportar una nova forma d'interactuar amb el maquinari, que sumada a altres formes d'interacció, ens aporte un altre punt de vista respecte a la comunicació persona-màquina. Una perspectiva de comunicació més humana que, aplicada a la vida real, pot nodrir futurs projectes on, les emocions de les persones, són un dels factors més importants a tindre en compte. Un dels dominis d'aplicació d'un sistema de reconeixement d'emocions és la monitorització de persones amb discapacitat, persones menudes i persones majors per tal d'obtindre dades d'interès enfront a nous estímuls externs. També podríem parlar de la integració d'aquest sistema amb el reproductor de música per tal de reproduir cançons més adients al nostre estat anímic. És, també, una gran motivació el contribuir en una línia d'investigació del Grup de Tecnologia Informàtica - Intel·ligència Artificial (*GTI-IA*) de la Universitat Politècnica de València, degut a que, aquest treball, forma part d'un projecte d'investigació finançat amb el programa Prometeo de la Generalitat Valenciana (PROMETEOII/2013/009).

1.2 Objectius

L'objectiu principal d'aquest projecte és el **reconeixement d'emocions** i de **persones** en un temps raonable, que siga multiplataforma, desplegable en una plataforma del núvol

i fàcil d'implantar. A més a més, per aconseguir aquest objectiu principal haurem de definir uns subobjectius.

- **Estudi d'entorns de virtualització**. Estudiar quines opcions tenim a l'hora d'implementar la nostra plataforma en un entorn virtualitzat.
- **Anàlisi de les diferents tècniques de reconeixement d'identitat**. Analitzar les tècniques d'aprenentatge automàtic que s'adapten millor al nostre corpus per identificar a les persones.
- **Reconèixer l'emoció i la identitat en un temps raonable**. Una interacció més fluida entre el sistema i la persona, predisposarà a una percepció més agradable i més humana per part de la persona. A més, com que s'han de fer moltes operacions en cada instant, convé que la resposta siga ràpida. Per aquest motiu, el temps de resposta no deurà ser superior als 10s.
- **Experimentació sobre la plataforma**. Es plantejaran diferents experiments al llarg del projecte per tal de validar la correcta implementació i funcionament del sistema.
- **Obtindre una precisió superior al 90%**. Tant el reconeixement d'emocions, com el reconeixement de la identitat, han de tindre una precisió superior al 90% per a les emocions.
- **Utilitzar un sistema multiagent**. Per tal de facilitar la comunicació, l'escalabilitat i la portabilitat a altres llenguatges, la plataforma haurà d'implementar-se mitjançant agents intel·ligents.

1.3 Estructura de la memòria

La resta d'aquest escrit s'organitza principalment en set capítols de la forma següent:

- El **Capítol 2: Estat de l'art** revisa breument la situació de les tècniques i algorismes d'aprenentatge automàtic que s'utilitzen actualment. A més a més, proporciona alguns coneixements bàsics referents als sistemes multiagent i a la situació actual de la robòtica. Aquest capítol també revisa treballs relacionats amb la problemàtica del reconeixement d'emocions. Per altra banda, els mètodes de validació i les mètriques emprades en aquest projecte són explicades al final del capítol.
- El **Capítol 3: Sistema multiagent per al reconeixement d'emocions d'un grup de persones: Administració, gestió i provisió d'API** revisa breument la descripció del problema així com les bases de dades utilitzades en aquest treball. També s'explica el procés de cerca de característiques, incloent el procés d'extracció de punts característics i càlcul de distàncies.
- Al **Capítol 4: Arquitectura** s'explica com es s'estructura la nostra plataforma, és a dir, com es disposen les parts fonamentals del sistema. A més a més, es definirà el funcionament de cada mòdul i les entrades i eixides d'aquest.
- El **Capítol 5: Experimentació** té com objectiu descriure la validació i experiments realitzats sobre la plataforma.
- Al **Capítol 6: Casos d'ús** veurem unes possibles aplicacions a la vida real on la nostra plataforma adquiriria bastant rellevància.

- El **Capítol 7: Conclusions i treball futur** resumeix de quina manera i com es compleixen els objectius plantejats en el present treball. A més a més, estableix les conclusions i esbossa el treball futur.

CAPÍTOL 2
Estat de l'art

En aquest capítol analitzarem els distints articles que tenen a veure amb el reconeixement d'emocions i la representació d'aquestes per a grups de persones utilitzant models d'emocions grupals.

2.1 Treballs relacionats amb emocions, models d'emocions individuals i grupals

Per a l'extracció o reconeixement d'emocions, Sohail i Bhattacharya (2008) fa una aproximació utilitzant distàncies entre elements de la cara (boca, ulls, nas...) per tal de crear un classificador capaç de predir quina emoció s'està representant.

Per una banda, segons l'article de Meftah, Thanh i Amar (2010), cada emoció és modelable mitjançant una representació algebraica d'emocions. Per tant, podem representar totes les emocions com a vector en un espai de 3 dimensions on cada eix representa una emoció bàsica. Aquest model multidimensional permet representar infinitat d'emocions i proporciona poderoses eines matemàtiques per a l'anàlisi i el processament d'aquestes emocions.

A més a més, en Mehrabian (1996), l'autor revisa l'evidència sobre el model d'estat emocional del *Pleasure-Arousability-Dominance* (PAD), o Plaer-Excitabilitat-Dominància en valencià, i va demostrar que les seves tres dimensions gairebé ortogonals proporcionaven una descripció prou àmplia d'estats emocionals. Per això, ofereix fórmules per a l'ús de puntuacions de temperament de P, A i D per calcular i predir una varietat de puntuacions de personalitat (per exemple, ansietat, depressió, pànic, somatització, empatia, afiliació, assoliment, extroversió, cercador d'ansietat, solitud, neuroticisme, suïcidi, alimentació irritant, abús de substàncies, estabilitat emocional, dependència, agressivitat i confiança). També, en Bakker et al. (2014) fa una altra anàlisi utilitzant aquest model.

Per altra banda, en Huang, Ali i Liao (2017) aplica el model PAD teòric al domini dels videojocs per tal d'interactuar amb ells mitjançant el llenguatge no verbal, els gestos de la boca.

Respecte als models grupals, existeixen articles que permeten extreure l'emoció social d'un grup d'entitats intel·ligents (J. Rincon, Prieta et al. 2017) i (J. A. Rincon, Julian i Carrascosa 2015). També, segons J. Rincon, Costa et al. (2018), les emocions d'un individu d'un grup de persones pot contagiar les demés, per tant, es pot usar el model PAD per identificar aquestes situacions.

2.2 Docker

El *Cloud Computing*, o computació en el núvol, està intrínsecament arrelat en les tecnologies de virtualització. Recentment, les noves tecnologies de virtualització lleugeres, com ara els contenidors, s'han tornat cada vegada més populars i són avui dia una part essencial de les ofertes del núvol. Els contenidors també s'integren estretament en el sistema operatiu de l'amfitrió, reduint les càrregues de programari impostes per màquines virtuals (VM) (Xavier et al. 2013). No obstant això, diversos factors, inclosa l'acceleració del cicle de desenvolupament (com ara mètodes àgils i DevOps), una pila d'aplicacions cada vegada més complexa (principalment serveis web i els seus marcs), i la pressió del mercat per densificar les aplicacions dels servidors han provocat la necessitat d'una solució ràpida i senzilla per utilitzar la forma d'impulsar el codi a la producció.

El contenidor ofereix la possibilitat d'executar múltiples versions de les aplicacions en la mateixa màquina (vegeu la figura 2.1). Les noves instàncies de contenidors es poden crear de manera instantània per afrontar un màxim de la demanda del client, que és convenient per generar aplicacions sota demanda o per moure ràpidament un servei.

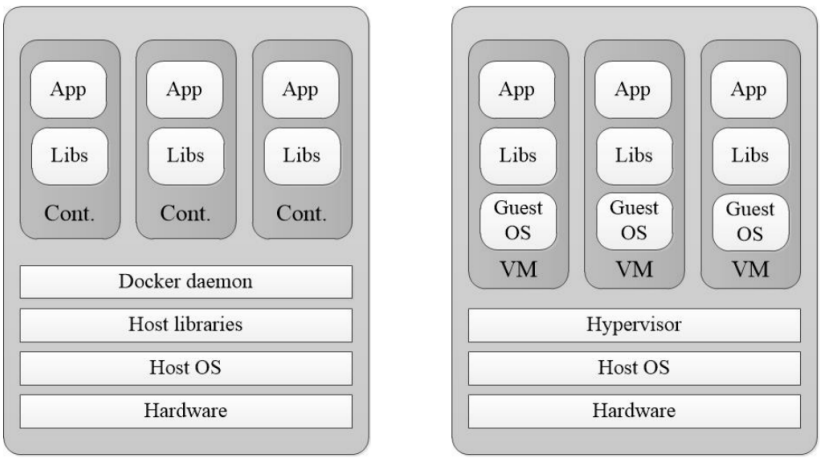

Figura 2.1: Comparació del model d'execució de l'aplicació per a l'amfitrió Docker amb el servidor VM

Els contenidors es poden integrar en un entorn multiinquilí, augmentant així l'ús mitjà dels recursos compartits del maquinari. Això s'aconsegueix compartint el nucli amb la màquina amfitriona. A diferència de les VM, els contenidors no incrusten el seu propi nucli, sinó que s'executen directament al nucli de l'amfitrió. Això escurça la ruta d'execució de cridades al sistema eliminant el nucli invitat i la capa de maquinari virtual. A més, els contenidors poden compartir recursos de programari (com ara llibreries) amb l'amfitrió, evitant així la duplicació de codi. L'absència de nucli i algunes llibreries del sistema fan contenidors molt lleugers (les mides d'imatge es poden reduir a uns quants megabytes), cosa que permet un procés d'arrencada ràpid. En el nostre projecte utilitzarem `Docker` com a base per construir la nostra plataforma.

2.2.1. `Dockerfile`

`Dockerfiles` permet als usuaris especificar una imatge base i una seqüència d'ordres que es realitzaran per construir la imatge, juntament amb altres opcions, com els ports exposats específics de la imatge. La imatge es construeix amb l'ordre de compilació de l'arrencador automàtic llegint les instruccions d'un fitxer `Dockerfile`. Els fitxers `Dockerfile`

```
1 FROM ubuntu
2 MANTENIMENT Christian Verdú (chverma@upv.es)
3 RUN apt-get update
4 RUN apt-get install-i nginx
5 EXPOSE 80
```

Figura 2.2: Exemple del fitxer Dockerfile

comencen per definir una imatge a partir de la qual (`FROM`) s'inicia el procés de compilació. Seguit per diversos mètodes, ordres i arguments (o condicions).

En l'exemple (Figura 2.2), estem construint una imatge que executarà el servidor *proxy* `nginx`. Després de l'estàndard `FROM`, el `MANTENIMENT` és una ordre que no s'executa, declara l'autor i, per tant, estableix el camp de l'autor de les imatges. No obstant això, ha de venir després de FROM.

A més, s'utilitza una instrucció `RUN` per executar qualsevol ordre. En aquest cas estem executant una actualització de paquets i després instal·lant `nginx`. Estem utilitzant l'ordre `EXPOSE` ací per informar a quin port s'escoltarà el contenidor. La llista completa de comandaments del fitxer `Dockerfile` es pot trobar al lloc web Docker (2018).

2.2.2. `Docker image`

Una imatge és un fitxer de plantilla inert, immutable i de només lectura que és essencialment una instantània d'un contenidor i es creen amb l'ordre `docker build` per a un fitxer `dockerfile`. Com que poden arribar a ser bastant grans, les imatges estan dissenyades per estar compostes de capes d'altres imatges, permetent enviar una quantitat mínima de dades quan es transfereixen imatges a través de la xarxa. Les imatges `Docker` són el component de compilació de `Docker`.

2.2.3. `Docker container`

Una instància d'una imatge s'anomena contenidor. En altres paraules, si inicieu una imatge, teniu un contenidor d'execució d'aquesta imatge. Els contenidors `Docker` són similars a un directori on tenen tot el que es necessita perquè una aplicació s'execute. Cada contenidor es crea des d'una imatge `Docker`. Els contenidors `Docker` es poden executar, iniciar, aturar, moure i esborrar. Cada contenidor és una plataforma d'aplicacions aïllada i segura. Els contenidors `Docker` són el component d'execució de `Docker`. Un dels seus punts forts és la possibilitat d'executar molts contenidors aïllats de la mateixa imatge.

2.2.4. `Docker daemon`

El programari `Docker` s'executa com un dimoni a la màquina *host*. Pot llançar contenidors, controlar el seu nivell d'aïllament, controlar-los per activar accions (com reiniciar-se) i generar `shells` a contenidors en funcionament amb finalitats d'administració. El programari pot canviar les regles de taules IP a l'*host* i crear interfícies de xarxa. També és responsable de gestionar imatges del contenidor, incloent ordres com `push` i `pull` d'imatges en un registre remot (com el `Docker hub`).

2.2.5. `Docker hub`

El dipòsit en línia anomenat `Docker hub` permet als desenvolupadors carregar les imatges `Docker` i descarregar-les posteriorment. Els desenvolupadors poden registrar-se per obtenir un compte gratuït, on tots els dipòsits seran públics. Els repositoris de desenvolupadors tenen un nom d'espai on seu nom és "desenvolupador/dipòsit". També existeixen repositoris oficials, proporcionats directament per `Docker`.

Per tant, usarem els contenidors `Docker` per contindre els diferents elements que formaran part de la nostra plataforma per a l'ús d'emocions socials. Açò ens proporcionarà un marc per tal de dissenyar, implementar i desplegar cada servei que formarà part de la nostra plataforma.

2.3 Sistemes Multiagent

Un sistema multiagent (*SMA*), segons Wooldridge (2009) i Russell i Norvig (2010) és un sistema format per diversos agents intel·ligents. Aquests interactuen entre si dins d'un entorn determinat. A més a més, poden ser programats per realitzar tasques amb unes metes i objectius comuns. També, cal afegir que redueixen la complexitat de realitzar feines que serien més costoses de realitzar per un sol agent.

Per tant, podem definir com a l'element més important, dins del sistema multiagent, l'agent. A continuació, en la Figura 2.3 podem visualitzar el concepte d'agent inte·ligent de forma gràfica. A més a més, podem definir l'agent com una entitat capaç de percebre el seu entorn, processar les percepcions recuperades i actuar de manera racional, per interaccionar amb l'entorn. També, els agents posseeixen un mecanisme de comunicació per comunicar-se entre si.

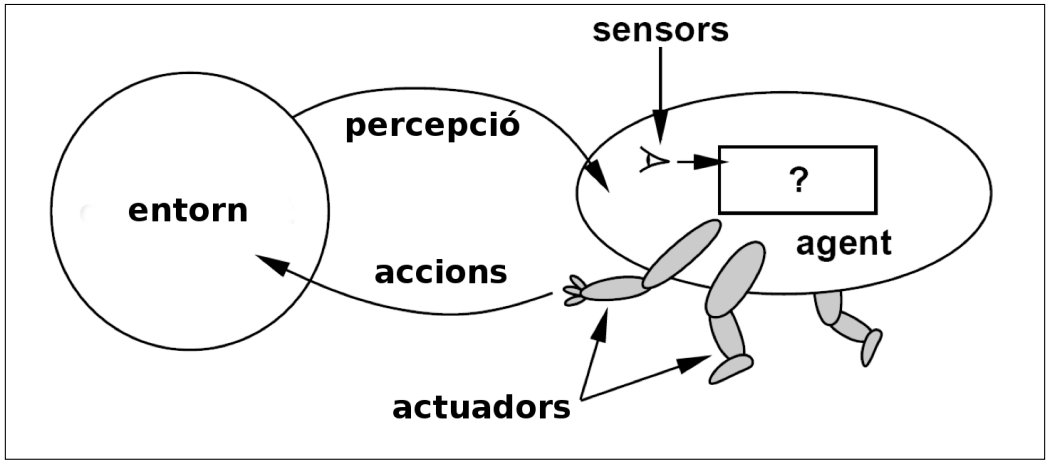

Figura 2.3: Representació gràfica d'un agent intel·ligent

Atès a la complexitat d'aquests, podem distingir tres tipus bàsics:

- Agents passius. Són agents molt simples que no disposen de cap objectiu.
- Agents actius. Són agents intel·ligents que disposen d'un objectiu simple, per exemple, esperar un missatge i reenviar-lo.

- Agents complexos. Són agents actius on el objectiu a assolir no és tan simple. Són capaços de realitzar càlculs complexos, així com de dur a terme un gran ventall de funcionalitats.

Els agents intel·ligents tenen tres característiques importants:

- Autonomia. Els agents són, almenys, parcialment autònoms.
- Visió local. Cap agent té una visió global del sistema perquè es massa complexe per a un agent fer un ús pràctic d'eixos coneixements.
- Descentralització. No hi ha un agent de control designat, el sistema està distribuït.

2.3.1. *SPADE*

Una vegada hem revisat com es formen els sistemes multiagent i la seua unitat mínima, en aquest treball usarem la *Smart Python Agent Development Environment* (SPADE)[1], desenvolupada pel grup GTI-IA de la *UPV*. SPADE és una plataforma multiagent que habilita de forma simple la interacció entre agents intel·ligents, ja que està basada en la tecnologia de missatgeria instantània *XMPP*, que és un estàndard en missatgeria. Aquest últim concepte és fonamental ja que els missatges que intercanvien els agents s'agrupen en converses, cosa que permet separar i identificar el contingut de la informació transmesa. A més d'açò, *SPADE* ofereix un parell d'utilitats per a crear un sistema multiagent. Existeix a la disposició del desenvolupador una llibrearia per a *Python* que ajuda en la creació d'agents i en la implementació dels mètodes necessaris per a rebre i enviar missatges en el sistema.

A més a més, disposa d'una interfície web per a gestionar la plataforma i els agents que estiguen connectats a ella, a més de mantenir un històric de les converses. Parlant amb més detall de la llibreria, permet usar models d'agents que compten amb mecanismes per a connectar a la plataforma, un emissor de missatges i un conjunt de comportaments bàsics per a l'agent. Com es pot observar, aquestes característiques afavoreixen bastant la tasca de la implementació, ja que donen feta una part bastant gran del procés de crear un sistema multiagent.

[1] Consulta https://github.com/javipalanca/spade per a més detalls de com obtindre SPADE

CAPÍTOL 3

Sistema multiagent per al reconeixement d'emocions d'un grup de persones: Administració, gestió i provisió d'API

En aquest capítol descriurem el problema al qual s'enfrontem. Seguidament, es detallaran els subproblemes que s'han anat encontrant, com ho són el conjunt de mostres a utilitzar, la selecció de característiques i la generació de distàncies. Finalment, exposarem unes conclusions extretes d'aquest capítol.

3.1 Descripció del problema

El reconeixement d'expressions facials (emocions), el reconeixement de la persona que les realitza i la gestió d'aquestes dades formant un sistema de Reconeixement d'emocions d'un grup de persones és l'objectiu principal d'aquest projecte i, per tant, com obtindre aquestes emocions i la identificació de les persones, mitjançant tècniques d'imatge digital, es converteixen en els problemes principals. Una part d'aquest problema ja es va abordar en el Treball de Fi de Grau (Verdú 2016) i n'és el del reconeixement d'emocions. No obstant, aquest sistema presentava moltes inconveniències a l'hora de la disponibilitat del servei, ja que usava molts recursos, i sobretot memòria, moltes vegades es penjava i era necessari reiniciar la màquina. A més a més, tampoc s'enregistrava ninguna dada respecte a les emocions detectades, imatges analitzades... Tampoc se'n feia ús del reconeixement de la persona, és a dir, qui és qui, com identificar-lo inequívocament. Per altra banda, tampoc tenim un sistema per gestionar les emocions d'un grup de persones, per aquest motiu, es decideix ampliar el treball fet anteriorment. Per tant, agafant el codi anterior, hi ha que integrar-lo en una mena de sistema de virtualització, on la posada en funcionament del sistema siga ràpida, no requerisca de massa configuració per part del usuari, siga escalable, multiplataforma i tolerant a fallades. També volem construir un sistema modulable basat en l'arquitectura de microserveis. Atès a aquests requeriments, desenvoluparem el nostre sistema baix la capa de virtualització de `Docker`. Açò ens proporcionarà la infraestructura i els recursos per tal de desplegar contenidors per tal d'implementar cada microservei o mòdul.

No podem obviar que el sistema necessitarà d'una interacció per part d'un usuari administrador, el qual gestionarà els resultats de la plataforma i podrà interactuar amb cada funcionalitat que implemente el sistema. Ja que tenim aquests requeriments, decidirem

desenvolupar una plataforma web que interactue amb el sistema. Aquesta plataforma proveirà de les ferramentes necessàries per a gestionar la plataforma, consultar el estat i fer peticions al sistema.

Per totes les característiques i requeriments anteriorment descrits caldrà desenvolupar el software corresponent per a gestionar el sistema complet, és per això que en els següents capítols explicarem el funcionament pas a pas del sistema així com les comunicacions i processos que es duen a terme internament. A continuació, parlarem dels subproblemes trobats i com s'han resolt per a sol·lucionar-los. El distins subproblemes es podrien categoritzar en:

- Primer, cercar en la imatge totes les cares que hi apareguen.
- Segon, concentrar-se en cada fotografia i ser capaç de saber si una imatge d'una cara, encara que estiga en una direcció o posició estranya, continua siguen la mateixa cara.
- Tercer, ser capaç d'extraure unes característiques úniques de cada cara per a diferenciar-la d'altres cares.
- Quart, ser capaç d'extraure unes característiques úniques de cada cara per saber quina emoció s'està representant.
- Finalment, comparar les característiques úniques de la cara actual amb la resta de cares conegudes per a determinar el nom de la persona. A més a més, classificar, mitjançant altres característiques obtingudes, a quina emoció correspon eixe vector de característiques.

Per tant, abordarem aquest problema pas a pas. Per a cada pas, s'explicarà com s'ha resolt i l'algorisme d'aprenentatge automàtic que s'ha usat en cada cas.

3.1.1. Pas 1: Trobar totes les cares

El primer pas en el nostre oleoducte és la detecció de rostres. Òbviament, hem de situar les cares en una fotografia abans de poder intentar distingir-les.

La detecció de rostres es va incorporar a principis dels anys 2000 quan Paul Viola i Michael Jones (Viola i Jones 2001) van inventar una manera de detectar cares que era prou ràpida per funcionar en càmeres barates. No obstant això, hi ha solucions molt més fiables. Anem a utilitzar un mètode inventat el 2005 anomenat Histograma d'Orientats Gradients (HOG).

Per trobar cares en una imatge, començarem fent que la nostra imatge siga en blanc i negre, perquè no necessitem dades de colors per trobar cares. Podem observar un exemple en la Figura 3.1.

A continuació, per a cada píxel d'una imatge, volem observar els píxels que l'envolten tal i com es mostra en la Figura 3.2.

Tot seguit, el nostre objectiu és esbrinar la foscor comparant el píxel actual amb els píxels que l'envolten directament. A continuació, volem dibuixar una fletxa que mostre en quina direcció la imatge s'està enfosquint, com en la Figura 3.3.

Si repetim aquest procés per a cada píxel de la imatge, acabarem amb cada píxel reemplaçat per una fletxa. Aquestes fletxes es diuen gradients i mostren el flux de la llum a la foscor a tota la imatge. Un exemple el podem trobar a la Figura 3.4.

Figura 3.1: Imatge d'una cara

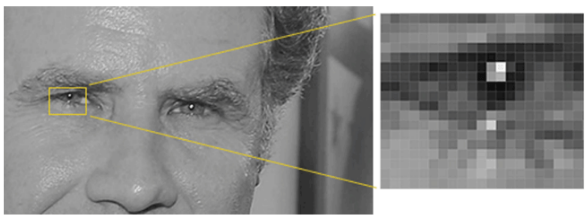

Figura 3.2: Zona de píxels

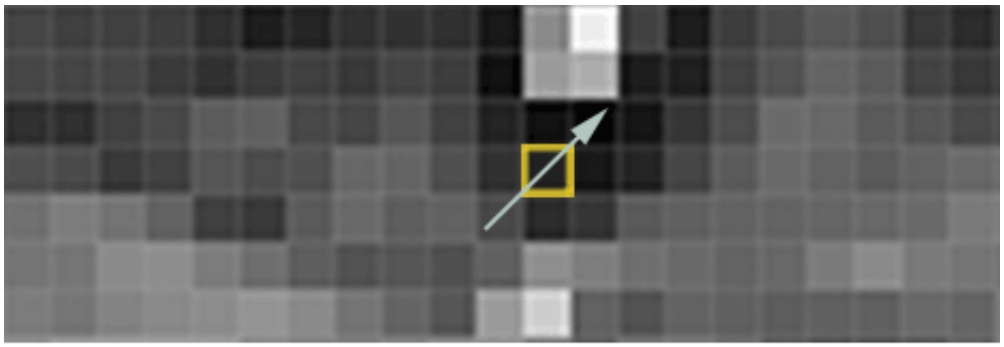

Figura 3.3: Detecció de gradient

Això pot semblar un procés aleatori, però hi ha una bona raó per substituir els píxels per gradients. Si analitzem píxels directament, les imatges fosques i les imatges sobreil·luminades de la mateixa persona tindran valors de píxels totalment diferents. Però només tenint en compte la direcció que canvia la brillantor, tant les imatges realment fosques com les imatges realment brillants acabaran amb la mateixa representació exacta. Això fa que el problema siga molt més fàcil de resoldre.

Però guardar el gradient per a cada píxel ens dóna massa detalls. Seria millor si poguéssem veure el flux bàsic de brillantor/foscor a un nivell superior per poder veure el patró bàsic de la imatge.

Per fer-ho, dividirem la imatge en petits quadrats de 16x16 píxels cadascun. A cada quadrat, explicarem quants gradients apunten en cada adreça principal (quants punts apunten, apunten cap a la dreta, apunten cap a la dreta...). A continuació, substituirem aquest quadrat a la imatge amb les instruccions de la fletxa més fortes.

El resultat final és convertir la imatge original en una representació molt senzilla que captura l'estructura bàsica d'una cara d'una manera senzilla, com es pot veure a la Figura 3.5.

Figura 3.4: Gradients

Figura 3.5: Imatge on s'aplica el filtre HOG

Per trobar cares en aquesta imatge, tot el que hem de fer és trobar la part de la nostra imatge que es veu més semblant a un patró HOG conegut que es va extraure d'un grup d'altres rostres d'entrenament. Un exemple es pot veure en la Figura 3.6.

Amb aquesta tècnica, ara podem trobar cares en qualsevol imatge, com mostra la Figura 3.7.

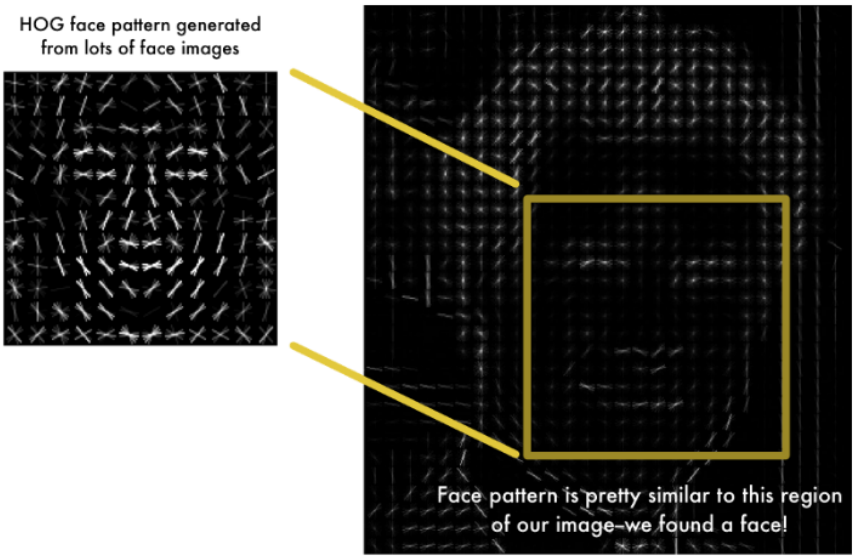

Figura 3.6: Filtre d'HOG aplicat

Figura 3.7: Trobant cares

3.1.2. Pas 2: Posar i projectar cares

En aquest pas, aïllem les cares de la nostra imatge. Però ara hem de lluitar amb el problema que les cares tornades en direccions diferents es veuen totalment diferents per un ordinador. Podem trobar un exemple a la Figura 3.8.

Per tenir açò en compte, tractarem de deformar cada foto on els ulls i els llavis estiguen sempre en el lloc base en l'imatge. Açò ens facilitarà molt la comparació de cares en els pròxims passos.

A continuació, utilitzarem un algorisme anomenat estimació fita. Hi ha moltes maneres de fer-ho, però anem a utilitzar l'enfocament inventat el 2014 per Vahid Kazemi i Josephine Sullivan (Kazemi i Sullivan 2014a).

La idea bàsica és que obtindrem 68 punts específics (anomenats punts de referència) que existeixen a cada cara: la part superior de la barbeta, la vora externa de cada ull, la vora interna de cada cella, etc. A continuació, entrenarem un algorisme d'aprenentatge

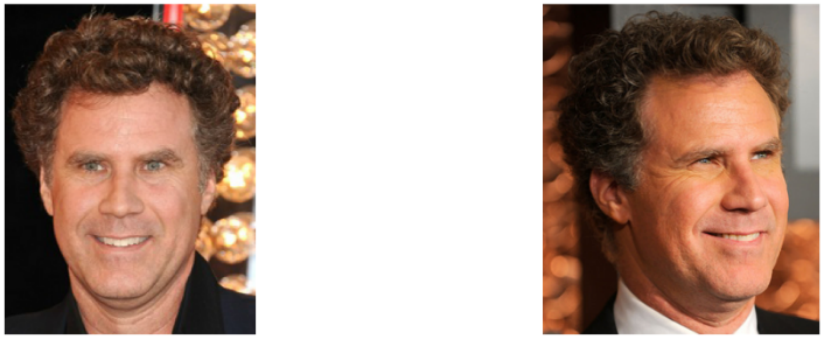

Figura 3.8: Cares amb diferents direccions, diferents enfocaments

automàtic per poder trobar aquests 68 punts específics en qualsevol rostre. Els punts els podem observar a la Figura 3.9.

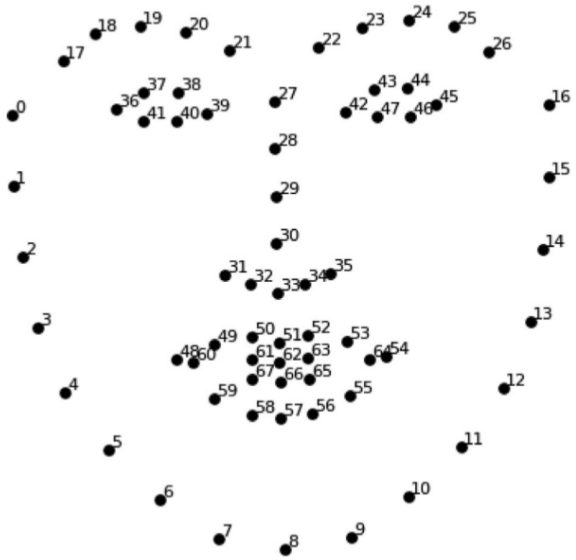

Figura 3.9: Punts de referència

A continuació, podem trobar el resultat de localitzar els 68 punts sobre la nostra imatge (Figura 3.10).

Ara que sabem on són els ulls i la boca, simplement girarem, escalarem i tallarem la imatge perquè els ulls i la boca estiguin centrats al millor possible. Només utilitzarem transformacions d'imatge bàsiques com la rotació i l'escala, que preserven línies paral·leles (anomenades transformacions afines), com podem observar en la Figura 3.11

En aquest punt, no importa com es gira la cara, som capaços de centrar els ulls i la boca i posar el patró de punts en la mateixa posició, és a dir, estandarditzem la posició de la cara. Això farà que el següent pas siga molt més precís.

3.1.3. Pas 3: Codificant cares

En aquest punt, es trobem en el nucli del problema. L'enfocament més senzill per fer front al reconeixement de les persones (qui és qui) és comparar directament la cara desconeguda que trobem al pas 2 amb totes les fotos que tenim de persones que ja s'han etiquetat. Quan trobem una cara etiquetada anteriorment que es veu molt similar a la nostra cara desconeguda, ha de ser la mateixa persona.

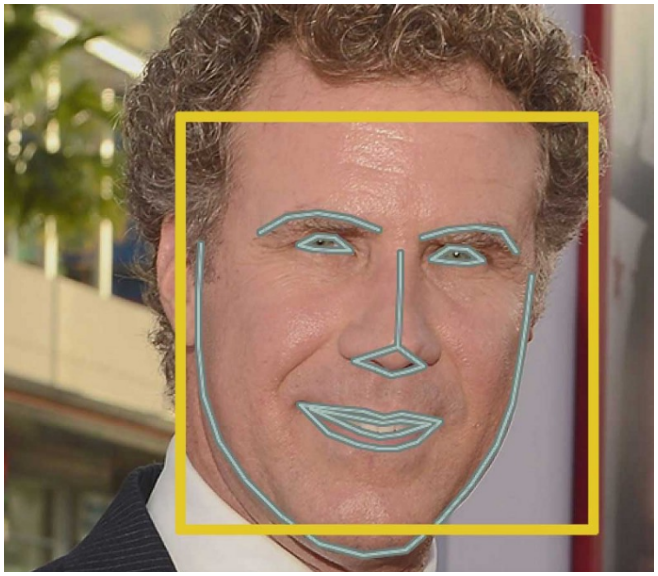

Figura 3.10: Punts de referència en la imatge de mostra

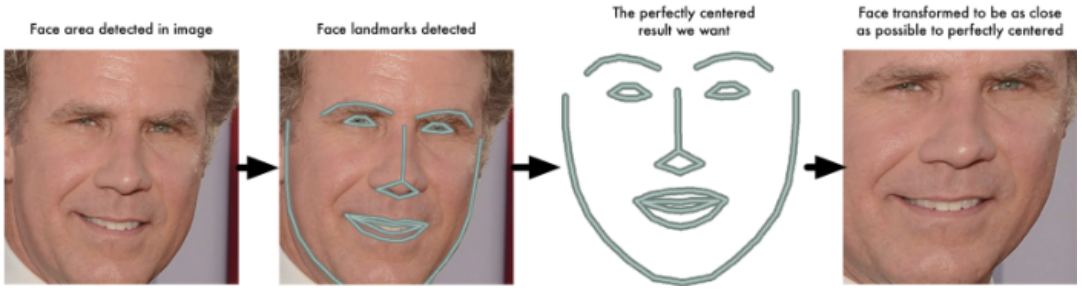

Figura 3.11: Transformacions aplicades

Realment hi ha un gran problema amb aquest enfocament. Un lloc com *Facebook* amb milers de milions d'usuaris i un bilió de fotos no poden fer el pas a través de totes les cares etiquetades prèviament per comparar-les amb cada foto que s'haja carregat recentment. Això tardaria massa temps. Han de ser capaços de reconèixer cares en mil·lisegons, no en hores.

El que necessitem és una forma d'extraure unes quantes mesures bàsiques de cada cara. A continuació, podríem mesurar la nostra cara desconeguda de la mateixa manera i trobar la cara coneguda amb els mesuraments més propers. Per exemple, podem mesurar la mida de cada oïda, l'espaiat entre els ulls, la longitud del nas, etc.

Resulta que les mesures que ens semblen òbviament humanes (com el color de l'ull) no tenen sentit en una computadora que busca píxels individuals en una imatge. Els investigadors han descobert que l'enfocament més precís és permetre que l'ordinador endevine les mesures. L'aprenentatge profund fa un treball millor que els humans per esbrinar quines parts d'una cara són importants per mesurar.

La solució és formar una Xarxa Neural Convolucional Profunda. Per tant, l'entrenarem per generar 128 mesures per a cada rostre.

El procés d'entrenament funciona mirant 3 imatges a la vegada:

- Una imatge de cara d'entrenament d'una persona coneguda
- Una altra foto de la mateixa persona coneguda
- Una foto d'una persona totalment diferent

A continuació, l'algorisme analitza les mesures que s'està generant actualment per a cadascuna de les tres imatges. A continuació, modifica la xarxa neuronal lleugerament, de manera que s'assegure que les mesures que genera per a #1 i #2 estiguen lleugerament més a prop, assegurant-se que les mesures per als #2 i #3 estan una miqueta més separades:

Després de repetir aquest pas milions de vegades per a milions d'imatges de milers de persones diferents, la xarxa neuronal aprèn a generar de manera fiable 128 mesures per a cada persona. Qualsevol de les imatges diferents de la mateixa persona hauria de donar aproximadament les mateixes mesures.

Tot seguit, una vegada que s'ha entrenat la xarxa, es poden generar mesures per a qualsevol rostre, fins i tot les que mai abans havia vist. Per tant, aquest pas només s'ha de fer una vegada. Afortunadament per a nosaltres, la gent d'*OpenFace* ja ho va fer i van publicar diverses xarxes entrenades que podem utilitzar directament.

Per tant, tot el que necessitem fer és gestionar les nostres imatges facials a través de la seua xarxa preentrenada per obtenir les 128 mesures per a cada rostre. En la Figura 3.12 tenim les mesures per a la nostra imatge de prova.

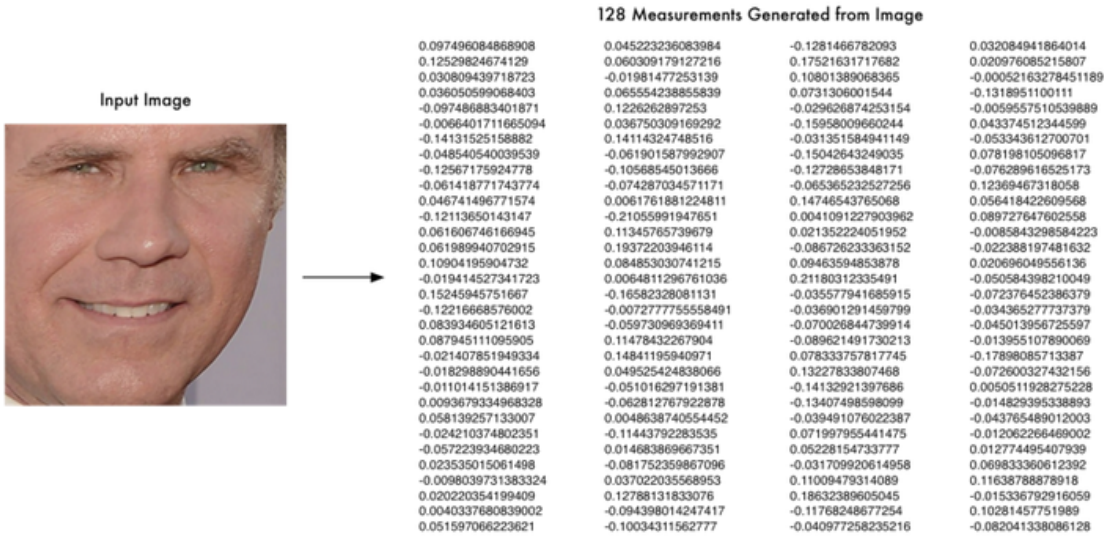

Figura 3.12: Mesures generades per a la imatge de mostra

Llavors, quines parts de la cara són representades per aquests 128 números? Resulta que no tenim cap idea. En realitat no ens importa. Tot el que ens importa és que la xarxa genera gairebé els mateixos números quan es mira dues imatges diferents de la mateixa persona.

3.1.4. Pas 4: Trobar el nom de la persona donada una codificació

Aquest últim pas és, en realitat, el pas més senzill de tot el procés. Tot el que hem de fer és trobar la persona a la nostra base de dades de persones conegudes que tinga les mesures més properes a la nostra imatge de prova.

Podeu fer-ho usant qualsevol algorisme bàsic de classificació d'aprenentatge automàtic. No es necessiten trucs d'aprenentatge profund. Utilitzarem un simple classificador SVM lineal, però molts algorismes de classificació podrien funcionar.

Tot el que hem de fer és entrenar un classificador que puga prendre les mesures d'una nova imatge de prova i li diu a quina persona coneguda és la coincidència més propera.

L'execució d'aquest classificador tarda milisegons. El resultat del classificador és el nom de la persona. Per tant, un classificador de k-veïns ens solucionarà la nostra problemàtica.

3.1.5. Pas 5: Codificar distàncies

Com ja sabem, l'elecció de característiques, en qualsevol problema de reconeixement de patrons, és una de les tasques més importants dintre del preprocés, anterior al entrenament del model. Una bona elecció de les característiques que defineixen les distintes emocions, ens permetrà obtindre un model més precís i eficient de cara a la posterior classificació de noves mostres alienes al conjunt de dades d'entrenament.

Per a la classificació de les expressions facials, ens basarem en les distàncies entre diferents punts característics, obtinguts del rostre facial. Per obtindre aquests punts, utilitzarem dos mètodes de la llibreria *Dlib*[1], denominats detector i predictor. Detector, s'encarregarà de trobar cares de persones en una imatge i predictor d'estimar la seua postura. Aquesta última pren la forma de 68 punts de referència, com ja hem dit anteriorment. Aquests són punts característics sobre la cara; com les cantonades de la boca, al llarg de les celles, en els ulls, i així successivament com podem observar en la Figura 3.13

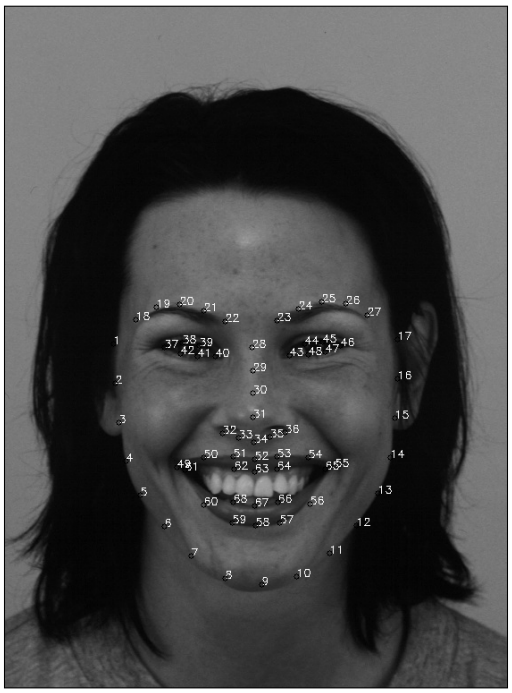

Figura 3.13: Punts obtinguts amb els mètodes *detector* i *predictor* a partir de la imatge *AF33HAS* pertanyent al conjunt de dades *KDEF*

Aquest detector de cares està desenvolupat usant el clàssic Histograma de Gradients Orientats (*HGO*) com a característiques combinades amb un classificador lineal, una piràmide d'imatge, i l'esquema de detecció de finestra lliscant. L'estimador de postura fou creat a partir de la implementació de l'article de (Kazemi i Sullivan 2014b) i fou entrenada amb el conjunt de dades de punts de referència de cares denominat *iBUG 300-W*.[2]

[1] Consulta http://dlib.net/ per a més informació
[2] Consulta http://ibug.doc.ic.ac.uk/resources/300-W per a més detalls de com obtindre *300-W*

Distàncies entre punts

Utilitzant la idea proposada en (Sohail i Bhattacharya 2008) de calcular distàncies usant només uns pocs punts, es decideix estudiar, a partir de totes les possibles distàncies entre els distints punts, quines són les més significatives, per a proporcionar a l'entrenador, un conjunt de mostres amb menys soroll. A més a més, les dades o característiques que obtindrem, les normalitzarem utilitzant la funció que es mostra a continuació:

$$z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)} \tag{3.1}$$

per a $x = (x_1, ..., x_n)$ i z_i serà la i-èsima mostra normalitzada. Amb açò, reduirem el sobreajustament, millorarem la precisió i per últim, disminuirem el temps d'entrenament del nostre classificador. La distància entre dos punts, on cada punt és una coordenada (x,y), que s'utilitzarà per a quantificar la longitud entre els dos punts, és la distància euclídia i es mostra en l'exemple següent:

$$d_E(P_1, P_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{3.2}$$

Per a determinar quines són les distàncies o característiques més significatives per a discriminar entre les diferents expressions facials, usarem el *Recursive Feature Elimination* (*RFE*) i *Feature Importance* (*FE*). Aquest últim, està basat amb arbres aleatoris. Per a fer aquest anàlisi, necessitarem construir un vector amb totes les possibles distàncies (en avant, característiques). En total, es calcularan 2278 característiques (68punts*67punts/2) per a cada cara. Una vegada calculades per a tot el conjunt de dades, li'l proporcionarem als dos algorismes nombrats anteriorment. Amb *RFE* obtenim una classificació per ordre d'importància de cada característica. Per altra banda, amb el *FE*, s'obté un valor relatiu d'importància per a cada característica. Analitzant la gràfica obtinguda amb *FE*, ens n'adonem que tan sols un 16 % de les característiques calculades (364 distàncies), tal i com es pot veure a la Figura 3.16, ens aporten la major informació a l'hora de distingir entre les diferents expressions facials. Es per açò que, es decideix, per als dos algorismes d'anàlisi, seleccionar el percentatge de característiques anteriorment descrit. A més a més, es proporcionen dues imatges (figures 3.14 i 3.15), on es pot visualitzar quines són les característiques o distàncies seleccionades per ambdós algorismes.

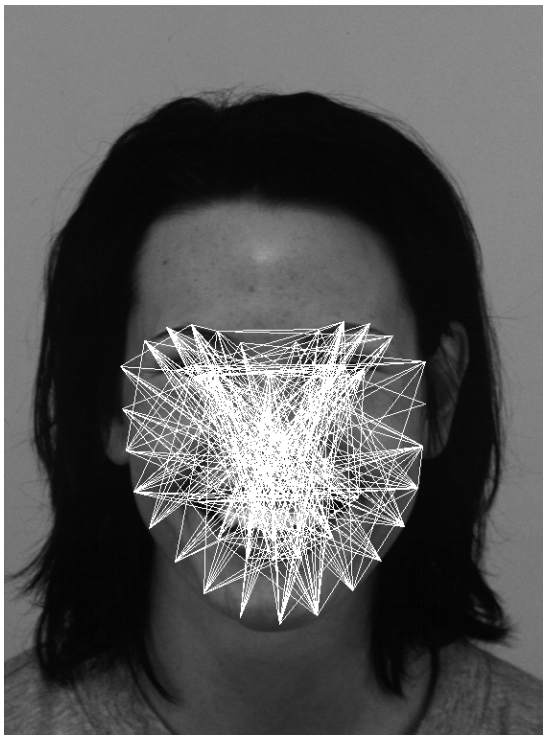

Figura 3.14: Distàncies obtingudes amb *RFE* de la imatge *AF33HAS* de la col·lecció *KDEF*

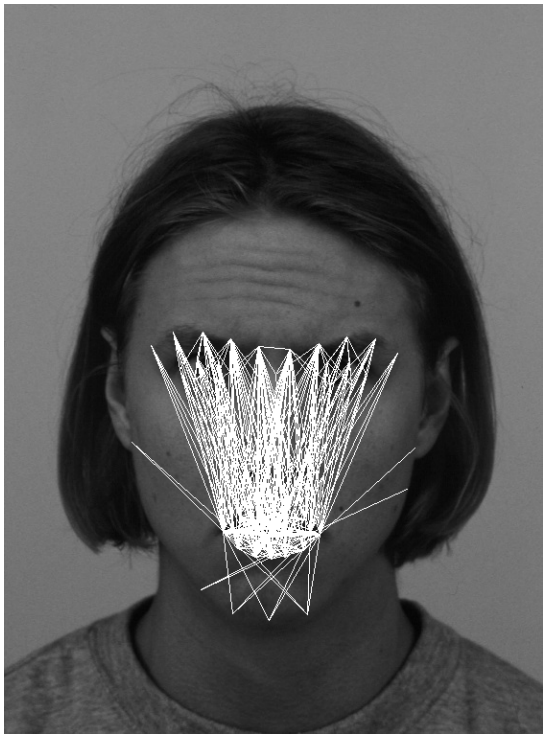

Figura 3.15: Distàncies obtingudes amb *FE* de la imatge *AM20AFS* de la col·lecció *KDEF*

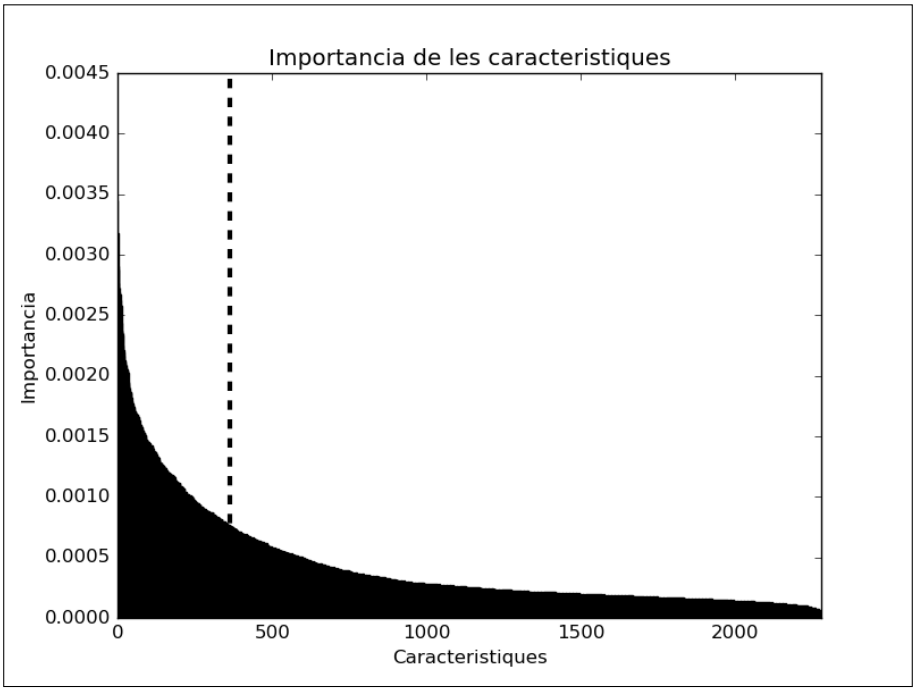

Figura 3.16: Gràfica de la importància relativa de les característiques utilitzant *FE*

3.1.6. Pas 6: Predicció d'emoció

Utilitzant les distàncies o característiques citades anteriorment, entrenarem diversos algorismes d'aprenentatge automàtic per tal de predir la emoció representada en una imatge.

3.2 Base de dades d'expressions facials

Explicat el problema, se'ns fa necessària l'obtenció d'un conjunt de dades d'imatges de cares amb les diferents expressions facials, correctament etiquetades, per tal d'estudiar la naturalesa de les característiques que podem obtindre d'aquestes. Es per això que rere una minuciosa recerca, s'obtenen les següents bases de dades:

- ***JAFFE***. La *JApanese Female Facial Expressions* (1998) és una base de dades que conté 217 fotos de 10 dones japoneses posant per expressar felicitat, tristesa, por, ira, sorpresa, disgust i neutralitat.[3] Aquesta està formada per 30 imatges expressant ira, 29 de disgust, 32 de por, 31 de felicitat, 30 de neutralitat, 31 de tristesa i 30 de sorpresa. Usada com a conjunt de dades per a la realització de l'article (Dailey et al. 2010)

- ***CAFE***. La *CAlifornia Facial Expressions* és un conjunt de dades creat a la Universitat de Califòrnia, San Diego.[4] *CAFE* està composta per 135 fotografies de models provinents del sud de Califòrnia que representen la felicitat, la tristesa, la por, la ira, sorpresa i disgust, així com expressions neutrals. Aquesta col·lecció consta de 13 imatges de ira, 21 de disgust, 13 de por, 40 de felicitat, 13 de tristesa, 17 de sorpresa i 18 d'expressions neutrals. Usada també, al mateix article que l'anterior base de dades.

- ***KDEF***. La *Karolinska Directed Emotional Faces* és una base de dades creada al Institut Karolinska, Estocolm, Suècia.[5] Conté les sis emocions anteriorment descrites amés d'imatges expressant neutralitat, en total la col·lecció consta de 980 fotografies. Aquesta ha sigut empleada en diferents articles, com per exemple (Calvo i Lundqvist 2008). La *KDEF* està composta de 140 fotografies per a cada expressió facial. A la Figura 3.17 podem trobar exemples de les imatges que formen part d'aquest conjunt de dades.

- **CFE**. Creada a partir d'imatges pròpies, obtingudes amb la càmera del robot *NAO* i mitjançant una plataforma web[6] creada per a recaptar imatges de tothom el que volguera participar amb el projecte. Aquesta pàgina web fou desenvolupada mitjançant *Nodejs*, que permet l'execució de programes escrits amb *Python*. Així, la integració del reconeixedor d'emocions, es podria implantar d'una forma molt senzilla, encara que aquest, no forma part dels objectius d'aquest projecte. La *CFE* conté 168 fotografies de l'expressió de disgust, 168 de por, 197 de sorpresa, 164 de tristesa, 151 de por, 169 de felicitat i 162 de neutralitat.

En la següent secció, en serà necessari seleccionar una col·lecció de les anteriors, per tal d'avaluar i identificar les característiques més significatives que defineixen cada expressió facial. Així, utilitzarem una mescla de totes les col·leccions.

3.3 Conclusions

En aquest capítol hem parlat de la problemàtica del problema actual. També s'ha descrit des d'on partíem fins al treball a realitzar.

[3] Consulta http://www.kasrl.org/jaffe.html per a més detalls de com obtindre *JAFFE*
[4] Consulta http://www.cse.ucsd.edu/ gary per a més detalls de com obtindre *CAFE*
[5] Consulta http://www.emotionlab.se/resources/kdef per a més detalls de com obtindre *KDEF*
[6] Consulta http://emotion.animaldesekia.info per a visualitzar la ferramenta web

Figura 3.17: Subconjunt d'imatges que formen part de *KDEF*

CAPÍTOL 4
Arquitectura

En aquest capítol, entrarem en detall en com s'estructura la nostra proposta d'arquitectura de serveis i com interactuen entre ells. Primerament, haurem de remarcar que s'ha habilitat un contenidor *docker* per a cada servei que ofereix la nostra plataforma. Açò ens permetrà, com ja hem dit, obtindre un sistema modularitzat capaç de ser escalable, amb tolerància a errors i tolerància a fallades. En cada contenidor hi serà executat un sistema operatiu propi amb el programari i llibreries necessàries per a ser executat.

En cada apartat següent es descriuran els mòduls o unitats que formen part del sistema. Per a cada mòdul, descriurem el funcionament d'aquest, els mòduls que es comuniquen amb aquest, els paràmetres que espera rebre i la configuració *docker* que s'ha utilitzat per a construir el contenidor en concret. A continuació podem visualitzar l'arquitectura de sistema que hi proposem (Figura 4.1).

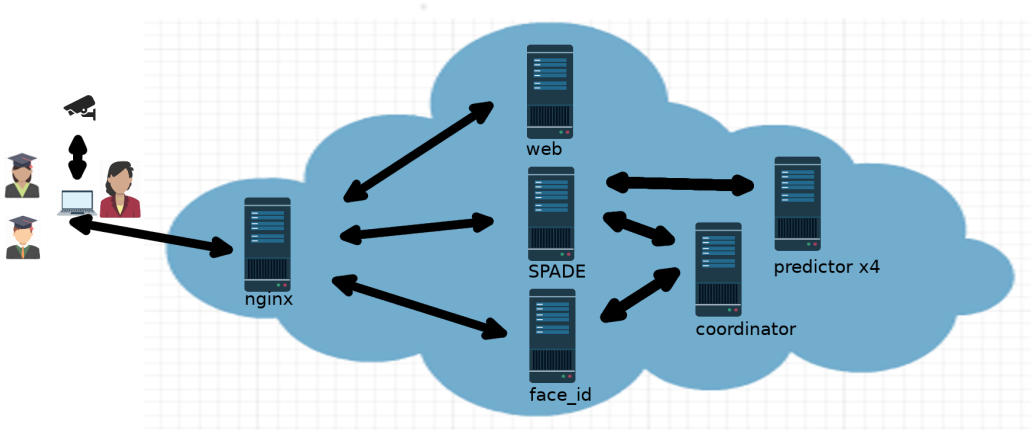

Figura 4.1: Arquitectura proposada

4.1 Contenidor `nginx`

Aquest primer contenidor té la finalitat de fer de servidor web, és a dir, rep les peticions web i les adreça cap al contenidor que siga necessari per tal de retornar la resposta esperada. Per a la nostra sort, existeix una imatge al *docker hub* anomenada `nginx:1.13` que ens proporciona un sistema operatiu lleuger, és a dir, que ens ocuparà poca memòria d'emmagatzematge. Amb aquesta imatge i utilitzant la ferramenta `docker-compose` podem modificar la configuració del `nginx` amb, tan sols, muntar un volum on es compartisca el fitxer de configuració que tinguem en local. D'aquesta manera li indicarem que es munte en `/etc/nginx/conf.d/default.conf` dintre del contenidor.

A més a més, aquest contenidor serà enllaçat mitjançant el paràmetre `links` amb el contenidor `SPADE` (Secció 4.2), al Servei de Reconeixement d'Identitat (Secció 4.6) i al Servei d'Interacció amb la Plataforma (Secció 4.7). Com a annexe s'adjunta el fitxer de configuració per a visualitzar els ports i rutes internes usades per cada contenidor i permeses per el nostre servidor `nginx` (Apèndix A).

4.2 Contenidor `SPADE`

En aquest cas, per a aquest contenidor no disposem de cap imatge al *docker hub* que ens puga suplir els requeriments de programari necessaris per a fer córrer la plataforma per a agents intel·ligents `SPADE`. Per a això, necessitarem crear-la nosaltres mateixos a partir d'una imatge `docker` anomenada `ubuntu:16.04`. Posteriorment, instal·larem `python`, `pip` i l'`SPADE`. Finalment desarem l'estat de l'imatge i la pujarem al servei `docker hub`. Esta imatge ens servirà de contenidor per a aquest mòdul. Cal recordar que este mòdul serà el nucli de la missatgeria interna, ja que és el que gestiona la comunicació entre els diversos agents intel·ligents.

4.3 Agent coordinador

En aquest apartat descriurem l'agent coordinador, així com el seu funcionament i els paràmetres d'entrada i d'eixida.

4.3.1. Descripció i funcionament

L'agent coordinador s'encarregarà de gestionar les peticions de classificació i identificació personal que hi vagen arribant mitjançant la plataforma de sistemes multiagent *SPADE*. Aquest agent rebrà la imatge proporcionada per l'agent (o agents) capturador(s) d'imatges (Secció 4.5). Aquesta imatge serà processada per tal d'extraure els punts característics proposats en Verdú (2016). Aquest punts característics seran enviats a l'agent classificador (Secció 4.4) per tal d'obtindre l'emoció que representa la persona fotografiada. Aquests punts característics també seran enviats al Servei d'Identificació Personal (Secció 4.6) per tal d'obtindre una identificació única d'aquesta persona. Una vegada obtinguda la identificació de la persona i l'emoció que representa, aquesta informació queda desada al sistema i es tornada com a resposta a l'agent iniciador, en aquest cas, l'Agent Capturador d'imatges.

4.3.2. Paràmetres

Donat que aquest agent forma part del sistema multiagent *SPADE*, no cal informar d'un punt d'entrada o *endpoint*. Sols cal indicar els paràmetres del missatge que rebrà i el procés que s'executarà després de rebre'l.

Així, podem identificar els tres tipus de missatges que en rebrà l'Agent Coordinador:

- Missatge de petició de reconeixement d'imatge provinent de l'Agent Capturador d'imatges: Aquest missatge portarà com a performativa `inform`, la odontologia serà `predict` i el contingut del missatge serà la pròpia imatge en binari codificada amb `base64`
- Missatge d'imatge classificada provinent de l'Agent Classificador: La performativa d'aquest missatge serà `inform`, la odontologia `result` i el contingut del missatge

deurà ser un missatge `JSON` amb els camps corresponents a la identitat de la persona reconeguda a través de la imatge així com l'emoció que n'és representada

- Missatge de *login* provinent de l'Agent Classificador: La performativa establerta per a aquest missatge serà `inform`, la odontologia `login` i el contingut del missatge deurà ser `tipus de client` i `número de dimensions` del color, blanc i negre (1) o rgb (3).

4.3.3. Contenidor `docker`

Partirem de la imatge anterior, creada en l'apartat 4.2, on ens farà falta instal·lar les llibreries necessàries perquè són dependències del Agent Contenidor. En aquest cas, les llibreries serien la `dlib`, `opencv`, `pillow`, `sklearn`, `scipy`, `matplotlib`, `requests` i `face_recognition`, a més a més de diverses depèndencies de la llibreria `dlib`.

Una vegada creada la imatge, l'utilitzarem per desplegar un contenidor que arranque un script `python` que es comunique amb el Contenidor `SPADE`. Per açò, serà necessari enllaçar-lo al Contenidor `SPADE` amb el paràmetre de `docker` anomenat `link`. També, haurà de ser enllaçat amb el contenidor del `Sistema de Reconeixement d'Identitat`(Secció 4.6) Aquesta, serà la base per a construir la nostra imatge que es farà servir per desplegar un contenidor i llançar el programa que executa l'Agent Coordinador.

4.4 Agent classificador d'emocions

En aquesta secció tractarem d'explicar el mòdul de l'Agent classificador d'emocions, així com els paràmetres d'entrada i d'eixida d'aquest.

4.4.1. Descripció i funcionament

L'agent ens permetrà reconèixer l'emoció representada en una imatge facial mitjançant les distàncies calculades prèviament. Aquest contenidor, que conté un agent `SPADE`, té la peculiaritat de que pot ser llançat amb diferents instàncies d'aquest per tant d'incorporar tants classificadors amb distints models *ML* entrenats com vulga'm per tal de realitzar la classificació.

4.4.2. Paràmetres

D'aquest agent, com l'anterior, no s'informa d'un punt d'entrada o *endpoint*. Sols cal indicar els paràmetres dels missatges que en rebrà i el procés que s'executarà després de rebre'ls.

Així, podem identificar dos tipus de missatges que gestionarà l'Agent Coordinador:

- Missatge provinent de l'Agent Coordinador: Aquest missatge serà identificat amb la performativa `inform`, amb odontologia `emotion` i el contingut d'aquest serà un vector de distàncies i el nom de la persona a qui pertanyen. Una vegada, rebut aquest missatge, es classificarà el vector mostra i el resultat serà enviat de tornada a l'Agent Coordinador. El missatge serà codificat en format *JSON* on un dels paràmetres en serà l'`ID` de la persona identificada i l'altre paràmetre, el de l'emoció representada.

- Missatge de *login* que li serà enviat a l'Agent Coordinador: Aquest vindrà identificat amb la performativa `inform`, l'odontologia `login` i el contingut en serà `el tipus de client`

4.4.3. Contenidor `Docker`

Partirem de la imatge anterior, creada en l'apartat 4.2, on ens farà falta instal·lar les llibreries necessàries perquè són dependències de l'Agent Classificador. En aquest cas, sols necessitaríem la llibreria `opencv` per a carregar els models d'`ML` entrenats.

Una vegada creada la imatge, l'utilitzarem per desplegar un contenidor que arranque un script `python` que es comunique amb el Contenidor `SPADE`. Per açò, serà necessari enllaçar-lo al Contenidor `SPADE` amb el paràmetre de `docker` anomenat `link`. Aquesta, serà la base per a construir la nostra imatge que es farà servir per desplegar un contenidor dedicat i llançar el programa que executa l'Agent Classificador. D'aquesta imatge és poden desplegar diverses instàncies on el model empleat per a classificar siga diferent als demés. Els agents que s'executen dintre de cada contenidor es registraran en el sistema automàticament.

4.5 Agent capturador d'imatges

4.5.1. Descripció i funcionament

Aquest agent s'encarregarà de fer les peticions internament amb el sistema multiagent per tal d'obtindre l'emoció i la identitat de certa persona en una imatge donada.

4.5.2. Paràmetres

D'aquest agent, com l'anterior, no s'informa d'un punt d'entrada o *endpoint*. Sols cal indicar els paràmetres dels missatges que en rebrà i el procés que s'executarà després de rebre'ls.

Així, podem identificar dos tipus de missatges que gestionarà l'Agent Coordinador:

- Missatge enviat a l'Agent Coordinador: La performativa s'establirà en `inform`, la odontologia `predict` i el contingut del missatge serà la imatge capturada, en format binari, codificada en *base64*.

- Missatge rebut per l'Agent Coordinador: La performativa serà `inform`, l'odontologia `result` i el contingut vindrà donat per el resultat de les persones reconegudes en la imatge i les seues emocions. L'eixida de la resposta serà per eixida estàndard en el sistema Linux. A més a més, l'agent morirà una vegada haja mostrat el resultat.

4.5.3. Contenidor `Docker`

Per a aquest agent no en serà necessari crear un contenidor ja que, aquest, serà executat pel Servei d'Interacció (Secció 4.7). Malgrat això, també és possible executar aquest mòdul dintre d'un contenidor i connectar-li una webcam per a capturar imatges i anar processant-les pel sistema. Açò, ens obri les portes a portar aquest codi a qualsevol plataforma i poder ser usable per qualsevol sistema sense massa complicació.

4.6 Servei de reconeixement d'identitat

4.6.1. Descripció i funcionament

Aquest servei web estarà encarregat de gestionar l'històric de les emocions de les persones així com el model que ens permetrà obtindre la identitat d'una persona donada.

Aquest servei, estarà desenvolupat mitjançant una *API REST* en *Python* usant les llibreries *Flask* per a gestionar els *endpoints* i la llibreria *face_recognition* que ens permet obtindre els *face encodings* que identifiquen inequívocament a una persona.

El servei mantindrà una cua d'N persones junt amb el seu *face encoding* i l'`ID` d'aquesta. A més a més, proveirà de certs *endpoints* per a interactuar amb el servei. El comportament dels *endpoints* vindrà descrit en la subsecció següent.

4.6.2. *Enpoints*

- **/face/checkIdentity**: Aquesta ruta rebrà per mètode *POST* la imatge de la qual es pretén identificar a les persones fotografiades i el nom 'temporal' donat. L'anomenem temporal perquè, en cas de no ser identificat, s'etiquetaria i es desaria al sistema amb el nom proporcionat, l'`ID`. Tornarà com a resposta en format *JSON* tres atributs: un que indique si ha sigut detectada en la imatge al menys una cara, l'`ID` de les persones reconegudes i un altre paràmetre que indique si s'han trobat cares noves.

- **/face/save**: Aquesta ruta, cridada amb una petició *GET*, desarà l'estat actual de la plataforma així com els models d'identificació (*face encodings*), els `ID`'s de les persones així com les emocions d'aquestes.

- **/face/clear**: Aquesta ruta, cridada amb una petició `GET`, deixarà l'estat actual de la plataforma a valors de fàbrica, valors inicials.

- **/face/emotion**: Aquesta ruta, cridada amb una petició `POST`, modificarà l'emoció actual d'una persona, en la cua, mitjançant un missatge *JSON* amb paràmetres: l'`ID` de la persona i la emoció a assignar-li.

- **/face/group-model**: Aquesta ruta, consultada a través del mètode `GET` retornarà, amb format `JSON`, el nombre de cares que hi han actualment registrades al sistema, les emocions representades en format `PAD` i el valor semàntic de les emocions d'aquests punts.

4.6.3. Docker

Com s'ha comentat anteriorment, aquest servei necessitarà de cert programari per tal de gestionar i processar les peticions rebudes. És per això que partirem d'una imatge que duga instal·lada l'interpret del llenguatge `python`, l'instal·lador de paquets `pip` i les llibreries `Flask`, `face_encoding`, `dlib` i `requests`.

4.7 Servei d'interacció amb la plataforma

En aquest apartat tractarem d'explicar les diferents parts que formen el servei d'interacció amb la plataforma, així com els punts d'entrada per a interactuar-hi.

4.7.1. Descripció i funcionament

Aquesta part de la plataforma és la més interessant per a l'usuari ja que li permet interactuar amb la plataforma mitjançant un lloc web. Aquesta plana web consta de cinc parts diferenciades per tal de realitzar les tasques pertinents en la plataforma. A continuació detallarem els punts d'entrada i els distints serveis que s'hi ofereixen.

4.7.2. *Endpoints*

Tot seguit, descriurem els diversos punts d'entrada de la nostra plataforma web.

Dashboard

Aquesta finestra és la que sens mostrarà per defecte, la primera finestra que trobarem al sistema. Aquesta consta d'un menú horitzontal on podrem navegar de forma ràpida a la finestra que hi desitgem en tan sols un clic. A continuació, hi trobarem quatre caixes des d'on podrem accedir a cadascuna de les quatre finestres anteriorment enunciades. Finalment, podem trobar una imatge de la plana web a la Figura 4.2.

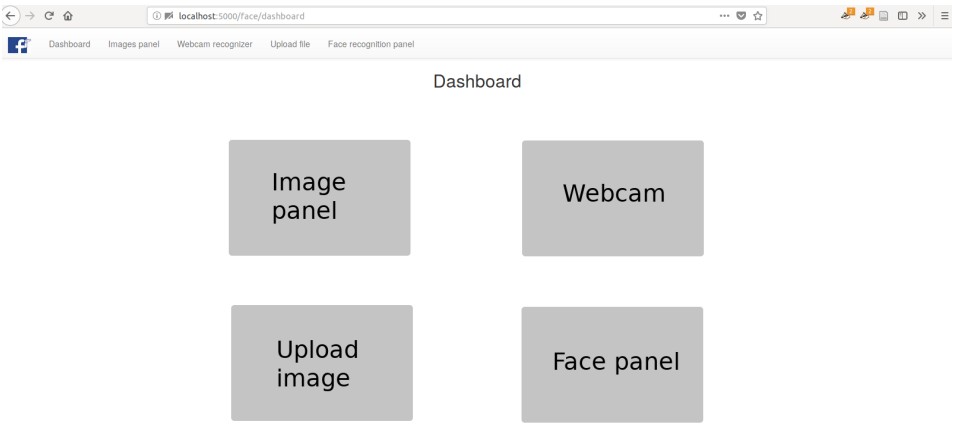

Figura 4.2: Plana web: Dashboard

Pujada d'imatges

La plana web ens permetrà pujar una imatge mitjançant el navegador web amb tan sols arrossegant-la al sobre, utilitzant una *dropzone* o bé utilitzant el gestor de directoris del sistema operatiu. A més a més, amb aquest panell permetrem a l'usuari la possibilitat de pujar una fotografia o imatge per tal d'extraure les possibles cares que hi puguen aparèixer, així com la identitat d'aquestes i les emocions associades.

Una vegada enviat el fitxer, veurem el resultat al sobre de la fotografia. Així doncs, podrem identificar el nom de la persona i l'emoció que representa. A la Figura 4.3 és mostra una imatge de la plana web que hem descrit.

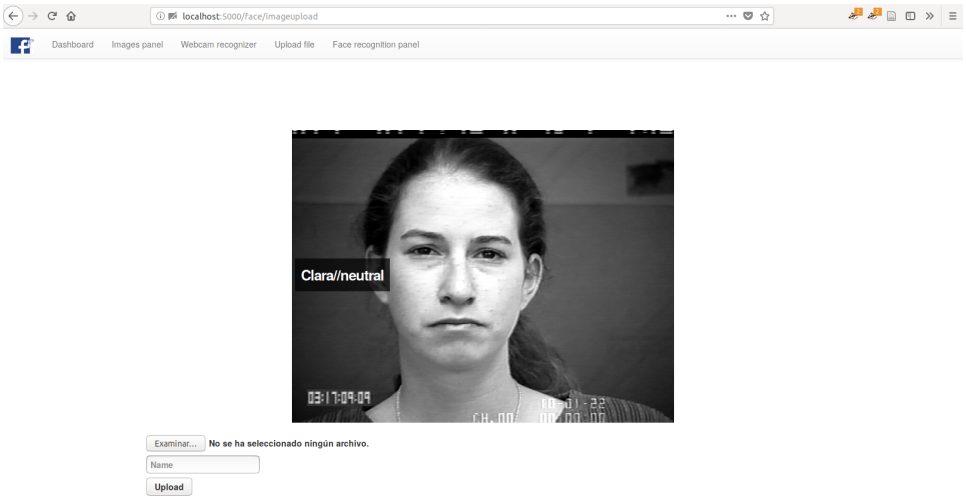

Figura 4.3: Plana web: pujada d'imatges

Pujada d'imatges mitjançant la *webcam*

La finestra que descriurem a continuació és molt pareguda a l'anterior, exceptuant que, en aquest cas, la imatge que s'envia prové directament de la càmera web. A la Figura 4.4 podem visualitzar aquesta plana web.

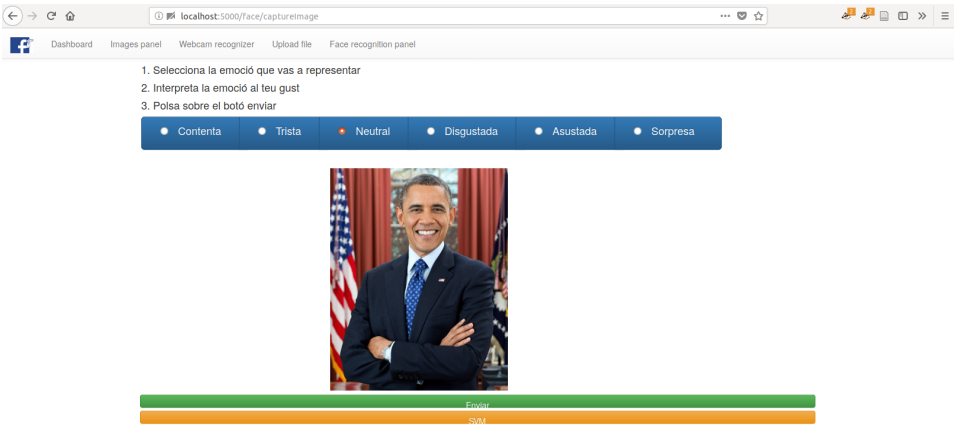

Figura 4.4: Plana web: pujada d'imatges mitjantçant la càmera web

Com podem observar, al principi se'ns informa d'uns passos que hem de seguir abans de prémer el botó d'enviar. A més a més, es pot 'etiquetar', de tal forma que, la imatge, serà desada en un directori diferent al de les demés emocions, per tal de poder revisar-les amb posterioritat. Una vegada enviada la fotografia, ens apareixerà el resultat obtingut en un *banner* superior flotant.

En aquest cas, el *backend* realitzat amb *nodejs* executarà un *script* de *python*, en concret l'Agent Capturador d'imatges(Secció 4.5). A l'agent li se supliran els paràmetres d'on pot trobar el fitxer de la imatge així com el nom de la persona que diu ser.

Panell del revisor

Al igual que les anteriors finestres, aquesta presenta la mateixa aparença distributiva. A la Figura 4.5 podem adonar-nos que, aquesta finestra, conté un element on poder visualitzar la imatge, així com uns `radio buttons` on poder etiquetar-la indicant-hi l'emoció representada. Aquesta imatge serà marcada com a etiquetada i ja no hi tornarà aparèixer. Aquestes imatges que hem etiquetat podran ser utilitzades posteriorment per reentrenar els models de `ML` i així millorar els resultats de predicció. A més a més, totes les fotografies que siguen processades pel sistema seran incorporades en aquest sistema de revisió.

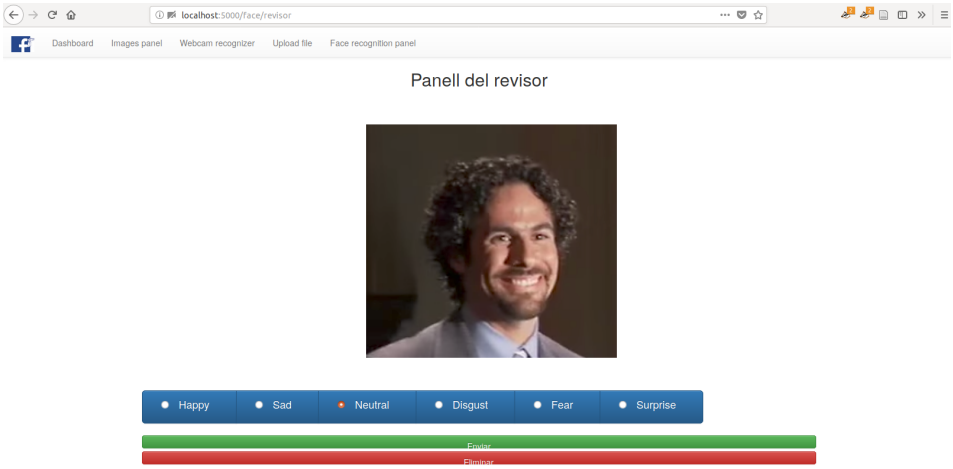

Figura 4.5: Plana web: panell del revisor

Panell d'administració

Aquesta finestra ens permetrà visualitzar l'estat actual del nostre model d'emocions grupals, és a dir, podrem observar els noms o identitats de les persones registrades així com les emocions que hi han presentat. A més a més, i com podem observar a la Figura 4.6, també proveirem a la plana web d'un visualitzador de tres dimensions, on podrem observar els valors `PAD` de les persones que hi hem registrat.

4.7.3. `Docker`

Per a aquest contenidor `docker` farem servir una imatge del `docker hub` anomenada `node`. Aquesta imatge porta com a sistema operatiu un `Debian` amb el necessari per desplegar aplicacions web basades en `nodejs`. A més a més, li instal·larem la llibreria `opencv` per a fer servir algunes de les seues utilitats que ens ofereix per al processament d'imatges.

4.8 Conclusions

En aquest capítol hem descrit l'arquitectura del sistema proposat així com també hem explicat com s'ha dissenyat cada mòdul, els paràmetres que rep i/o envia i la composició del programari de les imatges `Docker` que conformen cada mòdul. Com podem observar, hem dissenyat un sistema multiplataforma, preparat per a ser escalable i tolerant a fallades en grans sistemes.

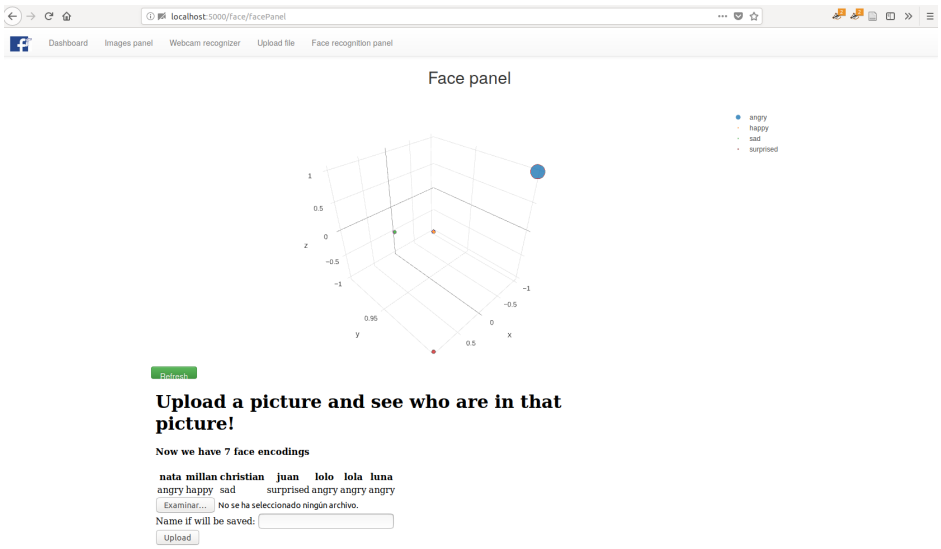

Figura 4.6: Plana web: panell de l'administrador

CAPÍTOL 5
Experimentació

5.1 Reconeixement d'identitat

Per realitzar l'experimentació sobre aquesta subtasca, abans de tot, haurem de definir les distintes particions d'entrenament i test. Per això, deixarem un 80% de mostres per a la partició d'entrenament i la resta per al test. A més, definirem el nombre d'iteracions que es situarà en 100. Per tant, en cada iteració es calcularan les 128 mesures, per a cada mostra, que seran generades per la Xarxa Neural Convolucional Profunda, ja entrenada amb mostres no vistes. A continuació, una vegada calculades, classificarem el vector obtingut usant el model de k-veïns de persones conegudes.

Per tant, si disposem de 4600 mostres i amb 100 iteracions obtenim una mitja de 40 errors, és a dir, 40 imatges de les quals no s'ha pogut identificar la persona retratada o que la persona identificada no es correspon amb la que hauria de ser. Així doncs, obtenim una taxa d'encerts del 99.13% que s'apropa bastant a l'obtinguda en *Labeled Faces in the Wild* (2018), que es del 99.38%.

5.2 Reconeixement d'emocions

Partint de l'aproximació feta en (Verdú 2016), s'han entrenat quatre models distints, dos *SVM* i dos *MLP*. Cada model serà servit per una instància *docker*. Per tant, per poder avaluar aquest nou model, format pel conjunt dels quatre models, plantejarem una sèrie d'experimentacions que es basaran en realitzar cent iteracions i entrenar quatre models distints, que seran usats pels agents classificadors. Abans de realitzar l'entrenament en cada iteració, realitzarem diverses particions en les mostres d'entrenament. Primerament, el conjunt de mostres totals serà dividit en dues particions, concretament una del 80% i la restant del 20%.

Reservarem aquesta última com a conjunt de test. La partició més gran, la dividirem en dues particions, el 56% de les mostres inicials i el 56% de les mostres finals. Aquestes particions estaran parcialment interseccionades, és a dir, les particions compartiran un 40% de les mostres d'entrenament. En cada iteració, s'entrenaran els dos models (*SVM* i *MLP*). També s'obtindran resultats de predicció per obtindre mètriques i comparar els models.

Per a realitzar aquestes particions, seleccionarem el 70% (56% del total de mostres) de les primeres mostres del conjunt més gran per al primer conjunt d'entrenament. Per a la segona partició utilitzarem el 70%, de les últimes mostres provinents del conjunt gran. En la Figura 5.1 es mostra gràficament com es divideixen les distintes particions. Per

tant, obtindrem tres particions; la partició del 20%, reservada per a la classificació, i dos conjunts que ens serviran per a l'entrenament.

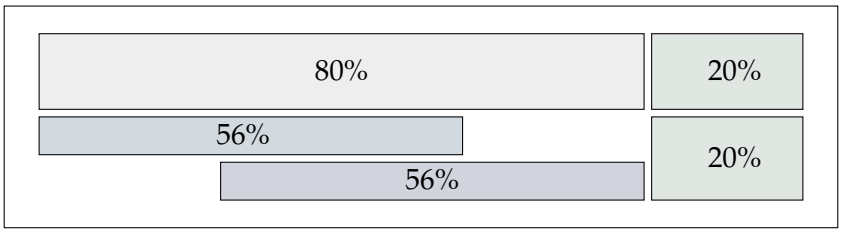

Figura 5.1: Representació gràfica del format de les particions utilitzat per a l'experimentació.

Amb el primer dels dos conjunts d'entrenament anteriorment formalitzats, entrenarem un parell de models. En concret, un dels dos conjunts serà usat en l'entrenament del model amb l'algorisme *SVM* i l'altre amb *MLP*. Amb el segon conjunt restant realitzarem el mateix procediment de tal forma que obtindrem dos classificadors més que faran un total de quatre.

Els resultats dels quatre models es mostren en les taules comparatives (Taula 5.1 i Taula 5.2). Cada taula correspon als resultats d'un model, *SVM* o *MLP*, comparats amb el model format pel conjunt de classificadors (etiqueta *MIX*).

Finalment, tan sols ens queda comparar els resultats obtinguts pel model que acabem de construir amb els models que el conformen. Per això, analitzarem els diferents valors de *TPR*, *SPC*, *PPV* i *NPV*.

Si ens centrem en la **taxa de verdaders positius** (*TPR*), ens adonem que, en comparació als resultats obtinguts pels submodels entrenats, la millora es produeix perquè el model *MIX* sempre maximitza la mètrica obtinguda per algun submodel. El model *MIX*, el que hem plantejat en aquest capítol, maximitza el valor dels resultats dels seus submodels.

Per altra banda, els resultats d'**especificitat** (SPC) i del **valor predictiu negatiu** (NPV) del model *MIX* no varien significativament dels resultats dels seus submodels.

Finalment, si comparem el **valor predictiu positiu** (*PPV*) dels resultats obtinguts pel model *MIX* amb els dels seus submodels, la diferència no és significativa, però si que podem afirmar que el *PPV* del model *MIX* s'acosta al valor més alt obtingut pels submodels.

5.3 Conclusions

Com a resum d'aquest capítol, hem obtingut les mètriques per a la subtasca de identificació de persones amb un resultat similiar al obtingut a (*Labeled Faces in the Wild* 2018). A més, hem construït quatre models diferents per a la subtasca de reconeixement d'emocions que, independentment, emetran una predicció. Aquesta predicció serà recollida pel sistema de votacions i processada, per tal d'arribar a un consens on es trie la classe a la qual prediem que pertany.

Amb la nostra proposta, hem aconseguit millorar el *TPR* i *PPV* mitjançant el model *MIX* en comparació als submodels creats.

Per a acabar amb aquesta conclusió, farem un especial èmfasi en la millora respecte a l'escalabilitat i disponibilitat del sistema, ja que, amb aquesta implementació hem

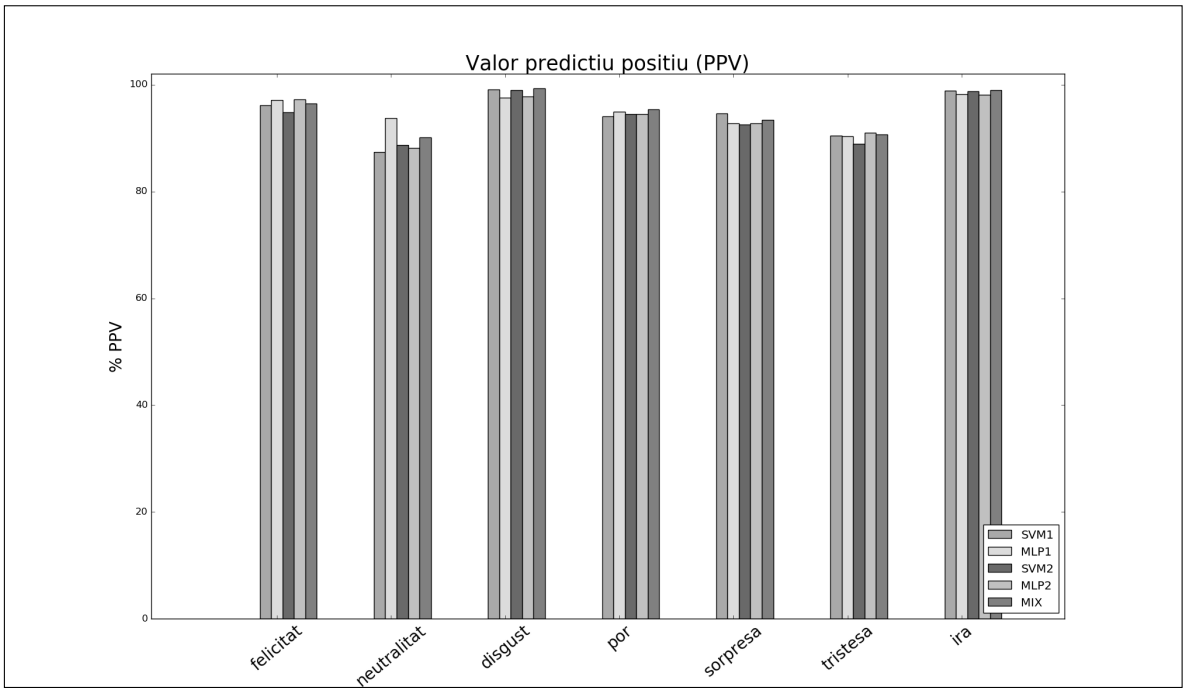

Figura 5.2: Gràfica on es comparen els valors de PPV de cada emoció i per a cada model.

aconseguit distribuir els càlculs entre els distints agents i replicar els distints agents per proporcionar-li una disponibilitat més alta, a més de proporcionar una implementació capaç de ser escalable en un moment donat.

Taula 5.1: Mètriques obtingudes amb el primer conjunt de dades d'entrenament

		felicitat	neutralitat	disgust	por	sorpresa	tristesa	ira
	svm	679	577	670	526	631	576	687
TP	mlp	697	553	680	530	642	586	683
	mix	623	611	680	554	633	606	657
	svm	28	41	18	70	43	52	2
FN	mlp	10	65	8	66	32	42	6
	mix	32	44	4	76	39	39	2
	svm	27	83	6	33	36	61	8
FP	mlp	21	37	17	28	50	63	13
	mix	23	67	5	27	45	62	7
	svm	3866	3899	3906	3971	3890	3911	3903
TN	mlp	3872	3945	3895	3976	3876	3909	3898
	mix	3922	3878	3911	3943	3883	3893	3934
	svm	96,04	93,37	97,38	88,26	93,62	91,72	99,71
TPR	mlp	98,59	89,48	98,84	88,93	95,25	93,31	99,13
	mix	95,11	93,28	99,42	87,94	94,20	93,95	99,70
	svm	99,31	97,92	99,85	99,18	99,08	98,46	99,80
SPC	mlp	99,46	99,07	99,57	99,30	98,73	98,41	99,67
	mix	99,42	98,30	99,87	99,32	98,85	98,43	99,82
	svm	96,18	87,42	99,11	94,10	94,60	90,42	98,85
PPV	mlp	97,08	93,73	97,56	94,98	92,77	90,29	98,13
	mix	96,44	90,12	99,27	95,35	93,36	90,72	98,95
	svm	99,28	98,96	99,54	98,27	98,91	98,69	99,95
NPV	mlp	99,74	98,38	99,80	98,37	99,18	98,94	99,85
	mix	99,19	98,88	99,90	98,11	99,01	99,01	99,95
	svm	0,69	2,08	0,15	0,82	0,92	1,54	0,20
FPR	mlp	0,54	0,93	0,43	0,70	1,27	1,59	0,33
	mix	0,58	1,70	0,13	0,68	1,15	1,57	0,18
	svm	3,82	12,58	0,89	5,90	5,40	9,58	1,15
FDR	mlp	2,92	6,27	2,44	5,02	7,23	9,71	1,87
	mix	3,37	8,54	1,02	4,61	5,59	7,58	1,01
	svm	3,96	6,63	2,62	11,74	6,38	8,28	0,29
FNR	mlp	1,41	10,52	1,16	11,07	4,75	6,69	0,87
	mix	2,69	6,47	1,45	9,73	4,75	6,85	0,29
	svm	98,80	97,30	99,48	97,76	98,28	97,54	99,78
ACC	mlp	99,33	97,78	99,46	97,96	98,22	97,72	99,59
	mix	99,07	97,96	99,63	98,17	98,48	98,02	99,80

Taula 5.2: Mètriques obtingudes amb el segon conjunt de dades d'entrenament

		felicitat	neutralitat	disgust	por	sorpresa	tristesa	ira
	svm	610	602	678	552	623	596	658
TP	mlp	630	590	676	555	641	586	657
	mix	623	611	680	554	633	606	657
	svm	45	53	6	78	49	49	1
FN	mlp	25	65	8	75	31	59	2
	mix	32	44	4	76	39	39	2
	svm	33	77	7	32	50	74	8
FP	mlp	18	79	15	32	50	58	13
	mix	23	67	5	27	45	62	7
	svm	3912	3868	3909	3938	3878	3881	3933
TN	mlp	3927	3866	3901	3938	3878	3897	3928
	mix	3922	3878	3911	3943	3883	3893	3934
	svm	93,13	91,91	99,12	87,62	92,71	92,40	99,85
TPR	mlp	96,18	90,08	98,83	88,10	95,39	90,85	99,70
	mix	95,11	93,28	99,42	87,94	94,20	93,95	99,70
	svm	99,16	98,05	99,82	99,19	98,73	98,13	99,80
SPC	mlp	99,54	98,00	99,62	99,19	98,73	98,53	99,67
	mix	99,42	98,30	99,87	99,32	98,85	98,43	99,82
	svm	94,87	88,66	98,98	94,52	92,57	88,96	98,80
PPV	mlp	97,22	88,19	97,83	94,55	92,76	90,99	98,06
	mix	96,44	90,12	99,27	95,35	93,36	90,72	98,95
	svm	98,86	98,65	99,85	98,06	98,75	98,75	99,97
NPV	mlp	99,37	98,35	99,80	98,13	99,21	98,51	99,95
	mix	99,19	98,88	99,90	98,11	99,01	99,01	99,95
	svm	0,84	1,95	0,18	0,81	1,27	1,87	0,20
FPR	mlp	0,46	2,00	0,38	0,81	1,27	1,47	0,33
	mix	0,58	1,70	0,13	0,68	1,15	1,57	0,18
	svm	5,13	11,34	1,02	5,48	7,43	11,04	1,20
FDR	mlp	2,78	11,81	2,17	5,45	7,24	9,01	1,94
	mix	3,37	8,54	1,02	4,61	5,59	7,58	1,01
	svm	6,87	8,09	0,88	12,38	7,29	7,60	0,15
FNR	mlp	3,82	9,92	1,17	11,90	4,61	9,15	0,30
	mix	2,69	6,47	1,45	9,73	4,75	6,85	0,29
	svm	98,30	97,17	99,72	97,61	97,85	97,33	99,80
ACC	mlp	99,07	96,87	99,50	97,67	98,24	97,46	99,67
	mix	99,07	97,96	99,63	98,17	98,48	98,02	99,80

CAPÍTOL 6
Casos d'ús

En aquest capítol es plantegen distints casos d'ús per als quals, la nostra plataforma, en seria útil. A més a més, farem unes clarificacions i exposarem com s'adapta la nostra aplicació als distints dominis.

6.1 Educació

Podria donar-se el cas en el qual un docent, que està impartint una classe, volguera que tot el grup d'alumnes estiga en el mateix punt emocional. O, si més no, que sabera com els seus alumnes reaccionen davant les seues explicacions. Per exemple, si tenim un grup emocionalment bastant homogeneïtzat amb un estat neutral o feliç, però, hi han dos alumnes que destaquen al grup perquè no segueixen l'emoció grupal, podríem interferir-hi per tal de solucionar la situació. De fet, si d'aquests alumnes, un està trist i un altre atemorit, el sistema podria recomanar al docent que incidisca d'alguna forma sobre aquests alumnes per detectar els problemes pels quals eixa persona no està seguint els estadis del grup. Finalment, podríem obtindre una classe homogeneïtzada emocionalment i que podria respondre de forma igualitària davant d'uns mateixos estímuls.

6.2 Psicòleg

D'altra banda, aquesta plataforma podria ser utilitzada en un psicòleg davant d'un grup de persones per saber com reaccionen davant d'un mateix estímul. Per tant, aquesta eina pot servir per enregistrar l'estat emocional actual i els diferents estats pels quals transiten els membres del grup (estat individual). A més a més, també podem obtindre les transicions grupals (estat grupal).

Per això, seria interessant utilitzar aquesta plataforma perquè ens facilita la feina d'enregistrar els diferents estadis pels quals transitem les persones davant d'un mateix estímul. A més a més, no només s'analitza l'estat individual, sinó també, l'estat emocional col·lectiu. Totes aquestes anàlisis s'obtenen per a cada instant, és a dir, que es pot consultar l'històric d'estadis pels quals han transitat.

6.3 Comerç

Una de les característiques on els comerços posen més èmfasi és l'opinió dels clients o compradors de productes. Per tant, aquesta aplicació podria mesurar el grau de satisfacció a l'hora d'estar comprant en una tenda, a l'hora d'elegir un producte, les emocions

que els provoquen les tendes des de fora als clients... Totes aquestes emocions o sentiments podrien estar enregistrats per la plataforma per determinar com responen els nostres clients davant dels diferents estímuls que podríem trobar en el comerç. Aquestes dades li donarien a la empresa una retroalimentació per la qual les empreses podrien millorar diferents aspectes del comerç per fer-li l'estada més agradable possible al client.

6.4 Persones majors

A més a més, la nostra plataforma podria servir per a l'atenció a les persones majors. Les càmeres podrien estar connectades a Internet per tal d'enviar les imatges a la plataforma. Aquesta podria estar gestionada per persones capacitades per detectar anomalies i actuar en conseqüència. Podrien detectar-se mals hàbits i ajudar a les persones majors a corregir-los. A més a més, també es podrien detectar situacions de pànic i actuar urgentment.

6.5 Atenció de xiquets amb alguna discapacitat

Finalment, creiem que aquesta plataforma serviria per a la gestió d'un centre de xiquets i xiquetes amb alguna discapacitat. Les seues emocions podrien estar enregistrades i avaluar els models de comportament, de les emocions, per tal de decidir les noves metodologies de treball. Aquestes noves metodologies podran ser avaluades de nou per esbrinar el resultat obtingut i poder fer millores sobre aquest.

6.6 Conclusions

En aquest capítol hem vist diferents aplicacions o casos d'ús per als quals la nostra aplicació és apta per a funcionar. A més, hem vist tres possibles dominis i la seua implementació, així com els diferents factors que es podrien analitzar. Com hem vist, aquesta és una aplicació molt generalista i que es pot adaptar a distints àmbits completament diferents i que, en principi, no tenen res a veure.

CAPÍTOL 7
Conclusions i treball futur

En el present capítol exposarem les conclusions obtingudes rere l'estudi, la implementació i l'avaluació d'una plataforma per la predicció d'emocions socials per a la millora de la interacció humà-agent.

7.1 Conclusions

En aquest Treball Final de Màster s'ha formalitzat, avaluat i implementat una plataforma amb la implementació de microserveis, destinats a la predicció d'emocions socials. En primer lloc, hem de reconèixer les persones que apareixen en les fotografies. A més a més, hem d'obtindre l'emoció de cada persona per tal de relacionar-la. No cal obtindre el nom verdader, simplement detectar cada persona com a única en el grup. A més a més, s'han enregistrat les emocions individuals per tal de formalitzar models PAD. Aquests models, podrien facilitar el treball de docents, psicòlegs i empreses, ja que aquests models reflecteixen l'estat anímic de les persones respecte a un estímul. Per tant, si usem aquesta ferramenta i la complementem amb registre de vídeo, podem obtindre una solució per a enregistrar les emocions de les persones, relacionar-les amb un o més estímuls. Així, podríem avaluar l'estat actual i suggerir-li al professional prenga alguna decisió per tal de portar al grup a un punt concret del model PAD. Per a complir l'objectiu principal hem hagut de estudiar els diferents mètodes i tècniques que representen l'estat de l'art en aquests dominis. També, hem desenvolupat aquest projecte utilitzant les diferents àrees temàtiques del Màster, com la Intel·ligència Artificial, el Reconeixement de Formes i la Imatge Digital.

D'altra banda, s'han avaluat i implementat diferents tècniques d'aprenentatge automàtic més usades els *SVMs* i *MLPs*. També, hem usat Xarxes Neuronals Convolucionals Profundes per generar uns vectors de 128 dimensions, per tal d'entrenar un model de k-veïns que represente la base de dades de cares conegudes.

Per tal de validar la nostra aproximació, hem calculat unes mètriques mitjançant una tècnica de validació creuada aleatòria. Els resultats obtinguts certifiquen el correcte funcionament de la nostra aproximació així com les altres mètriques obtingudes, obtenint una taxa de precisió per a cada emoció superior al 98%.

Respecte al disseny, la implementació i el desplegament de la plataforma, i del projecte en general, ha suposat tot un conjunt de reptes i metes per la gran quantitat de coneixements que s'han requerit per tal de solucionar el problema.

7.2 Treball futur

Com a treball futur, es proposa realitzar diferents millores per tal d'adaptar la plataforma i el seu codi, millorable per la comunitat. A més a més, també definirem les millores concretes per tal de realitzar-les en un treball futur.

- **Etiquetar les cares dins d'un grup**. Per tal de fer-nos la feina més fàcil, desenvolupar un mètode per tal d'etiquetar el nom de les persones en una foto grupal usant el lloc web. Així, les dades en la plataforma reflectirien el nom real de les persones i no un nom aleatori.
- **Modificar el nom de les persones**. Oferir la possibilitat de modificar el nom de les persones en les fotos ja vistes, així com el nom en el seu model personal.
- **Identificar colls de botella**. Caldria realitzar un anàlisi del codi generat per tal d'identificar colls de botella. Açò, reduiria el temps de càlcul.
- **Especificació d'una *API***. Hauríem de publicar una documentació de la plataforma per tal de facilitar al programador de la comunitat i a nosaltres mateixa la possible millora i manteniment que li se podria donar al projecte en un futur.
- **Analitzar la plataforma en un entorn real**. Caldria avaluar aquesta plataforma en un cas real, així podríem adonar-nos de com millorar la plataforma. A més a més, també trobaríem errors i mals funcionaments que sols serien detectables en el món real.
- **Integrar la *Microsoft Emotion API***. Podríem integrar la plataforma de *Microsoft* per tal d'oferir un altre predictor que milloraria el reconeixement, a més a més, de obtindre noves funcionalitats.
- **Complementar les bases de dades**. En un futur, podríem buscar més bases de dades de fotografies ja etiquetades amb els noms i emocions. Amb aquestes i amb la recol·lecció que fem en la nostra plataforma, complementarem la nostra base de dades. Amb aquestes noves mostres, farem més estudis i millores per tal d'adaptar-la al món real.

Bibliografia

Bakker, Iris et al. (2014). "Pleasure, Arousal, Dominance: Mehrabian and Russell revisited". A: *Current Psychology* 33.3, pàg. 405 - 421. ISSN: 1936-4733. DOI: 10.1007/s12144-014-9219-4 (v. la pàg. 5).

Calvo, Manuel G. i Daniel Lundqvist (2008). "Facial expressions of emotion (KDEF): Identification under different display-duration conditions". A: *Behavior Research Methods* 40.1, pàg. 109 - 115. ISSN: 1554-351X, 1554-3528. DOI: 10.3758/BRM.40.1.109 (v. la pàg. 22).

Dailey, Matthew N. et al. (2010). "Evidence and a computational explanation of cultural differences in facial expression recognition." A: *Emotion* 10.6, pàg. 874 (v. la pàg. 22).

Docker (2018). *Docker Reference*. URL: https://docs.docker.com/engine/reference/builder/ (cons. 03-06-2018) (v. la pàg. 7).

Huang, Minxue, Rizwan Ali i Junyun Liao (2017). "The effect of user experience in online games on word of mouth: A pleasure-arousal-dominance (PAD) model perspective". A: *Computers in Human Behavior* 75, pàg. 329 - 338. ISSN: 0747-5632. DOI: https://doi.org/10.1016/j.chb.2017.05.015 (v. la pàg. 5).

Kazemi, V. i J. Sullivan (2014a). "One millisecond face alignment with an ensemble of regression trees". A: *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pàg. 1867 - 1874. DOI: 10.1109/CVPR.2014.241 (v. la pàg. 15).

— (2014b). "One millisecond face alignment with an ensemble of regression trees". A: *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pàg. 1867 - 1874. DOI: 10.1109/CVPR.2014.241 (v. la pàg. 19).

Labeled Faces in the Wild (2018). URL: http://vis-www.cs.umass.edu/lfw/results.html#dlib (cons. 06-07-2018) (v. les pàg. 35, 36).

Meftah, I. T., N. L. Thanh i C. B. Amar (2010). "Towards an Algebraic Modeling of Emotional States". A: *2010 Fifth International Conference on Internet and Web Applications and Services*, pàg. 513 - 518. DOI: 10.1109/ICIW.2010.82 (v. la pàg. 5).

Mehrabian, Albert (1996). "Pleasure-arousal-dominance: A general framework for describing and measuring individual differences in Temperament". A: *Current Psychology* 14.4, pàg. 261 - 292. ISSN: 1936-4733. DOI: 10.1007/BF02686918 (v. la pàg. 5).

Rincon, J. A., V. Julian i C. Carrascosa (2015). "Social Emotional Model". A: *Advances in Practical Applications of Agents, Multi-Agent Systems, and Sustainability: The PAAMS Collection*. Ed. de Yves Demazeau et al. Cham: Springer International Publishing, pàg. 199 - 210. ISBN: 978-3-319-18944-4. DOI: 10.1007/978-3-319-18944-4_17 (v. la pàg. 5).

Rincon, J.A., A. Costa et al. (2018). "Introducing dynamism in emotional agent societies". A: *Neurocomputing* 272, pàg. 27 - 39. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2017.03.091 (v. la pàg. 5).

Rincon, J.A., F. de la Prieta et al. (2017). "Influencing over people with a social emotional model". A: *Neurocomputing* 231. Neural Systems in Distributed Computing and Artificial Intelligence, pàg. 47 - 54. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2016.03.107 (v. la pàg. 5).

Russell, S.J. i P. Norvig (2010). *Artificial Intelligence: A Modern Approach*. Prentice Hall series in artificial intelligence. Prentice Hall. ISBN: 9780136042594 (v. la pàg. 8).

Sohail, Abu Sayeed Md i Prabir Bhattacharya (2008). "Detection of Facial Feature Points Using Anthropometric Face Model". en. A: *Signal Processing for Image Enhancement and Multimedia Processing*. Ed. de Ernesto Damiani et al. Multimedia Systems and Applications Series 31. Springer US, pàg. 189 - 200. ISBN: 978-0-387-72499-7 978-0-387-72500-0. DOI: 10.1007/978-0-387-72500-0_17 (v. les pàg. 5, 20).

Verdú, C. (2016). "Desarrollo de un sistema multiagente para la detección de emociones en un robot bípedo". A: (v. les pàg. 11, 26, 35).

Viola, Paul i Michael Jones (2001). "Robust Real-time Object Detection". A: *International Journal of Computer Vision* (v. la pàg. 12).

Wooldridge, M. (2009). *An Introduction to MultiAgent Systems*. Wiley. ISBN: 9780470519462 (v. la pàg. 8).

Xavier, M. G. et al. (2013). "Performance Evaluation of Container-Based Virtualization for High Performance Computing Environments". A: *2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, pàg. 233 - 240. DOI: 10.1109/PDP.2013.41 (v. la pàg. 6).

APÈNDIX A
Configuració d'nginx

```
server {
listen 80;
server_name _;

location /spade/ {
    proxy_pass http://spade:8008/;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header Host $host;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
    proxy_set_header X-Forwarded-Port $server_port;
    proxy_redirect off;

    proxy_connect_timeout       3600;
    proxy_send_timeout          3600;
    proxy_read_timeout          3600;
    send_timeout                3600;

    # Use named locations from json snippet.
    error_page  301 302 =300 @json_redirect;
    error_page  404 =404 @json_not_found;
    error_page  500 502 504 =500 @json_server_error;
    error_page  503 =503 @json_maintenance;
}

location /face/model/ {
    proxy_pass http://face_recognizer:5001/;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header Host $host;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
    proxy_set_header X-Forwarded-Port $server_port;
    proxy_redirect off;

    proxy_connect_timeout       3600;
    proxy_send_timeout          3600;
    proxy_read_timeout          3600;
```

```
38      send_timeout                 3600;
39
40      # Use named locations from json snippet.
41      error_page  301 302 =300 @json_redirect;
42      error_page  404 =404 @json_not_found;
43      error_page  500 502 504 =500 @json_server_error;
44      error_page  503 =503 @json_maintenance;
45  }
46
47  location /face/ {
48      proxy_pass http://web_server:3000/;
49      proxy_set_header X-Real-IP $remote_addr;
50      proxy_set_header Host $host;
51      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
52      proxy_set_header X-Forwarded-Proto $scheme;
53      proxy_set_header X-Forwarded-Port $server_port;
54      proxy_redirect off;
55
56      proxy_connect_timeout       3600;
57      proxy_send_timeout          3600;
58      proxy_read_timeout          3600;
59      send_timeout                3600;
60
61      # Use named locations from json snippet.
62      error_page  301 302 =300 @json_redirect;
63      error_page  404 =404 @json_not_found;
64      error_page  500 502 504 =500 @json_server_error;
65      error_page  503 =503 @json_maintenance;
66  }
67
68  location @json_redirect {
69      default_type application/json;
70      return 200 \
71      '{"code":300,"error":true,"message":"Redirection found.","data":null}';
72  }
73
74  location @json_not_found {
75      default_type application/json;
76      return 200 \
77      '{"code":404,"error":true,"message":"Resource not found.", \
78      "data":null}';
79  }
80
81  location @json_server_error {
82      default_type application/json;
83      return 200 \
84      '{"code":500,"error":true,"message":"Server error.", \
85      "data":null}';
86  }
87
88  location @json_maintenance {
```

```
    default_type application/json;
    return 200 \
    '{"code":503,"error":true,"message":"Server temporarily down.", \
    "data":null}';
}

```