UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Universitat Politècnica de Catalunya (UPC) - BacelonaTech

Facultat d'Informàtica de Barcelona

Treball de Fi de Grau

Reconeixement d'actors en una escena mitjançant visió per computador

Autor	*Titulació*
Gerard Valls Ferrer	Grau en Enginyeria Informàtica
Director	*Especialitat*
Joan Climent Vilaró	Computació

28 d'octubre de 2018

Resum

L'Aprenentatge Profund és el principal responsable del creixement frenètic que ha viscut el camp de la Visió per Computador durant els darrers anys, permetent la invenció d'aplicacions sorprenents fins ara inimaginables. L'objectiu d'aquest projecte és crear un sistema exacte de reconeixement facial aplicat al món cinematogràfic, que sigui capaç de predir la identitat de tots els actors i actrius que apareguin en una escena. Per aconseguir-ho, s'implementarà un sistema software centrat en l'entrenament de Xarxes Neuronals Convolucionals amb definicions variades i algorismes diversos, mitjançant Python i Torch. Com a resultat, s'ha obtingut un sistema que reconeix 70 actors diferents amb una exactitud de validació de 98.29%, i suficientment ràpid com per ser executat en temps real. S'ha aconseguit aplicant la tècnica d'aprenentatge per transferència amb una Xarxa Neuronal Convolucional entrenada per representar mitjançant triplet loss. Són uns resultats encoratjadors, que fan albirar un futur més pràctic i funcional.

Resumen

El Aprendizaje Profundo es el principal responsable del crecimiento frenético que ha vivido el campo de la Visión por Computador durante los últimos años, permitiendo la invención de aplicaciones sorprendentes hasta ahora inimaginables. El objetivo de este proyecto es crear un sistema exacto de reconocimiento facial aplicado al mundo cinematográfico, que sea capaz de predecir la identidad de todos los actores y actrices que aparezcan en una escena. Para ello, se implementará un sistema software centrado en el entrenamiento de Redes Neuronales Convolucionales con definiciones variadas y algoritmos diversos, mediante Python y Torch. Como resultado, se ha obtenido un sistema que reconoce 70 actores distintos con una exactitud de validación de 98.29 %, y suficientemente rápido como para ser ejecutado en tiempo real. Se ha logrado aplicando la técnica de aprendizaje por transferencia con una Red Neuronal Convolucional entrenada para representar mediante triplet loss. Son unos resultados alentadores, que hacen vislumbrar un futuro más práctico y funcional.

Abstract

Deep Learning is the main reason of the frenetic growth that has experienced the field of Computer Vision over the last few years, allowing the invention of surprising applications that nobody ever imagined. The objective of this project is to create an accurate facial recognition system applied to the cinematic world, which is capable of predicting the identity of all the actors and actresses that appear in a scene. To achieve this, a software system will be implemented with the main focus on the training of Convolutional Neural Networks with different definitions and various algorithms, through Python and Torch. As a result, it has been obtained a system that recognizes 70 different actors with a validation accuracy of 98.29%, and fast enough to be executed in real time. This has been achieved by applying the technology of transfer learning with a Convolucional Neural Network trained to represent with triplet loss. These are encouraging results, which suggest a more practical and functional future.

Agraïments

En primer lloc, m'agradaria agrair a la meva família el seu suport incondicional i continu al llarg de tots aquests anys.

En segon lloc, agrair-li també al meu director de projecte Joan Climent Vilaró per haver confiat en mi i pel temps que ha dedicat a ajudar-me en l'elaboració d'aquest treball.

Índex

Capítol 1

Introducció

Abans de començar el treball és important contextualitzar una sèrie de conceptes de gran importància.

La Intel·ligència Artificial és un camp que comprèn totes les tècniques que permeten als ordinadors comportar-se com si fossin humans. Vehicles autònoms, assistents virtuals, diagnosi mèdica automàtica... incomptables aplicacions amb grans beneficis per la societat.

L'Aprenentatge Automàtic és una branca de la Intel·ligència Artificial que s'encarrega de permetre que les màquines aprenguin i siguin capaces de generalitzar comportaments que hagin vist amb anterioritat.

L'Aprenentatge Profund és el conjunt d'algorismes d'Aprenentatge Automàtic que pretén emular l'estructura i funcionament del cervell humà. Per fer-ho, simula les Xarxes Neuronals que es troben presents al cervell i imita les seves habilitats de reconeixement de patrons. Aquest camp ha renascut fa relativament pocs anys, en part gràcies als avenços de hardware, i sembla ser capaç de resoldre qualsevol situació que se li plantegi. És sens dubte la tècnica d'Intel·ligència Artificial més potent actualment, amb un brillant futur per davant.

La Visió per Computador és una branca de la Intel·ligència Artificial que bàsicament s'encarrega de dotar a la màquina de la capacitat d'entendre mostres visuals, ja siguin imatges o vídeos. Algunes aplicacions són el reconeixement òptic de caràcters, la conducció autònoma, el recompte de persones en un esdeveniment, el reconeixement facial... Actualment, els algorismes de Visió per Computador que millor funcionen provenen del camp de l'Aprenentatge Automàtic, i especialment del de l'Aprenentatge Profund, ja que les Xarxes Neuronals Convolucionals han superat amb escreix els algorismes de Visió per Computador convencionals.

El problema que es presenta a continuació correspon principalment al camp de la Visió

per Computador, encara que per resoldre'l s'utilitzaran també algunes tècniques d'Aprenentatge Automàtic que estan fora d'aquesta àrea.

1.1 Formulació del problema

El problema que es vol resoldre en aquest projecte és el reconeixement d'actors en una escena. En altres paraules, l'objectiu és predir la identitat de tots els actors que apareixen en un vídeo. Es tracta d'un problema de reconeixement facial (classificació multi-classe), ja que les característiques dels actors s'extrauran únicament de la cara, el component més característic de l'espècie humana.

Respecte els majors reptes d'aquest problema, a més d'haver de fer front a punts de vista diversos, oclusions i en ocasions una il·luminació complicada, el sistema de reconeixement facial haurà de ser resistent a variacions intrapersonals com són edat, maquillatge, expressions, accessoris (ulleres, barbes postisses), pentinat... i al mateix temps haurà de saber distingir quines són les variacions interpersonals. Alguns d'aquests reptes són molt difícils de superar, així que hi haurà una sèrie de restriccions que es descriuran a la secció de l'abast.

1.2 Motivació

El reconeixement facial és una tècnica molt important en l'àmbit de la seguretat, i crec que té un gran futur. Qui sap si d'aquí uns anys ens permetrà comprar a botigues de manera automàtica sense haver de passar per caixa, pagar automàticament al fer ús del transport públic, i moltes més aplicacions que ens permetran millorar la qualitat de vida. Ara bé, encara estem lluny d'això, i abans s'ha d'aconseguir que sigui infal·lible.

Més específicament, he decidit centrar-me en el reconeixement d'actors per tres raons principals: en primer lloc, sempre m'ha interessat el món cinematogràfic. En segon lloc, penso que pot ser un gran repte aplicar el reconeixement facial a un àmbit on una mateixa persona té diverses aparences en funció del paper que representa. Finalment, penso que el reconeixement d'actors pot tenir una sèrie d'aplicacions de gran utilitat (explicades més endavant a la secció 1.5.3).

Finalment, en relació als algorismes utilitzats per resoldre el problema, estic molt interessat en les tècniques d'Aprenentatge Automàtic aplicades al camp de la Visió per Computador, i més concretament en l'Aprenentatge Profund. Tot va sorgir quan per casualitat vaig llegir un article publicat a la revista *Nature* que parlava del *Deep Learning* [1]. Em

va semblar una tècnica amb grans possibilitats, i vaig decidir basar-hi el Treball de Fi de Grau. Tot i tenir un absolut desconeixement sobre el seu funcionament, estic molt interessat a estudiar i aprendre aquesta innovadora tècnica amb un gran futur per davant.

1.3 Objectius

Hi ha dos objectius principals que es pretenen complir en la realització d'aquest projecte per intentar resoldre el problema plantejat:

- Donat un vídeo o una imatge extret d'una pel·lícula o sèrie, reconèixer els actors i actrius que hi apareixen, etiquetant-los a la font original acompanyant a la corresponent cara, marcada amb una requadre.
- Buscar l'equilibri entre cost i fiabilitat, dos conceptes normalment oposats. El sistema ha de ser suficientment ràpid per poder tractar vídeos en temps real, i a la vegada ha de reconèixer els actors amb la màxima exactitud possible. Per trobar la solució més òptima s'hauran d'implementar i avaluar diferents tècniques i algorismes.

S'han creat nous objectius respecte la planificació inicial, els quals en realitat són subobjectius dels principals:

- Documentar-se i investigar diferents algorismes i tècniques relacionats amb el reconeixement facial que hi hagi actualment al mercat.
- Avaluar les tècniques existents i triar-ne la millor.
- Aprendre a utilitzar el software escollit per aplicar-lo a aquest problema.
- Obtenir conjunts d'imatges d'actors que permetin les tasques d'entrenament i de validació.
- Crear un sistema fiable de detecció de cares.
- Crear un sistema fiable de classificació de cares d'actors.
- Aplicar metodologies per dur a terme el projecte.

1.4 Abast

Com ja s'ha descrit a la formulació del problema, el projecte consisteix a desenvolupar un sistema que sigui capaç de reconèixer tots els actors que apareguin en una imatge o vídeo, tant si n'hi ha 1 com si n'hi ha més. El resultat s'ha de mostrar visualment sobre la font d'entrada, emmarcant amb un requadre les cares detectades i superposant un camp de text a sota de cadascun d'ells, indicant el nom de l'actor o actiu en qüestió (veure figura 1.1). Es tracta d'una de detecció, i una posterior de predicció, la qual necessita un classificador de tantes classes com actors tingui el sistema (70 com s'explicarà més endavant en el desenvolupament del projecte).

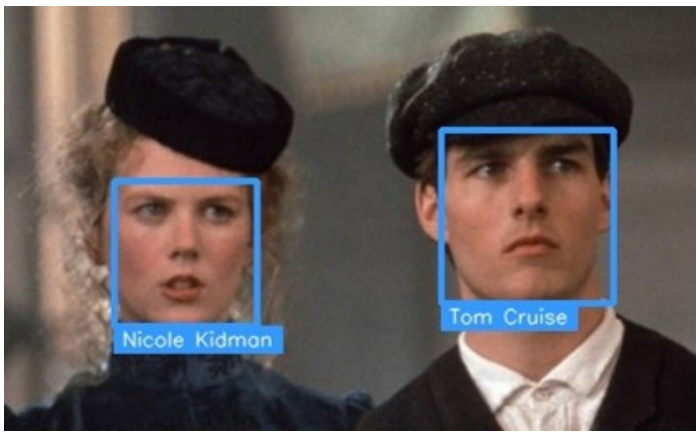

Figura 1.1: Exemple del sistema de reconeixement d'actors aplicat a una imatge.

Queda fora de l'abast d'aquest projecte el reconeixement d'actors desconeguts, ja que totes les cares que es detectin i no formin part de la llista de classes del sistema donaran una predicció errònia. El sistema de reconeixement d'actors s'aplicarà únicament a imatges o vídeos en què tots els actors que hi apareguin formin part del conjunt de classes recognoscibles.

Hi ha una sèrie de restriccions addicionals: totes les imatges i vídeos han de tenir bona definició i resolució, les cares han d'estar mirant cap a la càmera (o bé lleugerament rotades i de perfil), i la il·luminació pot ser variable però no extremadament dolenta. A més, l'entrada haurà de ser en color (canals RGB) i no en nivell de gris.

Respecte la part més relacionada amb l'implementació, es programaran 6 mòduls principals: la creació d'un conjunt d'imatges d'aprenentatge i validació, la detecció de cares, l'alineament i retallat de cares (preprocessament), la classificació d'una cara en l'actor corresponent, el sistema de reconeixement d'actors en imatge, i en vídeo. Per fer-ho, s'utilitzaran sobre tot les tècniques d'Aprenentatge Automàtic que es descriuran a l'estat de

l'art (secció 1.7.2). S'implementaran 2 variants del mòdul de detecció i 4 del de classificació, aplicant algorismes diferents. D'aquesta manera, es podrà analitzar quina és la combinació més adient per resoldre el problema.

Quant a la interfície gràfica, no es destinaran recursos a intentar millorar-la. Donat el meu desconeixement del tema i els alts requeriments de software del sistema (llibreries de Python i Torch), seria temporalment costós, i podria desviar-me de l'objectiu principal d'aquest projecte, que és comparar diferents algorismes i tècniques per obtenir el sistema de reconeixement més fiable possible. Per tant, l'aplicació s'executarà per consola afegint com a arguments els noms dels arxius d'entrada.

1.5 Actors implicats

En aquest treball es poden distingir fins a tres classes d'actors implicats: el desenvolupador del projecte, el director i els beneficiaris.

1.5.1 Desenvolupador

Jo, el desenvolupador del projecte, m'encarregaré que es compleixin tots els objectius tot intentant respectar la planificació temporal estipulada. També em considero part del grup de beneficiaris, per les causes descrites a la secció 1.2.

1.5.2 Director

El director del projecte és en Joan Climent Vilaró, qui s'encarregarà de supervisar el projecte, ajudant si és necessari en el seu desenvolupament i vetllant perquè es compleixin satisfactòriament totes les fites establertes.

1.5.3 Beneficiaris

Aquesta classe fa referència a totes les persones que poden fer ús de l'aplicació, i per tant treure'n profit. Les he dividit en dues categories: individus i empreses.

Els individus són tots aquells que poden utilitzar el sistema de reconeixement d'actors aplicat al món de l'oci. D'una banda, similar a l'aplicació *Shazam* (reconeixement de música), pot ser de gran utilitat per obtenir ràpidament el nom d'un actor que desconeixes i del que t'agradaria poder veure més pel·lícules. D'altra banda, també pot ser útil en l'àmbit

de les cerques basades en contingut. Per exemple, pot permetre als usuaris buscar tots els tràilers on aparegui un determinat actor, sense haver de recórrer al títol, les etiquetes... unes dades que no sempre són fiables.

Les empreses cinematogràfiques també poden treure profit d'aquest sistema, sent de gran ajuda per organitzar els diferents clips del rodatge en funció dels actors que hi apareixen, de forma completament automatitzada, fent que la posterior edició sigui més estructurada, còmoda i eficient. A més, també els permetria extreure estadístiques de les pel·lícules.

1.6 Possibles obstacles

Les dificultats que poden sorgir al llarg del treball són diverses. La primera d'elles és que el reconeixement facial és un cas difícil de reconeixement d'objectes, ja que totes les cares són molt semblants entre elles. A més, una cara humana no és un objecte rígid, sinó que la seva aparença és canviant, ja sigui a causa de factors intrínsecs (deguts a la naturalesa física de la cara com l'edat, expressió facial, ulleres...) o extrínsecs (deguts a l'entorn i a l'observador: il·luminació, oclusions, punt de vista, resolució...).

S'ha d'intentar que el sistema tingui en compte aquests factors de variació, semt resistent a variacions intrapersonals i al mateix temps sabent distingir les interpersonals. Per fer-ho, serà determinant utilitzar un conjunt d'imatges d'aprenentatge de gran diversitat, que inclogui imatges amb totes les variacions intrínseques i extrínseques possibles per cadascun dels actors. Aquest pas és determinant, ja que si només s'escollissin les fotografies que es fan els actors als festivals de cinema, podria ser que causés uns resultats decebedors amb imatges extretes d'una pel·lícula, on les expressions facials i altres condicions són molt diferents.

En relació a la implementació, tant l'Aprenentatge Automàtic com algunes de les llibreries utilitzades són plenament desconeguts per mi, i la documentació serà clau per aprendre el seu funcionament i corregir tots els errors que puguin sorgir.

La darrera i major dificultat en aquest projecte serà la gestió del temps, i és que són molts els entrebancs que poden causar desviacions respecte la planificació inicial.

1.7 Estat de l'art

El reconeixement facial és un problema que està en plena fase de recerca i desenvolupament i, com a tal, requereix de les millors tècniques dins el camp de la Intel·ligència Artificial. Per trobar-les, és de gran utilitat analitzar els sistemes de reconeixement facial més importants que hi ha actualment al mercat.

1.7.1 Sistemes

Hi ha moltes empreses importants que s'han dedicat a millorar el reconeixement facial, augurant-li una gran importància en el futur. És el cas de Google i Facebook, amb els sistemes FaceNet [2] i DeepFace (utilitzat en l'*auto tagging* de les persones que apareixen en una imatge) [3] respectivament. Són relativament recents, del 2015 i 2014, i fan ús de l'Aprenentatge Profund, una tècnica que ha passat a ser imprescindible els darrers anys, encarregada de la tasca d'extracció de característiques. FaceNet es caracteritza per un mètode d'entrenament pioner (*triplet loss*), i això afegit a un dataset d'aprenentatge de 260 milions d'imatges li permet obtenir uns resultats quasi perfectes. DeepFace utilitza un entrenament més bàsic (*cross-entropy loss*), però obté uns resultats igualment satisfactoris gràcies a preprocessar les cares, alineant-les, permetent reconèixer amb més fiabilitat cares que no estiguin mirant directament a la càmera sinó lleugerament de perfil o amb el cap inclinat.

Aquests són dos sistemes referència en el reconeixement facial, però n'hi ha un que ha incorporat les fortaleses d'ambdós. És el cas d'Openface [4], un sistema *open source* que, tot i disposar d'un conjunt d'imatges d'aprenentatge molt més reduït que els altres dos (500 mil imatges), ha obtingut uns resultats similars.

El reconeixement facial és un problema de gran complexitat i en boca de totes les grans empreses, així que més que crear un sistema completament innovador, que seria molt difícil, em basaré en les tècniques ja existents. Principalment faré ús del codi de Openface, compartit en un repositori públic [5], i a més de fer-hi les pertinents modificacions hi afegiré altres tècniques de detecció i classificació que m'ajudin a complir els objectius proposats. A continuació es fa una revisió de les tècniques més rellevants en l'àmbit del reconeixement facial, presents sobre tot en el sistema que prendré com a referència (Openface).

1.7.2 Tècniques

Per resoldre un problema d'aquesta magnitud és imprescindible utilitzar les tècniques més punteres dins de l'àmbit de la Visió per Computador, i és aquí on entra en joc l'Aprenentatge Automàtic, més específicament el supervisat, on la màquina aprèn per mitjà d'un conjunt de dades etiquetades amb la respectiva classe. Com es descriu a continuació, la majoria d'algorismes utilitzats en aquest treball estan relacionats amb aquesta branca de la Intel·ligència Artificial, tot i que encara es conserven algunes tècniques de Visió per Computador convencionals.

El primer pas pel reconeixement facial és la detecció de cares, una tasca que si no es resol adequadament pot arribar a suposar un coll d'ampolla en la resolució del problema general. S'utilitzaran dos algorismes diferents: un basat en característiques de Haar (Viola-Jones) [6], i l'altre en els Histogrames de Gradients orientats [7]. Són tècniques amb certa antiguitat però que segueixen donant uns resultats *state-of-the-art*.

El segon pas és el preprocessament de les cares detectades, alineant-les i així millorant els resultats del sistema amb les cares que estiguin lleugerament rotades. S'utilitzarà un conjunt d'arbres de regressió, seguint un algorisme del 2014 [8] que també està emmarcat dins l'àmbit de l'Aprenentatge Automàtic.

El tercer pas és la classificació d'actors, que anirà principalment a càrrec de l'Aprenentatge Profund, una classe de tècniques d'Aprenentatge Automàtic que és imprescindible per extreure característiques en els sistemes de reconeixement facial moderns. L'Aprenentatge Profund (*Deep Learning* en anglès) és una tècnica *state-of-the-art* que va sorgir cap als anys 80, però al ser tan cara computacionalment estava en una situació d'hivernació. Va ser cap al 2012 que, gràcies a les millores de hardware, va passar a ser una tècnica referent en l'extracció de característiques en imatges [9]. Per fer-ho, utilitza les anomenades Xarxes Neuronals Convolucionals, formades per un conjunt de capes, cadascuna d'elles composada a la vegada per múltiples neurones. Cada neurona té una sèrie de paràmetres, els quals s'inicialitzen aleatòriament i es van modificant a mesura que es van fent passades amb el conjunt de dades d'aprenentatge, extraient cada vegada característiques més rellevants pel problema concret. Per fer-ho s'intenta minimitzar la *loss*, una funció que representa l'error de cada configuració de la xarxa [10], [11].

Hi ha una última tècnica que serà clau en la detecció de cares i en la classificació d'actors, i que tot i datar del 1995 es segueix utilitzant en la majoria de projectes: són les *Support Vector Machines* [12], l'única tècnica d'aquest treball que no pertany al camp de la Visió per Computador.

1.8 Metodologia i rigor

És imprescindible seguir una bona metodologia per tal de respectar amb el màxim rigor possible la planificació inicial. Per fer-ho, s'utilitzarà un model de desenvolupament en cascada [13].

Com s'ha descrit a la secció 1.4, el treball es dividirà en una sèrie de mòduls: 1) la creació d'un conjunt d'imatges d'aprenentatge i validació, 2) la detecció de cares, 3) l'alineament i retallat de cares (preprocessament), 4) la classificació d'una cara a la classe corresponent, 5) el sistema de reconeixement en imatge i 6) el sistema de reconeixement en vídeo. Alguns d'aquests mòduls tindran diferents variants amb dissenys alternatius: el de detecció s'implementarà amb Histogrames de Gradients orientats i també amb l'algorisme de Viola-Jones; el de classificació es crearà aplicant 4 tècniques diferents, totes elles centrades en les Xarxes neuronals Convolucionals. A l'apartat de planificació temporal es farà un anàlisi més exhaustiu de les tasques realitzades i la seva execució en el temps.

Cadascun dels mòduls (variants si n'hi ha) a excepció del primer estarà dividit a la vegada en 4 fases iteratives: anàlisi, disseny, implementació i proves. S'haurà d'anar programant els mòduls per ordre, i només quan les proves d'un mòdul siguin satisfactòries es podrà començar amb el següent. Excepcionalment i si hi ha problemes de temps, es podrà saltar una variant d'un mòdul si ja n'hi ha una altra d'implementada (per exemple, un dels dos algorismes del mòdul de detecció de cares), i si no es creen dependències difícils de corregir.

La metodologia escollida (model de desenvolupament en cascada) és estructurada i eficient, així que s'adapta perfectament a aquest projecte donat que és principalment lineal i es pot fer per etapes. El principal avantatge és que els errors que sorgeixin estaran delimitats a un mòdul aïllat, i no serà gaire complicat detectar-ne la causa i corregir-los. En canvi, sí que ho seria si s'executés tot el codi al final, ja que s'hauria d'anar cap enrere analitzant mòdul a mòdul indefinidament.

Destacar també que s'intentarà informar al director del projecte al finalitzar cada mòdul, per tal d'obtenir un *feedback* molt valuós en la realització de les següents tasques.

1.9 Mètodes de validació

Primerament, les reunions presencials i no presencials que s'organitzaran amb el director permetran veure com d'avançat es troba el projecte, ajudant a corregir possibles obstacles que puguin sorgir.

En relació a la part més tècnica, la fase de proves de cada mòdul serà clau, ja que fins que no es superi no es podrà passar a l'anàlisi i disseny del següent mòdul, sinó que s'haurà de redissenyar l'actual. Les proves es faran principalment amb dos conjunts d'imatges: el d'aprenentatge i el de validació. Com s'explicarà posteriorment a l'apartat 3.1, ambdós han de tenir imatges dels mateixos actors. A més, han de ser disjunts, evitant que algunes de les imatges de validació s'hagin utilitzat en l'entrenament, fet que sobrevaloraria l'exactitud obtinguda. En mòduls com per exemple la detecció de cares, que no necessitin una fase d'entrenament, s'utilitzarà el conjunt d'imatges d'aprenentatge (més gran) com a conjunt de validació. Per provar el sistema de reconeixement en vídeo no es disposa de cap dataset, només una sèrie d'escenes extretes de pel·lícules a partir de les quals es podrà fer un anàlisi subjectiu.

1.10 Recursos de desenvolupament

En relació als recursos hardware, es programarà amb un ordinador de sobretaula amb processador Intel i5 7600k i memòria RAM de 16 GB. A més, s'utilitzarà un telèfon mòbil per la fase de documentació i recerca d'informació quan no es disposi de l'ordinador.

Respecte el software per implementar el projecte, es farà ús principalment del llenguatge de programació Python 2.7, amb algunes llibreries de Visió per Computador i Aprenentatge Automàtic com són OpenCV, Dlib, Pytorch i Sklearn principalment. Per entrenar Xarxes Neuronals Convolucionals mitjançant *triplet loss* s'utilitzarà també Torch, una plataforma de computació científica de gran eficiència que utilitza el llenguatge de programació LuaJIT. També necessitaré el codi del sistema Openface, el qual adaptaré a aquest problema concret. El sistema operatiu serà Ubuntu 16.04. Com que programaré sempre amb el mateix ordinador, no necessitaré GitHub pel control de versions. Això sí, faré còpies de seguretat periòdicament.

Per escriure els informes del projecte utilitzaré el sistema operatiu Windows 10, el software Microsoft Office 2016, i TeXstudio (amb la distribució MiKTeX).

Finalment, també s'invertiran recursos humans (com a desenvolupador del projecte realitzaré les tasques de cap de projecte, dissenyador, programador i tester), i també hi haurà despeses indirectes com són el consum de paper i d'electricitat.

Capítol 2

Marc teòric

Per entendre el funcionament dels algorismes proposats pel reconeixement d'actors és imprescindible tenir presents una sèrie de conceptes teòrics. En primer lloc s'explicaran les característiques de Haar, els Histogrames de Gradients orientats, i els *ferns*, utilitzats en la detecció i preprocessament de les cares. En segon lloc, s'analitzaran les característiques bàsiques de les Xarxes Neuronals Convolucionals, un model al voltant del qual es desenvolupa tot el treball. Finalment, les *Support Vector Machines* són una eina de gran utilitat en la tasca de classificació, present tant en la detecció de cares com en la classificació d'actors.

Al capítol 3 es posaran en pràctica tots els coneixements teòrics per intentar resoldre el problema plantejat.

2.1 Característiques de Haar

Les característiques de Haar són el resultat d'aplicar un conjunt de filtres de Haar en diferents posicions d'una imatge. Un filtre bàsic de Haar és una combinació de rectangles blancs i negres adjacents els uns amb els altres (veure petita mostra a figura 2.1). També n'hi ha de més complexos, amb diferents figures i orientacions. El resultat d'aplicar el filtre en una determinada posició d'una imatge es calcula com la diferència entre la suma de les intensitats dels píxels de la zona negra del filtre (positiu) i la suma dels valors de la zona blanca (negatiu). Això permet detectar canvis d'intensitat en tota la imatge, i poder identificar així arestes i línies per exemple.

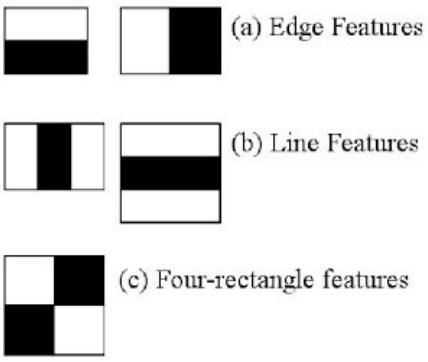

Figura 2.1: Filtres bàsics de Haar.

A la figura 2.2 es pot veure el resultat d'aplicar alguns filtres bàsics de Haar en totes les posicions d'una imatge (en aquest cas concret una cara amb un fons homogeni).

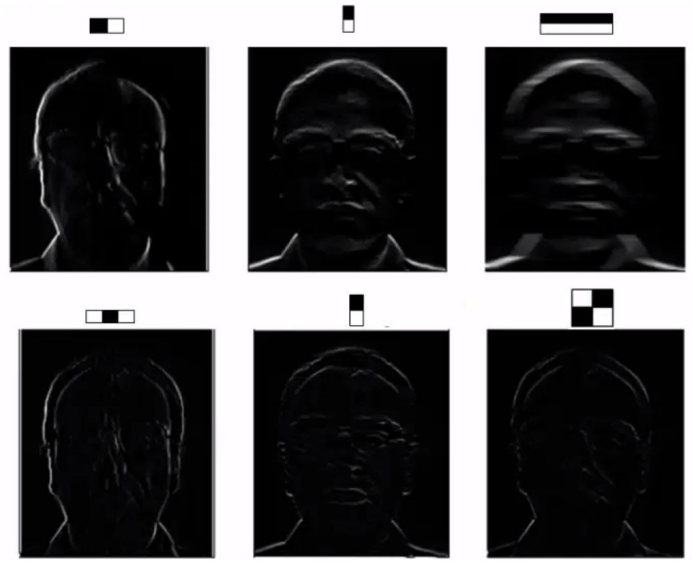

Figura 2.2: Filtres de Haar aplicats a la imatge d'una cara.

2.2 Histogrames de Gradients orientats

Els Histogrames de Gradients orientats o HoG són descriptors de característiques. La imatge d'entrada es divideix en blocs (solapats entre ells), els quals es divideixen a la vegada en cel·les i, per cadascuna d'elles, es calcula un histograma d'una dimensió (9 barres) que representa la magnitud de les direccions dels gradients de la cel·la. El descriptor final està

format per la concatenació dels histogrames de cada bloc, els quals són una normalització de la concatenació dels histogrames de cada cel·la del bloc. D'aquesta manera, una mateixa cel·la contribueix més d'una vegada en el descriptor final [7].

A la figura 2.3 es pot apreciar una mostra gràfica de les direccions del gradient en cadascuna de les cel·les de la imatge d'una cara (abans de convertir-les en un histograma). A la figura 2.4 es pot veure el procés d'obtenció de l'histograma de cada cel·la (cas simplificat de 4 barres), i l'histograma d'un determinat bloc (sense normalització).

Figura 2.3: Direccions del gradient en la imatge d'una cara.

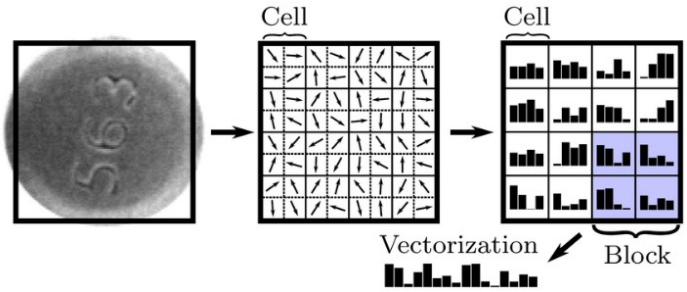

Figura 2.4: Procés d'obtenció dels Histogrames de Gradients orientats a partir d'una imatge.

2.3 *Fern*

Un *fern* és un arbre de decisió, més concretament un arbre de regressió ja que el resultat (valor de les fulles) és continu. Es tracta d'un arbre binari amb la característica que a cada nivell de l'arbre es realitza exactament el mateix test binari (veure estructura esquemàtica

a la figura 2.5). Utilitzant un conjunt de *ferns* es poden crear models predictius com s'explicarà a l'apartat 3.3.1, amb una gran eficiència i fiabilitat [14].

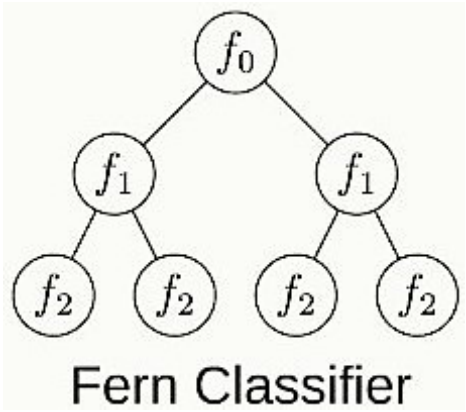

Figura 2.5: Estructura d'un *fern*, un arbre binari on es fa el mateix test als nodes d'un mateix nivell.

2.4 Xarxa Neuronal Convolucional

Una Xarxa Neuronal Convolucional o CNN (*Convolutional Neural Network* en anglès) és l'arquitectura d'Aprenentatge Profund més popular. És capaç de detectar automàticament i de manera eficient les característiques més rellevants per cada problema, mitjançant un entrenament amb un conjunt de dades etiquetades (imatges si la xarxa neuronal és convolucional com en aquest cas). A continuació s'analitza la seva arquitectura, el procés d'entrenament i dos problemes molt freqüents: l'underfitting i l'overfitting.

2.4.1 Arquitectura

Una CNN està formada a grans trets per una capa d'entrada i una de sortida, separades per una sèrie de capes ocultes. Cada capa està formada per múltiples neurones, les quals estan relacionades amb neurones de la capa anterior mitjançant pesos variables (paràmetres de la CNN). Cada neurona conté un valor, el qual s'anomena valor d'activació i es calcula a partir dels valors d'activació de les neurones de l'anterior capa amb qui està relacionada. La figura 2.6 mostra un esquema bàsic de l'estructura d'una Xarxa Neuronal.

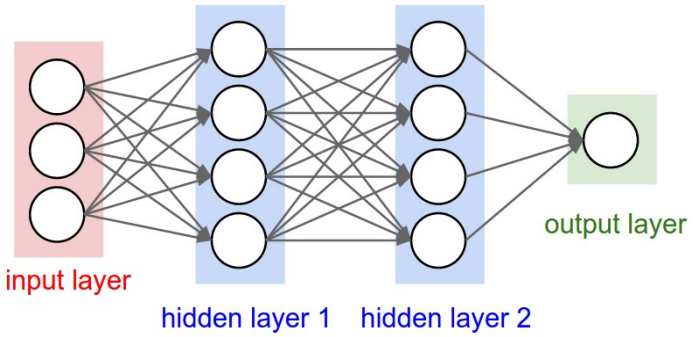

Figura 2.6: Arquitectura esquemàtica d'una Xarxa Neuronal Profunda, formada per una sèrie de capes connectades.

En problemes de Visió per Computador com aquest, la capa d'entrada és una imatge, i el valor d'activació de cadascuna de les seves neurones correspon a la intensitat de cada píxel de la imatge d'entrada (3 canals RGB). La capa de sortida és del tipus *fully-connected*, i els seus valors d'activació són determinants per fer prediccions. Les capes ocultes poden ser una seqüència de capes de dimensions i tipus diferents: convolucionals, *pooling*, *fully-connected* (la figura 2.7 representa un exemple bàsic de l'arquitectura d'una Xarxa Neuronal Convolucional). També hi a una sèrie de funcions que es poden aplicar als valors d'activació d'una capa com són *ReLU*, *softmax* i *batch normalization*. Finalment, hi ha uns mòduls anomenats *Inception* que estan formats per vàries convolucions i *pooling* en paral·lel. Aquests mòduls es poden concatenar amb altres mòduls *Inception* o altres capes.

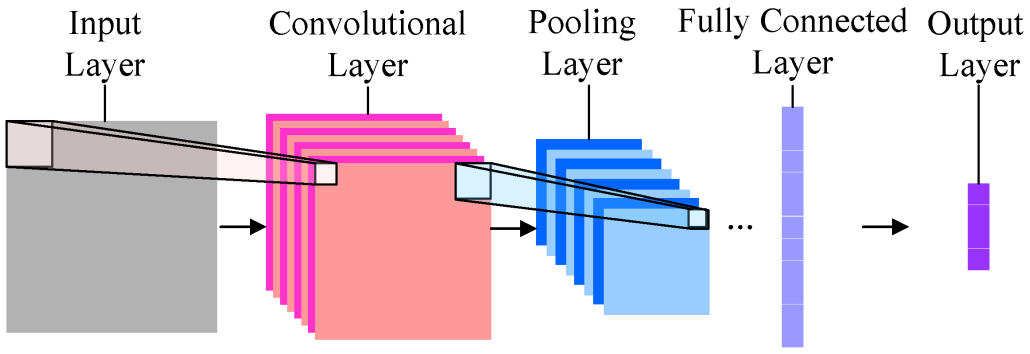

Figura 2.7: Exemple bàsic de l'arquitectura d'una Xarxa Neuronal Convolucional. La capa d'entrada és una imatge, la primera capa oculta és convolucional, la segona de *pooling*, i la darrera és *fully-connected*.

A continuació es descriu breument cadascun dels components principals d'una CNN.

2.4.1.1 Capa convolucional

Els valors d'activació de la capa convolucional es generen mitjançant una sèrie de filtres aplicats als valors d'activació de la capa anterior, o bé als de la capa d'entrada de la CNN. Un filtre és un array de dimensions HxWxD (D ha de coincidir amb la profunditat del volum d'entrada), format per diferents pesos (un per cada posició del filtre) i un bias. Aquest filtre, anomenat també kernel, es va lliscant pel volum d'entrada, avançant un determinat stride (la regió de l'entrada sobre la qual està el filtre s'anomena *receptive field*). A cada iteració, es calcula el producte dels pesos del filtre pels corresponents valors del *receptive field*, i se li suma el bias, obtenint un únic valor que s'emmagatzema a la corresponent posició d'un nou volum de sortida, que contindrà els valors de la capa actual. Després de recórrer tota l'entrada amb el mateix kernel s'acaba obtenint un volum de profunditat 1 (*activation map*). Per poder extreure més característiques i obtenir un sistema més complex, en una capa convolucional es poden utilitzar diversos filtres, donant lloc a un volum de sortida tan profund com filtres hi hagi. La figura 2.8 escenifica una convolució on l'input I és de 7x7x1, hi ha un únic filtre K de mida 3x3x1, i per tant l'array que s'obté (amb stride 1) és de 5x5x1. Si hi hagués 10 filtres, seria de 5x5x10. Si l'input tingués 3 canals (una imatge en color), el filtre seria de 3x3x3, però l'output seguiria sent 5x5x1. Com es pot veure, les dimensions H i W disminueixen. Per evitar-ho, es podria afegir un padding, amb el qual s'expandeix l'entrada amb 0's als marges, augmentant així la mida de l'array resultant [15], [16].

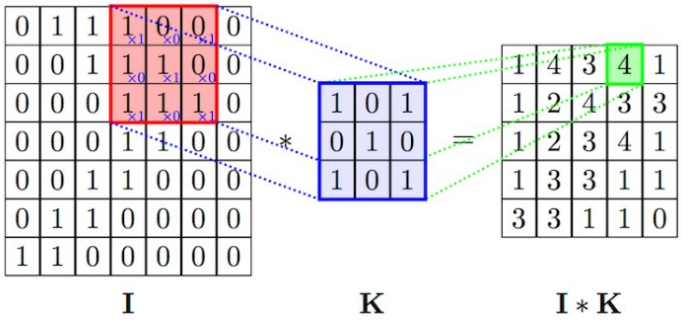

Figura 2.8: Exemple simplificat de convolució amb un fitre K de mida 3x3 (stride 1 i padding 0) aplicat a una imatge I de mida 7x7x1. El resultat és I*K.

Els paràmetres d'aquesta capa seran els pesos i el bias de cada filtre.

2.4.1.2 Capa de *pooling*

Una altra capa de gran utilitat és la de *pooling*, que consisteix a reduir la mida de l'entrada tot conservant la informació més rellevant. Amb això s'aconsegueix que la quantitat

de paràmetres decreixi notablement, disminuint així el cost computacional i l'overfitting (terme que s'explicarà més endavant). El *pooling* consisteix a aplicar la funció de mitjana, màxim o suma amb un filtre de mida HxWx1. Com amb la convolució, es poden afegir els paràmetres de stride i padding. Amb aquesta capa s'aconsegueix reduir l'alçada i l'amplada del volum d'entrada, però la profunditat es manté constant. A la figura 2.9 es representa el càlcul dels valors d'activació d'una capa de *max-pooling* amb filtre de 2x2 i stride 2, a partir dels valors de la capa anterior [15], [16].

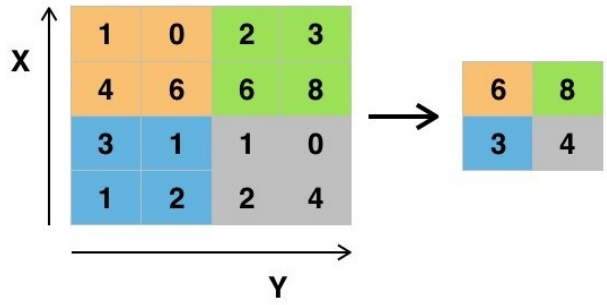

Figura 2.9: Exemple de *max-pooling* amb una mida de filtre de 2x2 aplicat a una imatge de mida 4x4x1 (stride 2 i padding 0). Els valors d'activació de la nova capa corresponen al valor màxim del corresponent *receptive field* de la capa anterior (mateix color).

Aquesta capa no té paràmetres.

2.4.1.3 Capa *fully-connected*

Aquesta capa té mida 1x1xD, i sol estar ubicada al final de les Xarxes Neuronals Convolucionals. Cadascuna de les neurones està connectada a totes les neurones de la capa anterior. El valor d'activació d'una neurona és el producte dels valors d'activació de la capa anterior i els pesos de la connexió amb aquests, sumat a un bias [15], [16]. Tant les capes ocultes com la de sortida de la figura 2.6 són *fully-connected*.

Per tant, el número de paràmetres dependrà del número de neurones d'aquesta capa i de l'anterior. Per exemple, si la capa anterior és també *fully-connected* de mida 1x1x512 i la actual és 1x1x64, per cadascuna de les 64 neurones hi haurà 512 pesos i 1 bias. És a dir, un total de 32832 paràmetres.

En cas que la capa anterior no sigui *fully-connected*, el que es sol fer és utilitzar abans la funció *flatten* per redimensionar el volum a la forma 1x1xD. Per exemple, si la capa anterior (*max-pooling*) té dimensions 8x8x16, es pot convertir a un volum de 1x1x1024 mitjançant una operació de *flatten*, i passar-lo com a entrada a una capa *fully-connected*.

2.4.1.4 Funció d'activació *ReLU*

Les Rectified Linear Units són funcions d'activació no lineals que solen aplicar-se immediatament després de les capes convolucionals. Consisteix a aplicar la funció $max(0, x)$ a cadascun dels valors d'activació x de la capa corresponent, abans d'utilitzar-los per calcular els valors de la següent capa. Amb això, principalment s'aporta no linealitat a una xarxa que només ha operat linealment durant la convolució, permetent així l'aprenentatge de característiques més complexes (si no s'apliqués aquesta funció d'activació seria equivalent a tenir una única capa amb més neurones). A més, també es redueix el temps d'entrenament, gràcies a facilitar el càlcul de derivades [17].

2.4.1.5 Funció d'activació *Softmax*

Aquesta funció d'activació genera una distribució de probabilitats sobre les classes a partir dels valors d'activació que li entren. En altres paraules, donat un vector de mida K (número de classes), retorna un vector de mida K on tots els valors sumen 1, i cada element representa la probabilitat que la imatge d'entrada a la CNN pertanyi a aquella classe [17]. És una funció que es sol aplicar a la capa de sortida de les Xarxes Neuronals que tenen com a objectiu la classificació. La funció es defineix per l'expressió 2.1.

$$\sigma(x_j) = \frac{e^{x_j}}{\sum_i e^{x_i}} \tag{2.1}$$

on x és el vector d'entrada, i x_j és l'element de x al qual se li està aplicant la funció.

La figura 2.10 mostra el resultat d'aplicar la funció *softmax* a un array de mida 3.

Figura 2.10: Exemple de la sortida obtinguda al aplicar la funció *softmax* a un array de mida 3.

2.4.1.6 Mòdul *Inception*

Aquest mòdul va sorgir per la dificultat de seleccionar la mida dels filtres de convolució més adient per cada situació. Per això, es fan convolucions amb filtres de mida 1x1, 3x3 i 5x5, i *max-pooling* de 3x3 amb els mateixos valors d'entrada, i es concatenen les 4 sortides

(veure figura 2.11). D'aquesta manera s'aconsegueix que sigui la pròpia xarxa que durant l'entrenament aprengui quins són els filtres més importants.

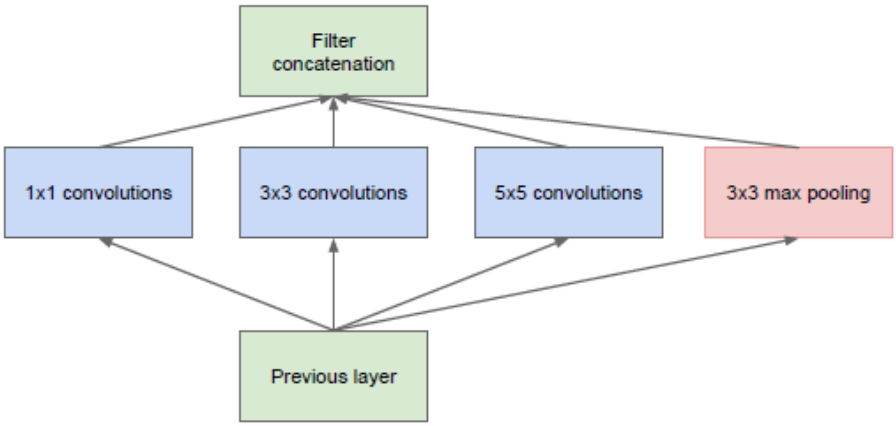

Figura 2.11: Arquitectura naïf del mòdul *Inception*, on simplement es concatenen les sortides de les convolucions 1x1, 3x3 i 5x5, i la sortida de *max-pooling* de mida 3x3.

El problema d'utilitzar aquest mòdul seria l'elevat número de paràmetres. És per això que s'utilitzen convolucions de mida 1x1, amb les quals s'aconsegueix una sortida de volum idèntic, però reduint considerablement el número de paràmetres. Per tant, s'està disminuint el cost computacional tot perdent la mínima informació [18], [19]. El mòdul *Inception* definitiu es veu a la figura 2.12.

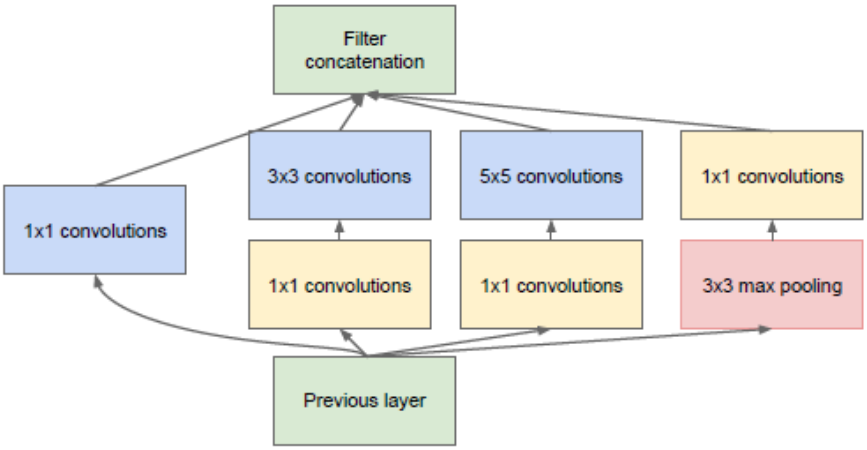

Figura 2.12: Arquitectura real del mòdul *Inception*, caracteritzada per les convolucions de mida 1x1 que permeten reduir el número de paràmetres mantenint la mida del volum de sortida.

2.4.2 Entrenament

Com s'ha vist a l'apartat d'arquitectura, una Xarxa Neuronal Convolucional està formada per una sèrie de paràmetres: pesos i bias. Aquests són desconeguts i inicialitzats aleatòriament, i s'han d'anar modificant per tal d'aconseguir la configuració que doni els millors resultats per resoldre un determinat problema. El procediment d'entrenament (amb minibatches) és el següent, suposant que es disposa d'un conjunt de dades d'aprenentatge [20].

- La xarxa s'entrena amb iteracions, les quals s'anomenen epochs. Cada epoch està format per una sèrie de minibatches, executats també iterativament. Els paràmetres de la xarxa s'actualitzen després de cada minibatch.

- Per cada minibatch es selecciona aleatòriament un subconjunt de les dades d'aprenentatge d'una mida determinada. Aquesta mida s'ha d'escollir a consciència, ja que si és molt petita hi haurà massa soroll i cost computacional, mentre que si és massa gran el cost computacional serà menor però al haver-hi poques actualitzacions pot suposar una convergència menys robusta, podent convergir ràpidament a mínims locals.

- Una vegada creat el conjunt del minibatch, s'administren les seves dades com a entrada a la xarxa. Al arribar a la capa de sortida, s'obtindran uns valors d'activació diferents per cada entrada. Aquesta fase és el que es coneix com *forward-pass*. Amb els valors de sortida de totes les dades del minibatch, es calcula la *loss* mitjana. Aquest càlcul es fa amb la funció *loss*, la qual depèn de cada problema concret, i es calcula tenint en compte l'etiqueta de la classe de cada dada d'aprenentatge. La *loss* es podria definir com l'error de la configuració actual de la xarxa, i com més petit sigui el seu valor millor està entrenada la xarxa.

- Per cada exemple individual del minibatch es calcula la derivada parcial de la *loss* respecte els valors d'activació de la capa de sortida, obtenint així els gradients de la sortida. A continuació, mitjançant el procés de *back-propagation*, es calcula el gradient per la resta de paràmetres de la xarxa, calculant les derivades parcials respecte cada paràmetre de cada neurona des de la capa final fins a la inicial. Després, es fa la mitjana dels gradients calculats per cadascun dels paràmetres. Una vegada es té el gradient mitjà de cada paràmetre, s'aplica un optimitzador per actualitzar els paràmetres de la xarxa. En la versió més bàsica, es multiplica el gradient per un coeficient d'aprenentatge *learning rate*, i se li resta a cada paràmetre. El *learning rate* per tant determina la velocitat d'entrenament, i ha de ser determinat amb cura ja que pot comportar una convergència massa lenta o bé l'estancament en un mínim local. Els algorismes d'optimització més utilitzats són SGD i Adam [21], i el seu objectiu és minimitzar la *loss* mitjana de cada minibatch.

- Per tant, per cada epoch, els paràmetres de la xarxa s'actualitzaran tantes vegades com minibatches hi hagi. Per identificar el moment en què la xarxa ja està entrenada, s'han de fer tests de validació al final de cada epoch amb un conjunt de dades que sigui disjunt amb el d'entrenament.

2.4.3 Underfitting i overfitting

L'underfitting es refereix a la situació en què una xarxa entrenada no dona bons resultats ni amb el conjunt d'aprenentatge ni amb el de validació. No acostuma a ser un problema de gran importància ja que és fàcil de detectar i de corregir. La solució sol ser afegir més paràmetres a la xarxa perquè pugui extreure més característiques i de major complexitat (afegint noves capes o més neurones a les ja existents), canviar l'algorisme d'optimització, o simplement seguir entrenant durant més epochs.

L'overfitting, en canvi, succeeix quan la xarxa dóna bons resultats amb el conjunt d'aprenentatge, però no tan satisfactoris amb el de validació. El model ha après cada detall de les imatges d'aprenentatge fins al punt que no és capaç de generalitzar a imatges amb les que no ha sigut entrenat [22]. Aquest problema és més complicat de resoldre que l'anterior, i s'hi acostuma a fer front amb més freqüència. Algunes possibles solucions són: ampliar el conjunt d'aprenentatge, reduir el número de paràmetres de la xarxa, no entrenar durant tants epochs (*early stopping*), canviar el mètode d'entrenament o bé utilitzar tècniques de regularització. Les que donen millors resultats són *dropout* i *batch normalization*:

- **Dropout:** és una tècnica de regularització utilitzada per reduir el risc d'overfitting. Consisteix a anular de forma aleatòria algunes neurones d'una determinada capa (les seves connexions d'entrada i sortida s'ignoren), per cada imatge de l'entrenament durant el *forward-pass*. Simbòlicament, aquesta tècnica és equivalent a entrenar múltiples CNNs i obtenir-ne la mitjana [23].
- **Batch Normalization:** Es tracta d'una tècnica de normalització que evita haver d'utilitzar la tècnica de *mean subtraction* (normalització de les imatges d'entrada), i a la vegada té un efecte de regularització tan positiu que fa que no es necessitin les capes de *dropout*. Consisteix a normalitzar els valors d'activació d'una determinada capa per cada exemple d'entrenament, tenint en compte els valors d'activació del minibatch [24] sencer. Amb tot això s'aconsegueix no només reduir el risc d'overfitting i augmentar la robustesa del model, sinó també fer que el temps d'entrenament sigui menor. La *batch normalization* es sol aplicar just abans de la funció *ReLU*.

La figura 2.13 mostra exemples de overfitting i underfitting en la tasca de classificació binària (amb 2 característiques). L'objectiu final és aconseguir entrenar una Xarxa Neuronal

Convolucional de manera que doni bons resultats tant amb el conjunt d'aprenentatge (no cal que siguin perfectes) com amb el de validació.

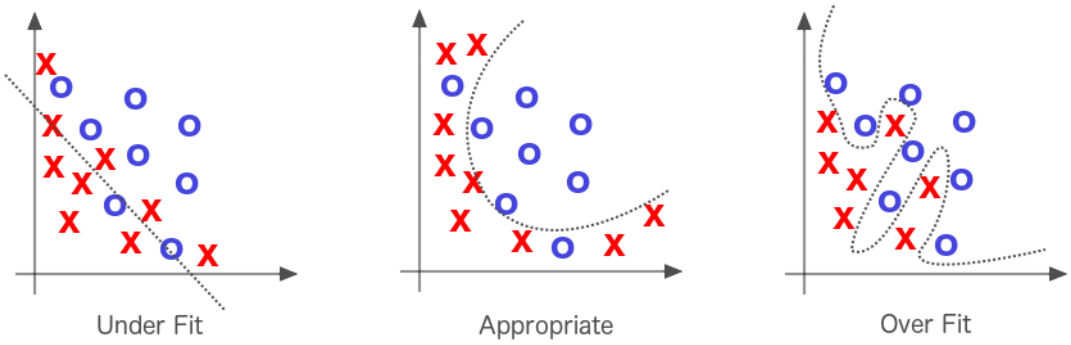

Figura 2.13: Exemple gràfic d'un model amb underfitting, òptim, i amb overfitting (d'esquerra a dreta), suposant que es tracta d'un classificador binari basat en 2 característiques.

2.5 *Linear Support Vector Machine*

Es tracta d'un conjunt d'algorismes que permet resoldre el problema de classificació: donat un vector de característiques, és capaç de predir a quina de dues classes pertany amb major probabilitat. Per fer-ho, prèviament es realitza la fase d'entrenament: s'extreuen els descriptors de característiques corresponents a les imatges d'entrenament, i es donen com a entrada a la *Support Vector Machine* (SVM) juntament amb l'etiqueta de la classe de cadascun. Amb aquestes dades, la SVM intenta trobar l'hiperplà que millor divideixi les característiques d'ambdues classes. Per fer-ho, hi ha dos objectius principals: d'una banda, maximitzar la distància fins cada classe (marge); d'altra banda, separar correctament el major número d'instàncies possible. Pot ser necessari prioritzar més un objectiu que l'altre, i s'utilitza l'anomenat regularization weight (C) per determinar quina de les dues condicions té més pes [12].

L'hiperplà tindrà tantes dimensions com la mida dels descriptors. Per exemple, si només s'extraguessin 2 característiques per imatge, l'hiperplà seria en realitat una recta. A la figura 2.14 es pot veure un exemple en què es compleixen satisfactòriament els 2 objectius. En canvi, a la figura 2.15 s'escenifiquen dues situacions en què el paràmetre de regularització C és determinant en el càlcul de l'hiperplà.

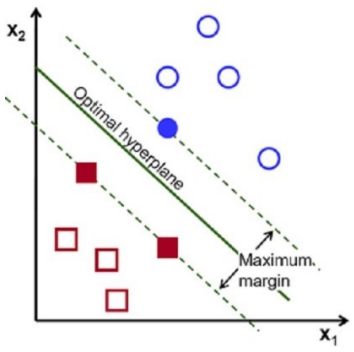

Figura 2.14: Exemple d'hiperplà òptim de 2 dimensions que divideix perfectament dos conjunts de característiques per crear un classificador lineal binari SVM.

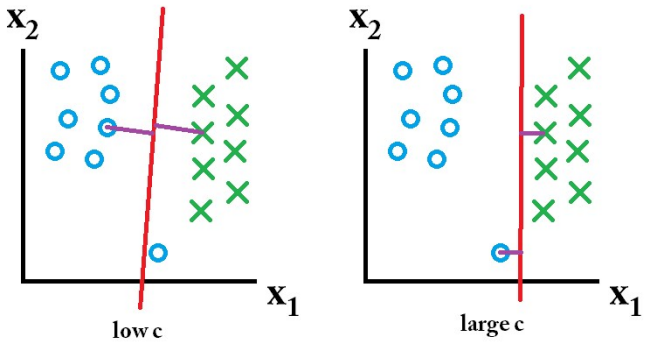

Figura 2.15: Variacions de l'hiperplà de 2 dimensions calculat durant l'entrenament del classificador lineal binari SVM en funció de C.

Una vegada obtinguda la frontera entre ambdues classes, per fer la predicció d'una imatge simplement s'han d'extreure les seves característiques, donar-les com a entrada a la SVM i veure a quina de les dues bandes cauen.

Els classificadors SVM són per tant classificadors binaris, ja que permeten separar dues classes. Tanmateix, són de gran utilitat també com a classificadors de més de dues classes. Per fer-ho, l'esquema més utilitzat és el *one-vs-one*. Si hi ha N classes, s'ha de crear un classificador SVM binari per cada parella de classes, per tant, $N(N-1)/2$ en total. Aleshores, cadascun dels classificadors binaris farà una predicció per una mateixa imatge d'entrada, i la predicció final serà la classe que hagi sigut predita més vegades.

Capítol 3

Desenvolupament del projecte

El sistema de reconeixement facial consta de diverses seccions ben diferenciades. Primerament s'ha de generar un conjunt de dades d'aprenentatge i validació. Posteriorment, s'ha de crear un detector de cares, cadascuna de les quals ha de passar per un procés d'alineament i retallat, obtenint així imatges de 96x96 píxels que conten cares amb els trets facials més destacats en una posició fixa. Després d'aquesta fase de preprocessament ve la de classificació d'actors, és a dir, la creació d'un conjunt d'algorismes que donada una cara preprocessada prediguin a quina classe (actor) pertany. S'han implementat dues alternatives per la detecció de cares i quatre per la classificació, amb l'objectiu de trobar la solució més fiable pel problema proposat. La millor combinació d'algorismes s'utilitzarà per crear el sistema de reconeixement d'actors en imatge, el qual s'utilitzarà en el reconeixement en vídeo. La figura 3.1 representa l'esquema complet del sistema de reconeixement d'actors en vídeo, així com la relació entre els mòduls que el formen.

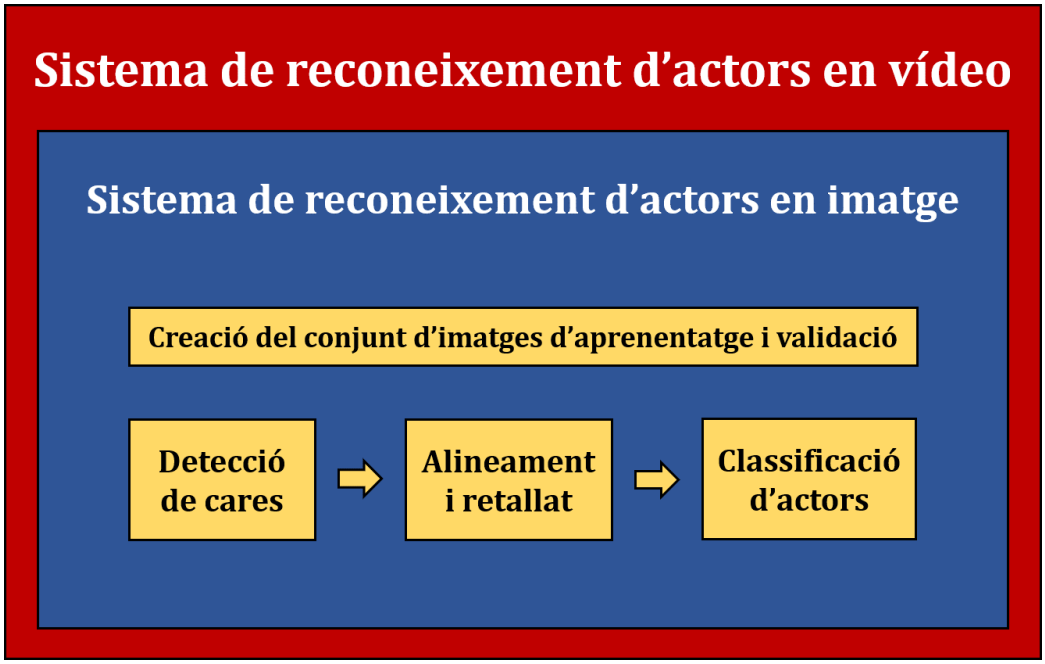

Figura 3.1: Esquema dels mòduls del sistema de reconeixement d'actors en vídeo.

La figura 3.2 escenifica gràficament un exemple de com serà el procés de reconeixement d'actors en imatge. Els passos marcats són 1) detecció de cares, 2) alineament i retallat, 3 classificació.

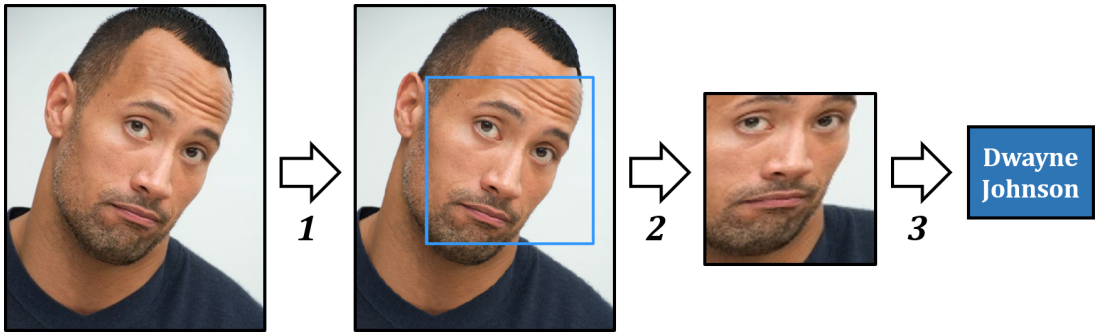

Figura 3.2: Exemple gràfic del procés de reconeixement d'actors en imatge.

Totes les tasques d'aquest projecte tenen un denominador comú: l'Aprenentatge Automàtic. És una branca de la Intel·ligència Artificial que, aplicant diferents algorismes, és capaç de resoldre problemes a priori complicats com són establir patrons en una base de dades i realitzar prediccions, per mitjà de l'aprenentatge.

A continuació es descriuen les diferents seccions d'aquest projecte, explicant el corresponent mètode, mostrant els resultats obtinguts i discutint-los, fent-ne una valoració.

3.1 Conjunt de dades d'aprenentatge i validació

S'ha decidit que el sistema sigui capaç de reconèixer els 70 actors que es mencionen a continuació.

1. Aaron Paul
2. Adam Sandler
3. Alexandra Daddario
4. Amy Adams
5. Angelina Jolie
6. Arnold Schwarzenegger
7. Ben Affleck
8. Benedict Cumberbatch
9. Ben Stiller
10. Bradley Cooper
11. Brad Pitt
12. Bruce Willis
13. Bryan Cranston
14. Cameron Diaz
15. Cara Delevigne
16. Channing Tatum
17. Christopher Walken
18. Christoph Waltz
19. Claire Danes
20. Dakota Johnson
21. Daniel Craig
22. Daniel Radcliffe
23. Denzel Washington
24. Dwayne Johnson
25. Ellen Page
26. Emma Stone
27. Emma Watson
28. George Clooney
29. Haley Joel Osment
30. Harrison Ford
31. Jackie Chan
32. Jake Gyllenhaal
33. James Franco
34. January Jones
35. Jason Statham
36. Jennifer Aniston
37. Jennifer Lawrence
38. Joseph Gordon-Levitt
39. Julianne Moore
40. Julia Roberts
41. Kate Mara
42. Kevin Costner
43. Kevin Spacey
44. Kristen Bell
45. Laurence Fishburne
46. Leonardo DiCaprio
47. Liam Neeson
48. Margot Robbie
49. Mark Ruffalo
50. Matt Damon
51. Meryl Streep
52. Miles Teller
53. Morgan Freeman
54. Nicolas Cage
55. Nicole Kidman
56. Octavia Spencer
57. Penelope Cruz
58. Robert De Niro
59. Robert Downey Jr
60. Ryan Gosling
61. Sandra Bullock
62. Scarlett Johansson
63. Sylvester Stallone
64. Tom Cruise
65. Tom Hanks
66. Viggo Mortensen
67. Vin Diesel
68. Will Smith
69. Zac Efron
70. Zach Galifianakis

Per la resolució del problema és necessari disposar d'un conjunt d'imatges d'actors i actrius, amb el qual fer les tasques d'entrenament i validació. En total s'han seleccionat 45 imatges per cada actor, formant un conjunt de dades total de 3150 imatges. Aquestes han estat extretes de *Google Images* [25], això si amb una sèrie de restriccions. Per tal de simplificar el procés d'implementació i avaluació de resultats, cada imatge (a color) ha de contenir un únic actor, amb la cara completament visible (a excepció d'ulleres o petites oclusions), mirant cap a la càmera o amb petites rotacions (lleugerament de perfil, mentó aixecat, cap torçat...), i ha de tenir una resolució suficientment alta com perquè la cara retallada tingui una mida superior a 96x96 píxels i així no es perdi definició al retallar-la. També s'ha d'intentar que les imatges siguin diverses, amb variacions d'il·luminació, punt de vista, perruqueria, maquillatge, accessoris, edat, expressions facials... Per aconseguir-ho, s'ha intentat seleccionar el màxim número d'imatges procedents d'escenes de pel·lícules, però tot i això no he pogut evitar que la majoria siguin de sessions de fotos o festivals de cinema, predominants a *Google Images*. Com més variades siguin, millors resultats s'obtindran. Les figures 3.3 i 3.4 contenen una petita mostra d'algunes de les imatges sense preprocessar utilitzades per les classes Brad Pitt i Scarlett Johansson respectivament.

Figura 3.3: Algunes imatges recopilades de l'actor Brad Pitt (sense preprocessar).

Figura 3.4: Algunes imatges recopilades de l'actriu Scarlett Johansson (sense preprocessar).

Una vegada obtingut el conjunt d'imatges, s'ha dividit aleatòriament en dos subconjunts: el d'entrenament serà utilitzat per crear els sistemes de reconeixement d'actors, mentre que el de validació tindrà l'objectiu d'avaluar-los, contribuint a la cerca dels millors algorismes i paràmetres per la resolució d'aquest problema. El conjunt d'aprenentatge ha de ser notablement més gran que el de validació, així que s'ha decidit que les proporcions siguin 8/9 i 1/9 respectivament. Com que el conjunt d'aprenentatge és bastant reduït (2800 imatges), s'ha utilitzat la tècnica de *data augmentation* per ampliar-lo i reduir així el risc d'overfit-

ting. L'única transformació possible és la simetria horitzontal (la rotació no serveix ja que el sistema de reconeixement funcionarà només amb cares lleugerament rotades, i l'escalat o translació tampoc ja que al retallar les cares no hi hauria cap diferència respecte la imatge original). Per tant, el conjunt d'aprenentatge duplica la seva mida i passa a ser de 5600 imatges. Al conjunt de validació no se li ha aplicat aquesta tècnica.

Finalment, destacar la importància que les imatges de validació siguin independents de les d'aprenentatge. Altrament, s'obtindrà una exactitud superior a la veritable. S'ha intentat que les 3150 imatges siguin úniques, però és inevitable que alguna estigui duplicada. En tot cas, estimo que el marge d'error és inferior al 2%.

3.2 Detecció de cares

Aquesta tasca és determinant en el desenvolupament del projecte, i és si els resultats són dolents pot suposar un coll d'ampolla per tot el sistema de reconeixement. Donada una imatge, s'han de detectar totes les cares que hi apareguin, sempre i quan compleixin els requisits establerts a l'abast del projecte. Per tant, s'hauran de detectar únicament les cares que estiguin orientades cap a la càmera, o bé amb lleugeres desviacions i rotacions, i guardar-ne les dades de les *bounding boxes* (es retallaran més endavant).

3.2.1 Mètode

S'han implementat dos algorismes diferents per poder comparar-los i escollir el que millors resultats proporcioni. El primer està basat en les característiques de Haar, mentre que el segon fa ús dels Histogrames de Gradients Orientats. Els dos conceptes s'han definit al marc teòric, i a continuació s'explica el seu paper en la detecció de cares.

3.2.1.1 Algorisme de Viola-Jones

Aquest algorisme va ser publicat l'any 2001 per Paul Viola i Michael Jones, i està basat en les característiques de Haar [6]. És un algorisme que tot i el pas dels anys, segueix sent una gran contribució en l'actualitat. El seu funcionament és el següent.

Per obtenir una gran quantitat de característiques de Haar en una sola imatge, es poden aplicar molts filtres en diferents posicions d'una imatge i amb diferents escales, aconseguint així un descriptor. Únicament utilitzant els filtres de Haar bàsics en una imatge de 24x24px es poden extreure més de 160000 característiques. Això pot ser molt costós, i per accelerar el procés s'utilitza la imatge integral, on cada posició conté la suma dels valors dels píxels ubicats en el rectangle superior esquerre, permetent minimitzar el número d'operacions. Tot i això, segueix havent-hi un gran marge de millora, donat que la majoria de les característiques són irrellevants.

Adaboost és un algorisme que permet reduir el nombre de característiques, buscant les més rellevants. Per cada característica (un determinat filtre aplicat en una posició i amb una escala concreta), troba el llindar que millor classifiqui un conjunt d'imatges d'aprenentatge (format per imatges de cares retallades i imatges on no apareix cap cara). Amb una única característica és difícil obtenir uns resultats perfectes, així que hi haurà alguns casos de classificació errònia. Només es tindran en compte les característiques amb la mínima taxa d'error, i així s'obtindran les més rellevants (les que millor classifiquin les classes de cares i

no-cares). Per cadascuna d'elles juntament amb el corresponent llindar es crearà un *weak classifier*.

Tot i això, el nombre de característiques seguirà sent elevat. Per accelerar el procés de predicció, s'utilitza una cascada de *weak classifiers*. Consisteix a agrupar-los i organitzar-los en diferents grups (*hard classifiers*), ubicat un darrere de l'altre. El conjunt de *hard classifiers* formarà el classificador final.

Una vegada entrenat el classificador, la tasca de detecció funciona de la següent manera. En primer lloc, es fan lliscar finestres de diferents mides per la imatge original. Per cada finestra, s'apliquen els *hard classifiers* de forma iterativa. El resultat de cadascun d'ells serà favorable únicament si tots els *weak classifiers* que el formen prediuen que és una cara. A la vegada, la predicció final serà únicament favorable si es passen tots els *hard classifiers* amb predicció positiva. Quan un dóna una predicció negativa, es torna a començar amb una nova finestra. Això fa que sigui un algorisme molt eficient. Quan finalment es tenen totes les finestres que han donat positiu, es dóna la situació que s'obtenen vàries *bounding boxes* per una mateixa cara. Per convertir-les en una de sola, es sol aplicar l'algorisme de supressió de no màxims, que consisteix bàsicament en ignorar totes les que es solapen significativament menys una (veure figura 3.5). Un algorisme alternatiu seria fer la mitjana de totes les *bounding boxes*, tot i que això no donaria tan bons resultats.

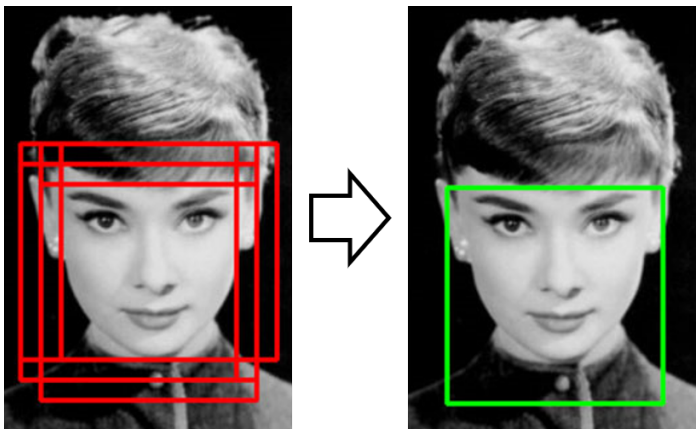

Figura 3.5: Procés de fusió de totes les *bounding boxes* que pertanyen a una mateixa cara, mitjançant un algorisme de supressió de no màxims.

Comentaris sobre la implementació: He utilitzat un detector de cares ja entrenat amb Viola-Jones que ofereix la llibreria OpenCV de Python [26].

3.2.1.2 Algorisme amb Histogrames de Gradients orientats

Aquest algorisme és posterior (any 2005), i és una adaptació de la detecció de cossos humans ideada per Navneet Dalal i Bill Triggs [7]. A continuació s'explicarà el mateix mètode però aplicat a la detecció de cares, que com el de Viola-Jones consta de dues fases: l'entrenament i la detecció.

Per fer l'entrenament, s'ha de disposar d'un conjunt d'aprenentatge format per un subconjunt d'imatges de cares retallades, i un altre d'imatges que no continguin cap cara (remarcar que aquestes han de ser més nombroses que les primeres). Aquest conjunt té les mateixes característiques que l'utilitzat a l'algorisme de Viola-Jones. Primerament, s'extreuen els descriptors d'Histogrames de Gradients orientats de totes les imatges d'aprenentatge i, amb aquests i la corresponent etiqueta (cara o no-cara), s'entrena un classificador lineal binari SVM (*Linear Support Vector Machine*). Després d'això ja es té un classificador que, donant-li com a entrada el descriptor de HoG d'una imatge, seria capaç de predir si pertany a la classe de cares o no-cares. Tanmateix, per augmentar la fiabilitat del classificador, s'aplica una tècnica que es coneix com *hard-negative mining*. Per cada imatge del subconjunt de no-cares d'aprenentatge, s'aplica una finestra lliscant amb diferents mides. Per cada finestra, s'extreu el descriptor de HoG i s'utilitza el classificador per fer una predicció. Si prediu que es tracta d'una cara (fals positiu), es guarda el descriptor juntament amb el percentatge de confidència. Després d'aquest procés s'obté un conjunt de mostres de falsos positius. S'ordenen per probabilitat i es reentrena el classificador SVM amb els nous descriptors i etiquetes (negatives). El classificador resultant serà molt més exacte, i ja no donarà tants falsos positius [27].

Per detectar les cares en una imatge, es fa lliscar una finestra de diferents escales. Per cadascuna d'elles s'extreu el descriptor de HoG, es passa al classificador i, si la predicció és positiva (per sobre d'una determinada probabilitat), es marca la finestra com a possible. Al finalitzar, com amb l'algorisme de Viola-Jones, s'haurà de fer una supressió de no màxims, que permet obtenir una única *bounding box* per cada cara (veure exemple a la figura 3.5).

Comentaris sobre la implementació: He utilitzat un detector de cares ja entrenat amb Histogrames de Gradients orientats que ofereix la llibreria Dlib de Python [28].

3.2.2 Resultats

S'han aplicat ambdós algorismes de detecció de cares a les 2800 imatges d'aprenentatge seleccionades per aquest treball (secció 3.1), ja que és un conjunt de dades més gran de validació, i cap dels dos algorismes requereix una fase d'entrenament donat que els classificadors ja estan entrenats per OpenCV i Dlib. La sortida d'aplicar l'algorisme no és binària (cara o no-cara), sinó que és la mateixa imatge d'entrada però amb les cares detectades emmarcades amb una *bounding box*. Tenint en compte que cada imatge d'aprenentatge conté una única cara, la sortida de cada imatge s'ha analitzat manualment i s'ha dividit en 4 casos possibles: 1 cara correcta (una única *bounding box* emmarca la cara correctament), 0 cares (no s'ha detectat cap cara), 1 cara incorrecta (només s'ha detectat una cara i és incorrecta), 2 cares (s'han fet dues deteccions, una correcta i l'altra incorrecta). En cap cas s'han detectat 2 o més cares incorrectament en una mateixa imatge. Els resultats de cada algorisme es mostren a la taula 3.1.

	Algorisme de detecció de cares	
	Viola-Jones	**HoG**
1 cara (correcta)	2634	2789
0 cares	107	9
1 cara (incorrecta)	8	0
2 cares (correcta + incorrecta)	51	2

Taula 3.1: Resultats sense processar de la detecció de cares amb els algorismes de Viola-Jones i HoG.

Com que en aquest test no s'han pogut comptar els veritables negatius, ja que es tracta d'una detecció amb finestra lliscant, no es pot donar un valor d'exactitud (*accuracy*). Tanmateix, existeix una altra mesura estadística que resulta de gran utilitat en aquest problema: la F_1*-score*. Aquesta té en compte tant la precisió (*precision*) com la sensibilitat/exhaustivitat (*recall*), calculant la mitjana harmònica d'ambdues mesures. La precisió indica quin percentatge d'objectes seleccionats són cares, mentre que la sensibilitat fa referència al percentatge de cares detectades respecte el total de cares existents. A la taula 3.2 es mostren els resultats obtinguts.

	Algorisme de detecció de cares	
	Viola-Jones	**HoG**
Veritables positius	2685	2791
Falsos positius	59	2
Falsos negatius	115	9
Precision	97.85 %	99.93 %
Recall	95.89 %	99.68 %
F_1-score	**96.86 %**	**99.80 %**

Taula 3.2: Resultats processats de la detecció de cares amb els algorismes de Viola-Jones i HoG.

Finalment, s'ha calculat el temps de la detecció de cares per cada algorisme per un petit conjunt d'imatges de 2 resolucions diferents (20 imatges cadascun). Els resultats consten a la figura 3.3.

Resolució		**Algorisme de detecció de cares**	
		Viola-Jones	**HoG**
1280x720px	Temps mínim	40 ms	89 ms
	Temps màxim	67 ms	92 ms
	Temps mitjà	**49 ms**	**90 ms**
854x480px	Temps mínim	19 ms	42 ms
	Temps màxim	45 ms	43 ms
	Temps mitjà	**28 ms**	**42 ms**

Taula 3.3: Temps d'execució de la detecció de cares amb els algorismes de Viola-Jones i HoG.

A continuació es mostra visualment el rendiment dels dos algorismes en la detecció de cares (en vermell amb el de Viola-Jones, en blau amb el d'Histogrames de Gradients orientats). Les cares de la figura 3.6 s'han detectat correctament amb els dos algorismes. Les de la 3.7 només amb el de HoG (Viola-Jones no ha detectat res), les de la 3.8 han estat detectades correctament amb HoG mentre que amb Viola-Jones hi ha hagut falsos positius, i les de la 3.9 no s'han detectat amb cap dels dos algorismes. La figura 3.10 mostra les úniques 3 imatges de tot el dataset d'aprenentatge en què Viola-Jones ha obtingut millors resultats que HoG.

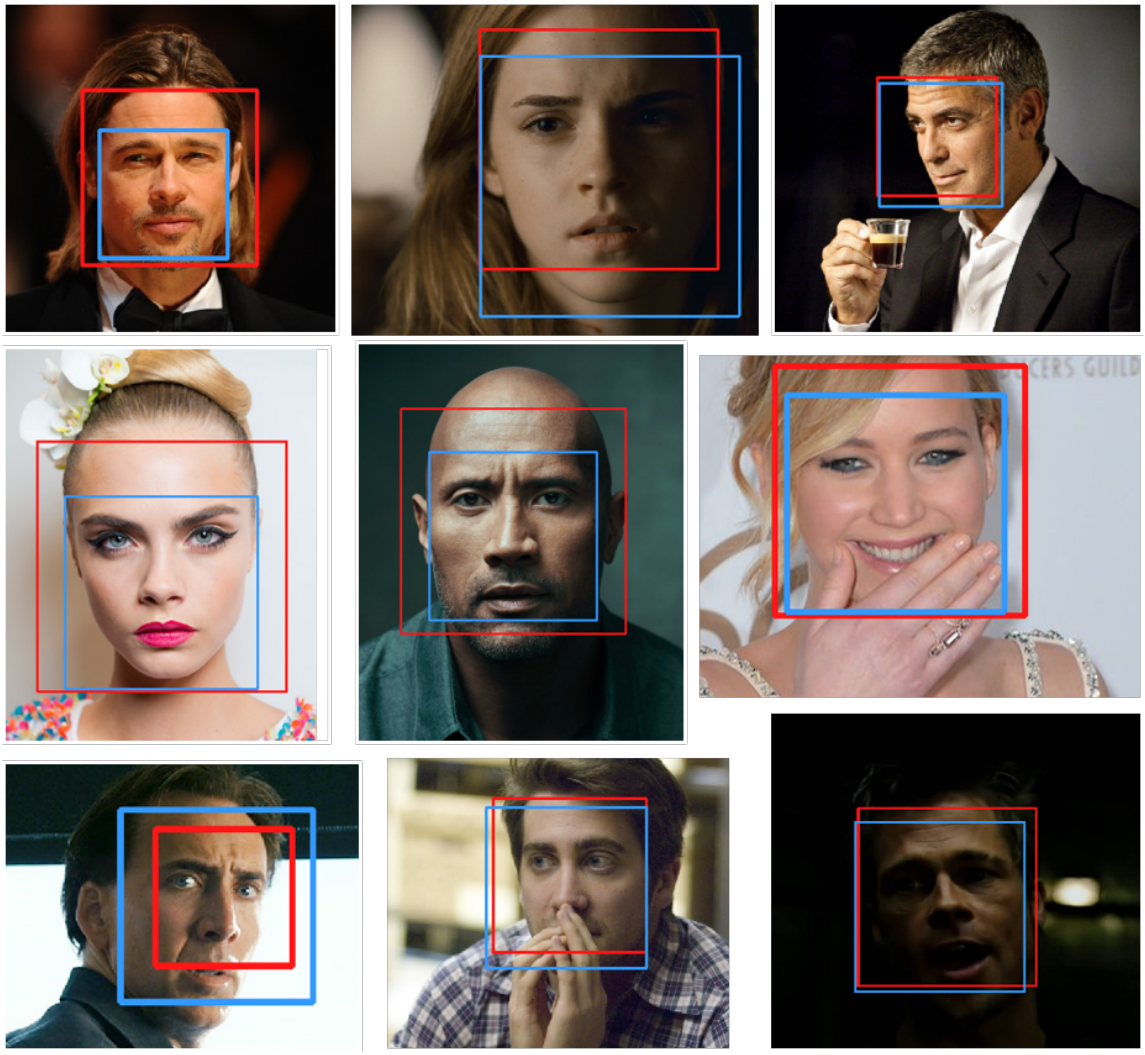

Figura 3.6: Deteccions facials correctes amb HoG i Viola-Jones.

Figura 3.7: Deteccions facials correctes amb HoG però no amb Viola-Jones (I).

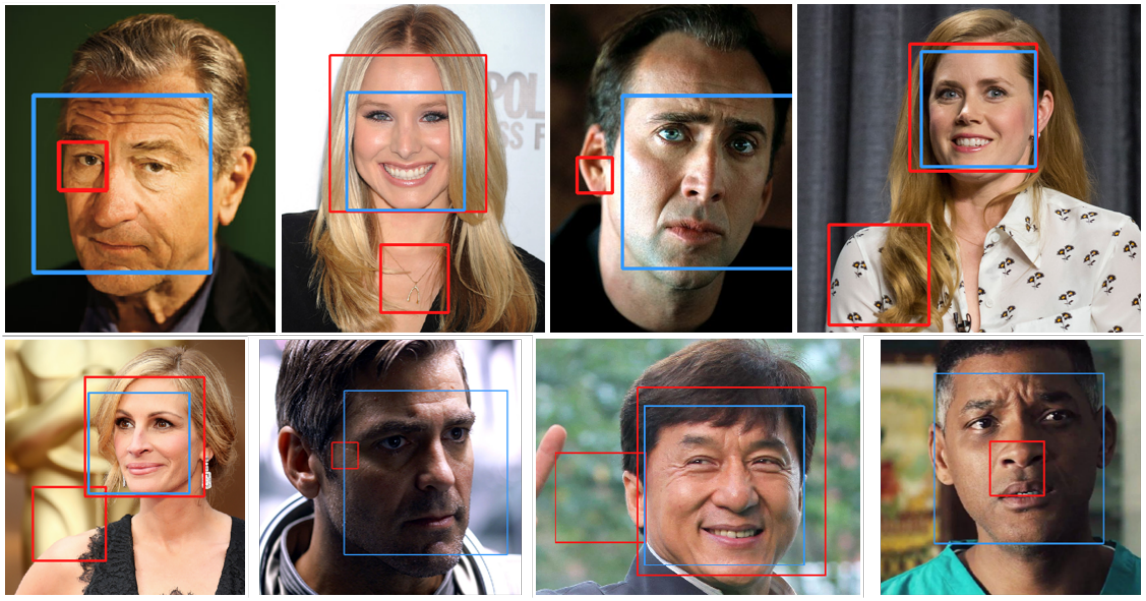

Figura 3.8: Deteccions facials correctes amb HoG però no amb Viola-Jones (II).

Figura 3.9: Cap cara detectada amb ambdós algorismes.

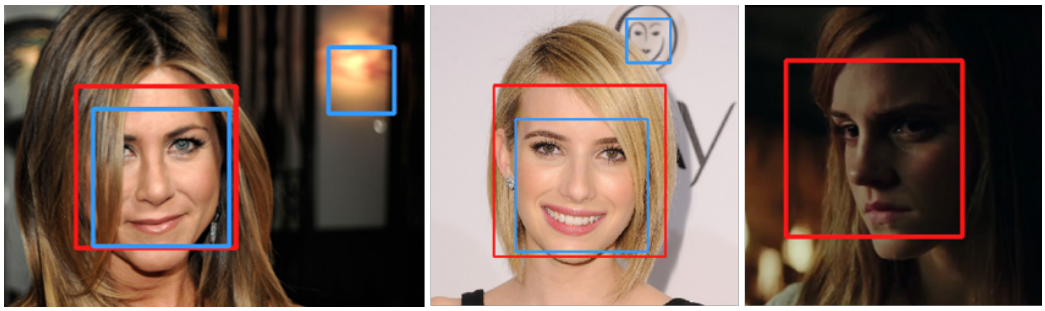

Figura 3.10: Únics casos en què Viola-Jones funciona millor que HoG com a detector facial.

3.2.3 Discussió

En primer lloc, destacar que les dues F_1-score són tan elevades ja que, com s'ha explicat a l'apartat 3.1, les imatges de la base de dades estan triades de manera que siguin fàcilment detectables (frontals o quasi frontals). Tot i això, dels resultats de la taula 3.2 es pot extreure que els dos algorismes són excel·lents en la detecció de cares, i la diferència en la F_1-score és de tan sols 2.94% a favor dels Histogrames de Gradients orientats. Respecte el temps de còmput, l'algorisme de Viola-Jones és de mitjana més ràpid, tot i que a causa de l'estructura en cascada del seu classificador, aquest temps és molt variable i depèn de la complexitat de les finestres lliscants de la imatge original i com de ràpid es puguin descartar, podent arribar a trigar més que el de HoG. Més endavant s'analitzarà si aquests temps són acceptables per un sistema de reconeixement facial en temps real.

Observant les mostres gràfiques de cares detectades es poden extreure una sèrie de conclusions: els dos algorismes són capaços de sortejar els problemes d'il·luminació i contrast, així com també petites oclusions. Tot i això, una primera diferència de rendiment que es pot apreciar a la figura 3.6 és que les *bounding boxes* obtingudes amb HoG s'ajusten millor

a les cares. A més, l'algorisme de Viola-Jones té més dificultats per detectar les cares de pell fosca i cares que estiguin lleugerament torçades, de perfil, o amb expressions facials estranyes, donant lloc a molts falsos negatius (veure figures 3.7 i 3.9). D'altra banda, també es caracteritza per donar molts falsos positius (veure figura 3.8), i en escenes extretes de pel·lícules aquesta debilitat encara es faria més evident donat que hi hauria més finestres a analitzar (a la base de dades d'aprenentatge la majoria de cares estan preses de prop i amb poc fons). És cert que amb els HoG hi ha alguns falsos positius i falsos negatius, però són molt infreqüents i anecdòtics: els 2 únics falsos positius es mostren a la figura 3.10, així com també l'únic fals negatiu que ha detectat correctament l'algorisme de Viola-Jones. La resta de falsos negatius de HoG (només 8) ho han sigut també amb Viola-Jones (veure un parell de mostres a la figura 3.9).

Per tot això, en aquest treball s'ha decidit utilitzar l'algorisme de detecció amb Histogrames de Gradients orientats, ja que un vídeo és tractat fotograma a fotograma i, per exemple, en una escena gravada a 30 fps de només 30 segons s'han d'analitzar 900 imatges. Segons els resultats obtinguts amb les imatges d'aprenentatge, el número de falsos negatius (cares no detectades) serien 37 i 3 en un cas i en l'altre, una diferència que visualment pot ser molt notòria i perjudicar el sistema de reconeixement. El número de falsos positius no es pot calcular ja que depèn del nombre d'objectes totals detectats, però en el cas de Viola-Jones seria bastant elevat, mentre que amb els Histogrames de Gradients orientats probablement no n'hi hauria cap.

Destacar que recentment s'han desenvolupat sistemes de detecció de cares que fan ús de les Xarxes Neuronals Convolucionals. Tanmateix, no s'han analitzat en aquesta primera fase ja que aquests dos algorismes ja donen uns resultats prou satisfactoris, i no seria adequat augmentar significativament el cost temporal d'aquesta fase per obtenir una ínfima millora. És cert que amb Xarxes Neuronals es podrien detectar cares amb orientacions més diverses (completament de perfil, mirant cap amunt...). Tanmateix, l'objectiu principal d'aquest treball és el reconeixement facial, i detectar les cares en orientacions estranyes empitjoraria els resultats dràsticament, ja que el procés d'alineament no seria possible en alguns casos (un ull no visible), i a la base de dades d'entrenament s'estarien barrejant imatges dels dos tipus. El que sí que es podria fer seria crear dos sistemes de reconeixement: un frontal i l'altre de perfil. Tanmateix, això seria temporalment costós i està fora de l'abast d'aquest treball.

3.3 Alineament i retallat de cares

El següent pas és l'alineament i retallat de cares, una tasca de preprocessament. L'entrada és una imatge juntament amb les dades de les *bounding boxes* de les cares que s'hi ha detectat. L'objectiu és, per cada cara detectada, generar una imatge de 96x96 píxels que contingui la cara de manera que els punts externs dels ulls i la part inferior del nas estiguin ubicats sempre en la mateixa posició de la imatge. Per fer-ho, és imprescindible detectar-ne els trets facials, una tasca que es durà a terme mitjançant arbres de regressió i tècniques d'Aprenentatge Automàtic.

3.3.1 Mètode

El procés que es vol aplicar consisteix en detectar primer els trets facials, i després aplicar una fase de transformació i retallat (per cada cara). Els dos procediments es descriuen a continuació, i s'hi s'aplica el concepte de *ferns* descrit al marc teòric. Per simplificar l'explicació, es suposarà que en una imatge només s'hi ha detectat una cara. La figura 3.11 descriu el procediment utilitzat, que consta de 3 fases: 1) detecció dels trets facials (3 en aquest cas simplificat) de la cara marcada amb la *bounding box*, 2) transformació de la imatge per alinear els trets facials, 3) retallat a mida de 96x96px al voltant dels trets facials.

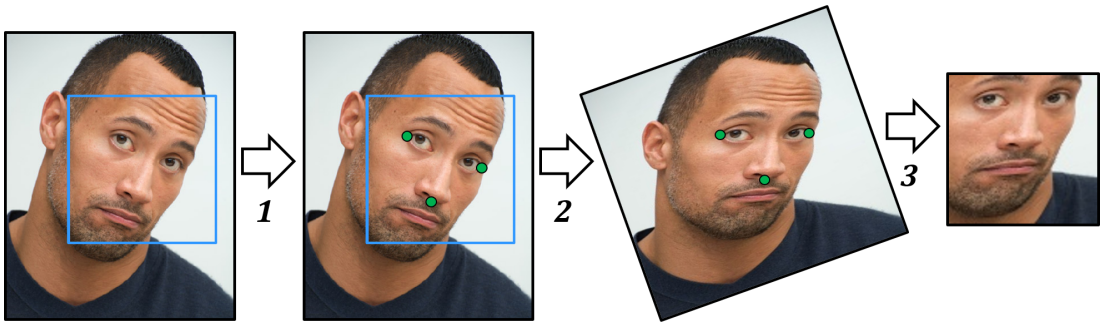

Figura 3.11: Esquema gràfic del procés d'alineament i retallat d'una cara, tenint la seva *bounding box* com a entrada.

3.3.1.1 Detecció dels trets facials

A la figura 3.12 hi ha exemples de la representació gràfica de 194 trets facials.

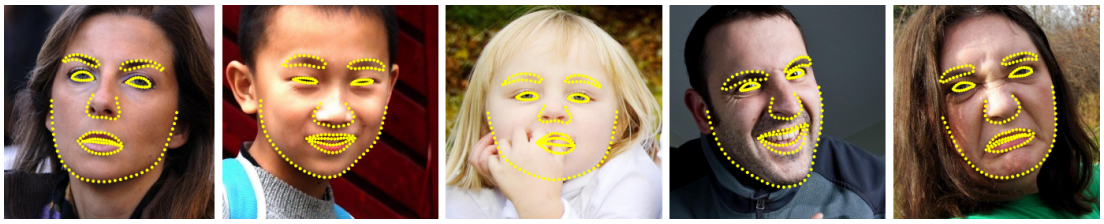

Figura 3.12: 194 trets facials representats en algunes cares de mostra. Seran els trets utilitzats en l'explicació de l'algorisme.

Tanmateix, en aquest projecte es detectaran únicament 68 trets facials (veure posicions a la figura 3.13), ja que així està implementat a la llibreria Dlib. Les imatges amb 194 trets facials només es mostren a mode d'explicació del mètode, i no són outputs d'aquest projecte.

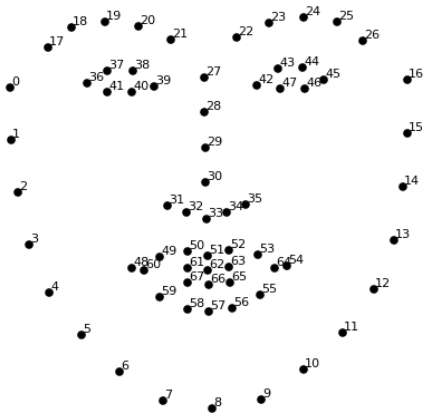

Figura 3.13: Distribució dels 68 trets facials que es detectaran per tal d'alinear les cares.

Per aconseguir extreure els trets facials d'una cara, hi ha dues parts ben diferenciades: l'aprenentatge i la predicció. Per fer l'explicació més entenedora, es començarà descrivint com s'obtenen les ubicacions dels 68 trets facials, suposant que es disposa d'un model de predicció ja entrenat. Posteriorment, s'abordarà com s'entrena aquest model i quins paràmetres el componen. Destacar que es tracta d'un problema de regressió i no de classificació, ja que per cada tret facial es prediu una posició (en 2 dimensions), que és valor continu i no discret. Com s'ha comentat anteriorment, l'algorisme està basat en arbres de regressió [8].

Fase de predicció:

Els trets facials no s'obtenen del no-res, sinó que es comença amb una forma mitjana (*mean shape*) que es pot veure a la figura 3.14, i aquesta es va modelant iterativament fins que s'ajusta als trets facials de la cara en qüestió.

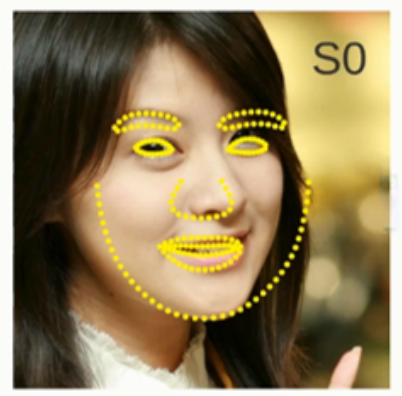

Figura 3.14: *Mean shape* S0 (ubicació dels trets facials a l'inici de la primera iteració).

Cada iteració segueix la mateixa estructura, així que fixant-se en un ja es pot generalitzar a la resta. A continuació, s'analitzarà la 2a iteració, per passar de S1 a S2, amb una lleugera millora de l'ajust dels trets facials a la cara (figura 3.15).

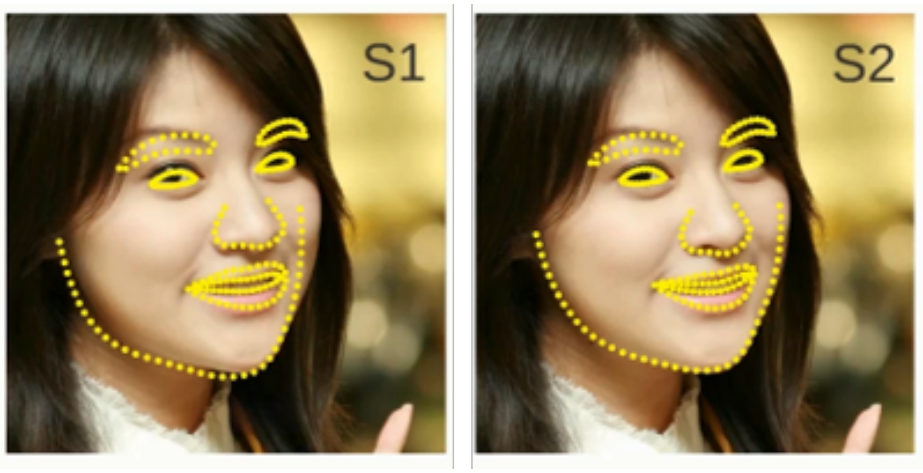

Figura 3.15: *Shape* S1 i S2 (ubicació dels trets facials a l'inici de la segona i tercera iteració respectivament).

- Primerament, s'extreuen 400 píxels. La posició d'aquests està emmagatzemada al model de predicció per cada una de les iteracions. Com que les seves posicions es donen en referència a la imatge de la *mean shape*, s'utilitza una transformada de

similaritat per veure com es trasllada cada píxel des de la S0 a la forma de l'actual iteració (S1). Això es mostra a la figura 3.16.

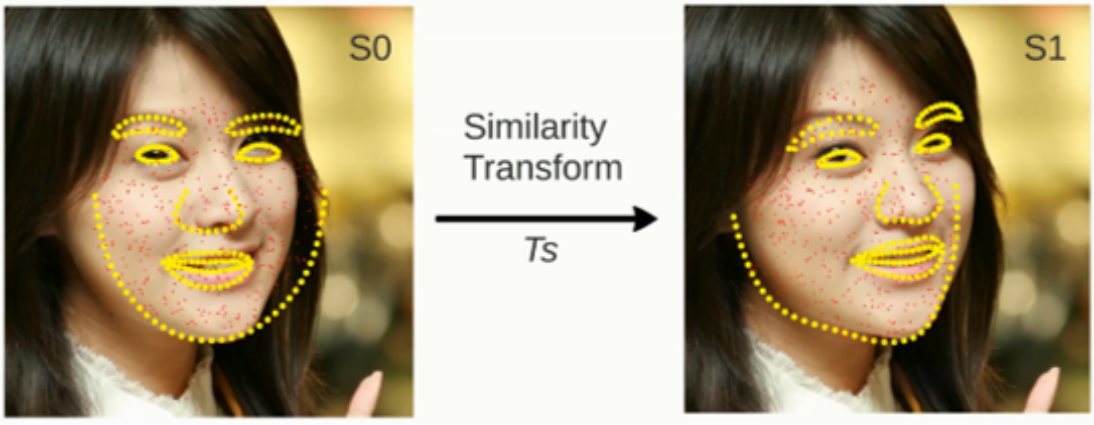

Figura 3.16: *Similarity Transform* per passar les ubicacions dels 400 píxels de la *shape* S0 a la S1.

- Una vegada es tenen les noves posicions dels 400 píxels, ajustades segons els trets facials actuals (S1), s'aplica un *regressor*, els paràmetres del qual es troben emmagatzemats al model de predicció. Cada iteració tindrà un *regressor* amb paràmetres diferents, cadascun dels quals permetrà passar de S0 a S1, de S1 a S2... a partir dels 400 píxels extrets a l'inici de cada iteració.
- Un *regressor* està format per una sèrie de *ferns*. Per cada *fern*, el test binari de cada nivell consisteix a comparar la diferència de la intensitat de 2 píxels (extrets del conjunt de 400 píxels) respecte un determinat threshold (veure figura 3.17). D'aquest manera, baixant nivells, s'acaba arribant a una fulla, que guarda informació sobre quant s'han de modificar les coordenades (x, y) de cadascun dels trets facials actuals ($\Delta landmarks$). Després d'haver-se aplicat aquestes modificacions, es segueix el mateix procés de forma iterativa amb la resta de ferns, cadascun dels quals contribuirà lleugerament en el desplaçament dels trets facials. Una vegada enllestits tots els *ferns*, s'haurà obtingut els trets facials de S2, i es pot seguir amb la selecció de píxels i aplicació del *regressor* de la següent iteració.

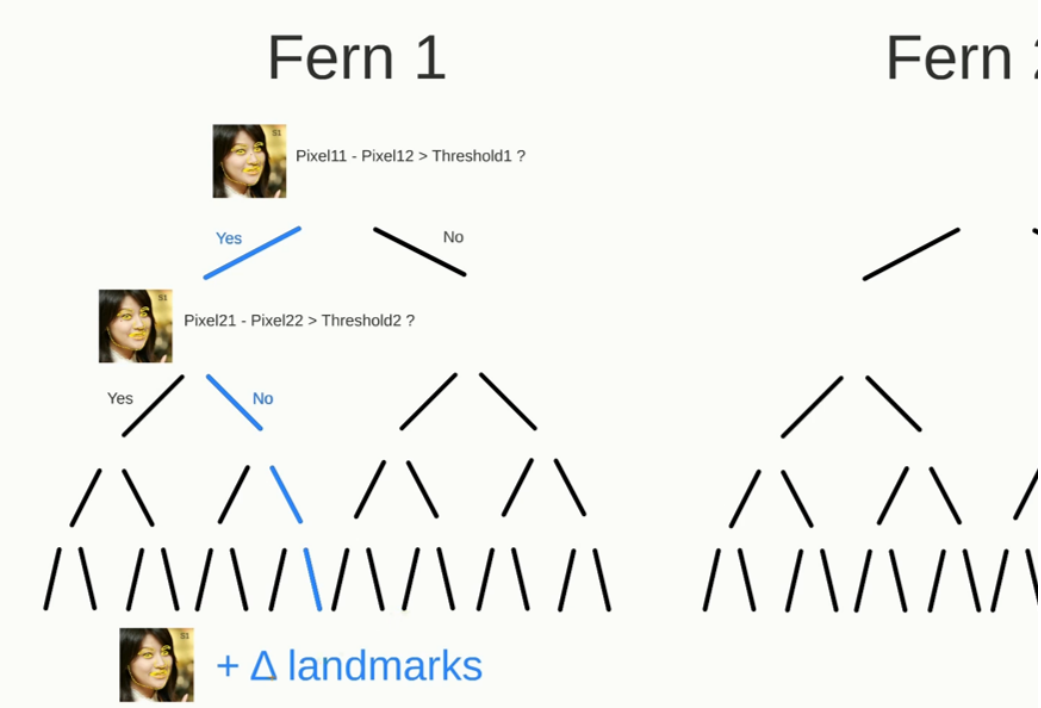

Figura 3.17: Exemple de l'estructura del primer *fern* de la iteració per passar de la *shape* S1 a la S2.

- En el predictor de trets facials utilitzat en aquest projecte, s'executen 10 iteracions. Per tant, hi ha 10 *regressors*, 500 *ferns* per cada *regressor*, i 5 nivells per cada *fern* (16 fulles). A la figura 3.18 es mostra la representació dels trets facials obtinguts després de la darrera iteració (S10). La figura 3.19 representa el procés complet per predir les seves ubicacions.

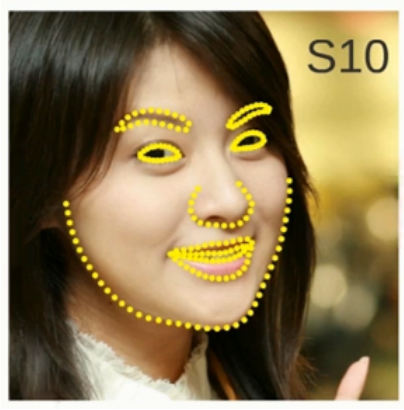

Figura 3.18: *Final shape* S10 (ubicació dels trets facials després de les 10 iteracions).

Figura 3.19: Procés complet del càlcul de les posicions dels trets facials.

Fase d'aprenentatge:

Aquest procés només cal realitzar-lo una vegada a l'inici, i després ja es poden fer infinites prediccions. L'objectiu és generar els paràmetres del model de predicció, que són els següents: la ubicació dels trets facials de la *mean shape* S0, la ubicació dels 400 píxels de cadascuna de les 10 iteracions (respecte la *mean shape*), les tripletes (2 identificadors de píxels i 1 threshold) necessàries per fer el test binari de cada nivell de cada *fern* del *regressor* de cada iteració, i el vector de $\Delta landmarks$ de mida 136 (coordenades xy de cada tret facial) per cada fulla de cada *fern* de cada *regressor*.

Per fer-ho, es disposa d'una conjunt d'imatges de cares (amb 2000 és suficient), a cadascuna de les quals l'acompanya un vector amb les posicions dels 68 trets facials, calculats manualment per un humà. Amb aquestes, es poden extreure tots els paràmetres que s'han esmentat a l'anterior paràgraf i que formen el model predictiu.

En primer lloc, per obtenir la *mean shape*, només cal fer el promig de les posicions dels trets facials de totes les imatges d'aprenentatge. D'altra banda, els 400 píxels de cada iteració es generen aleatòriament, dins del cercle que millor s'ajusta a la *mean shape*. Respecte els tests binaris de cada nivell d'un *fern*, les etiquetes dels 2 píxels es seleccionen aleatòriament del conjunt de 400, i el threshold també (dins d'un rang).

Ja només falta establir $\Delta landmarks$ per cada fulla. Per fer-ho, es passen les imatges d'aprenentatge pels *ferns* dels *regressors* de forma iterativa, resolent les preguntes binàries de cada nivell fins arribar a les fulles (només es passa al següent *fern* quan totes les imatges han passat per l'actual). Les imatges es distribuiran automàticament entre les 16 fulles, en funció de les intensitats dels píxels escollits. Aleshores, per cada fulla, es calcula la diferència del vector de posicions dels trets facials actuals respecte els teòrics de cadascuna de les imatges que hi ha caigut, se'n fa la mitjana i es multiplica per un factor d'amortiment (per evitar oscil·lacions). Quan s'ha acabat amb el primer *fern*, es modifiquen les coordenades dels trets facials amb el corresponent $\Delta landmarks$ calculat per cada fulla i es passa al següent, fins completar tots els *ferns* de totes les iteracions.

Comentaris sobre la implementació: Dlib (llibreria de Python), ofereix un model predictiu ja entrenat seguint el mètode descrit a la fase d'aprenentatge, que permet calcular la posició dels 68 trets facials de qualsevol cara [29].

3.3.1.2 Transformació i retallat de la imatge

Una vegada estan localitzats els trets facials dins de la *bounding box* de la cara detectada en una imatge, es transforma la imatge sencera, es retalla ajustant-la als trets facials i es redimensiona a una mida de 96x96 píxels, de manera que els punts externs dels ulls i la part inferior del nas estiguin ubicats en la mateixa posició en totes les imatges (veure els trets facials concrets a la figura 3.20), de manera que el sistema de reconeixement facial pugui reconèixer satisfactòriament cares que estiguin lleugerament de perfil o inclinades. L'alineament també es podria fer amb altres trets facials, com ara els punts interns dels ulls i la part inferior dels llavis. Tot i això, hi ha estudis que recullen que és la tècnica d'alineament que millor funciona [30].

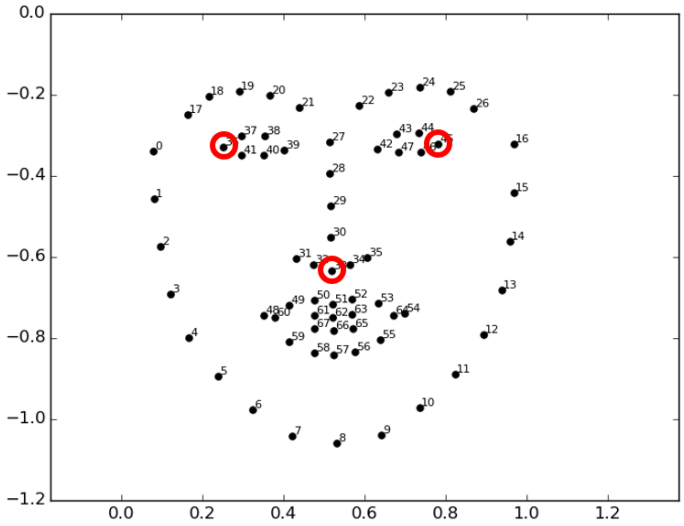

Figura 3.20: Localització dels 3 trets facials que s'utilitzaran per alinear una cara.

Centrant l'atenció en el procés d'alineament, s'utilitzen transformacions afins com són translacions, escalats, rotacions i shear mapping. Amb aquestes es preserven les línies paral·leles. Es podria fer també amb transformacions de 3 dimensions, però aquestes encara estan en desenvolupament [31], podrien provocar distorsions i un temps de còmput massa elevat. A més, només entra dins l'abast del projecte detectar cares lleugerament rotades.

Comentaris sobre la implementació: Per l'alineament i retallat d'una cara he utilitzat una funció implementada per Openface [32], que utilitza Dlib per extreure els 68 trets facials (explicat a l'anterior apartat) i OpenCV per transformar la imatge i retallar-la [33].

3.3.2 Resultats

La figura 3.21 mostra la sortida i el procés d'alineament i retallat per algunes cares, marcant en blau la *bounding box* de la cara i en verd els seus trets facials.

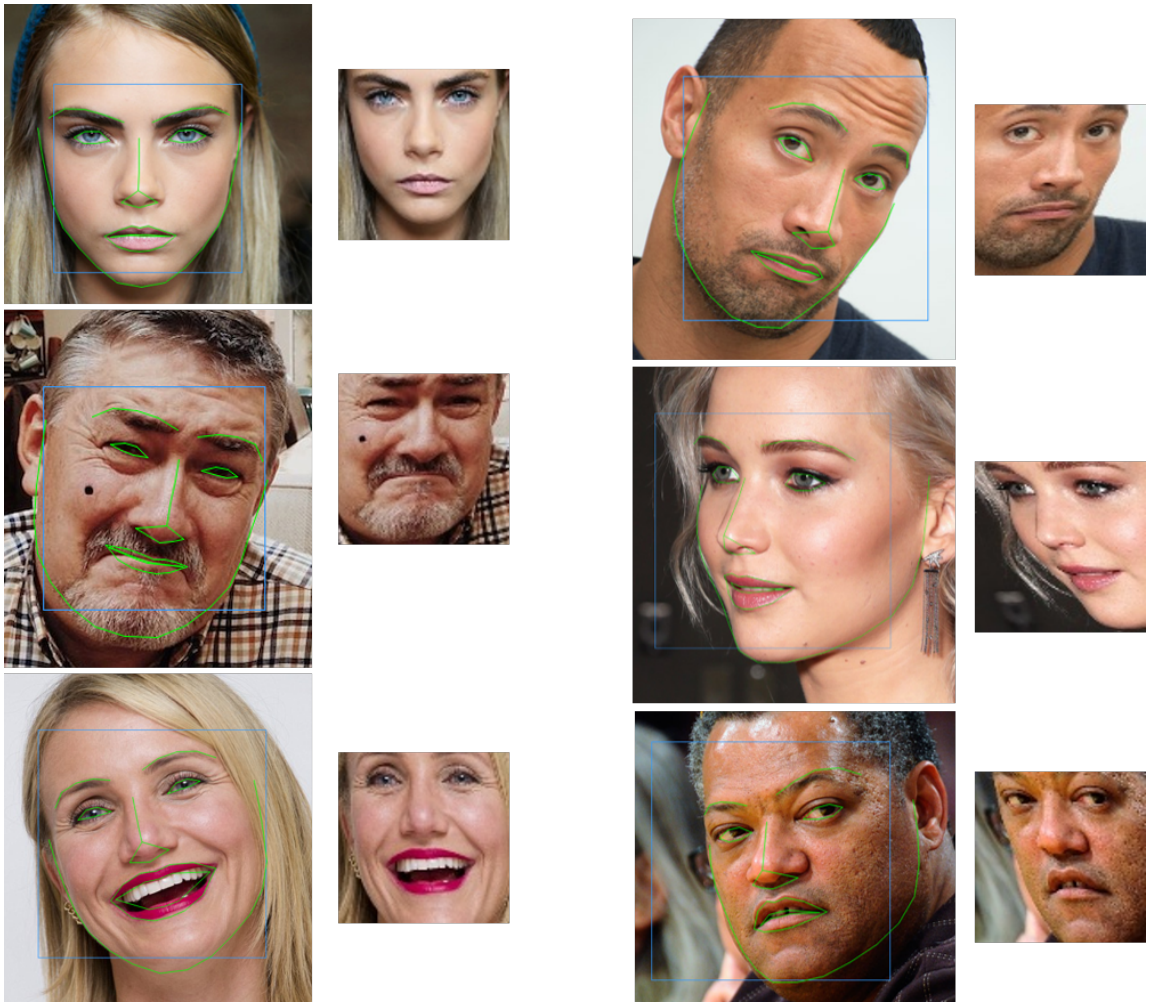

Figura 3.21: Procés d'alineament i retallat aplicat a imatges de mostra (amb la *bounding box* i els trets facials ja detectats).

És impossible avaluar les prediccions de regressió amb una mesura d'exactitud, ja que són prediccions contínues. En canvi, sí que es podria calcular el valor de *root mean squared error*. Tanmateix, això és molt difícil ja que no es disposa de les posicions teòriques de cadascun dels trets facials, una tasca que seria molt costosa i fora de l'abast d'aquest projecte. A simple vista, fent un anàlisi subjectiu dels trets facials extrets del conjunt d'imatges d'aprenentatge, els resultats són molt satisfactoris, ja que els trets facials s'ajusten correctament a totes les cares, i encara que en alguns casos hi ha petites imprecisions,

aquestes afecten marginalment a l'alineament i retallat de les cares. Una altra observació és que les cares inclinades es corregeixen a la perfecció. En canvi, les cares que estan molt de perfil, tenen difícil solució. Caldrà veure com afecta això a la fase de classificació. La figura 3.22 mostra algunes de les cares alineades i retallades de l'actriu Julia Roberts.

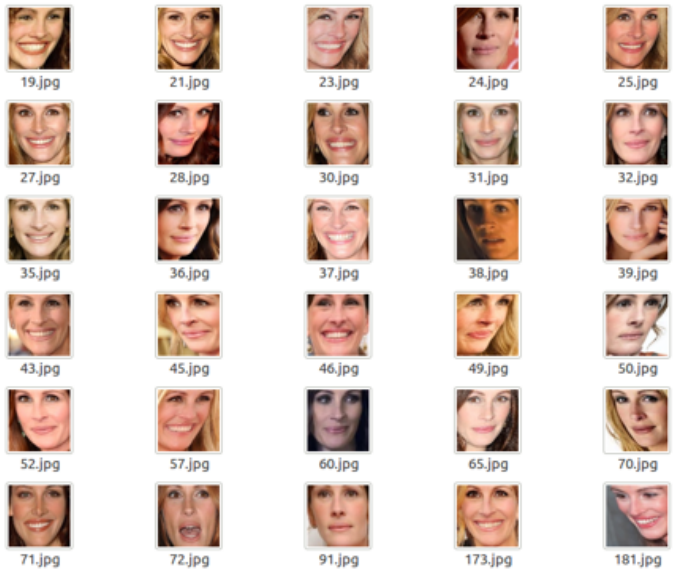

Figura 3.22: Algunes cares alineades de l'actriu Julia Roberts, que s'utilitzaran per la tasca de classificació.

Finalment, destacar que el temps d'execució és d'uns 2 ms de mitjana, així que es tracta d'un algorisme molt eficient.

3.3.3 Discussió

Com es pot extreure dels resultats, l'alineament de cares és un procés ràpid i de gran utilitat, que permetrà al sistema de reconeixement guanyar fiabilitat en els casos en què les cares no estiguin completament rectes ni orientades directament a l'objectiu de la càmera.

Destacar també que el resultat seria idèntic si es detectessin 3 trets facials en lloc de 68. Tanmateix, la diferència de temps seria mínima, i així puc reutilitzar el model predictiu d'extracció de trets facials entrenat per Dlib.

3.4 Classificació d'actors I: CNN entrenada per classificar

En la detecció de cares s'han vist dos algorismes tradicionals que donen grans resultats. Tanmateix, la tasca de classificació d'actors no és tan senzilla, ja que les diferències entre una cara i una altra poden ser mínimes. És per això que s'ha utilitzat les *state-of-the-art* Xarxes Neuronals Convolucionals, arquitectures d'Aprenentatge Profund (Aprenentatge Automàtic Supervisat). En aquest treball s'han dissenyat quatre opcions per resoldre la classificació de cares d'actors, dividides en 3 seccions diferents: I, II i III. Totes elles es basen principalment en Xarxes Neuronals Convolucionals, però tant la definició com la tècnica d'entrenament difereixen entre sí.

Aquesta primera alternativa consisteix a entrenar una Xarxa Neuronal Convolucional que, passant-li la imatge d'una cara preprocessada (alineada i retallada), doni uns valors d'activació a la capa de sortida que permetin directament fer una predicció sobre l'actor de la imatge d'entrada (probabilitat de pertànyer a cada classe). Per tant, s'entrenarà una CNN que classifiqui en actors directament [34], [35]. La figura 3.23 mostra esquemàticament un exemple del procés de predicció dissenyat, en cas que el problema a resoldre fos la classificació d'imatges entre gats i gossos. La classe predita seria *gos*.

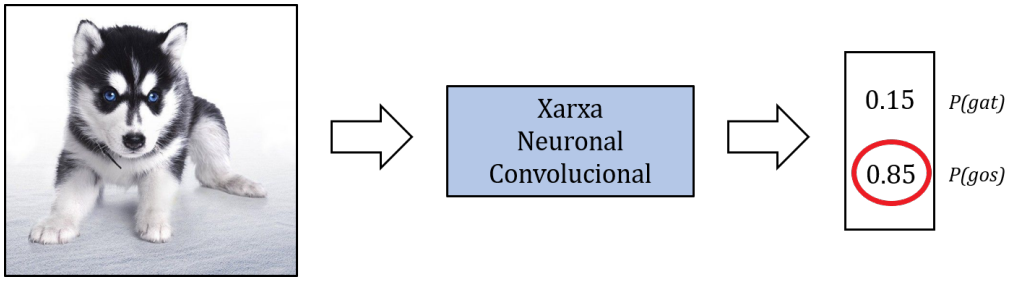

Figura 3.23: Exemple de CNN que classifica directament entre gats i gossos.

3.4.1 Mètode

Aquesta primera opció és la més directa de les tres que s'implementaran. Més específicament, consisteix a entrenar una Xarxa Neuronal Convolucional des de zero, que per cada imatge d'entrada doni com a sortida un volum unidimensional de tantes posicions com actors diferents hi hagi (70), indicant per cada una d'elles la probabilitat que té la cara d'entrada de correspondre a la respectiva classe/actor (distribució probabilística).

A continuació s'explicarà l'entrenament de la CNN, així com el procés de validació.

3.4.1.1 Entrenament de la CNN

En els següents subapartats es descriuen les característiques d'aquesta Xarxa Neuronal Convolucional, així com les tècniques per entrenar-la.

Conjunt de dades d'aprenentatge:

El conjunt d'imatges utilitzat per l'aprenentatge és el descrit a l'aparat 3.1, aplicant-li el procés de detecció, alineament i retallat. Per fer-ho, en primer lloc s'ha utilitzat el detector de cares amb l'algorisme dels Histogrames de Gradients orientats, per obtenir la *bounding box* de la cara continguda. Amb aquesta informació s'han obtingut els 68 trets facials mitjançant l'algorisme explicat a l'apartat 3.3.1, i finalment s'ha transformat i retallat la imatge a una mida de 96x96 píxels de manera que els trets dels ulls i el nas quedin ubicats en la mateixa posició en totes les imatges. Les 11 imatges que no han passat el procés de detecció de cares satisfactòriament han sigut modificades per unes de noves que sí l'han superat, i així s'ha pogut completar de nou el conjunt d'aprenentatge que consisteix en 2800 imatges de cares alineades, el qual s'ha duplicat mitjançant una simetria horitzontal. En total, el conjunt de dades d'aprenentatge consta de 5600 cares retallades i alineades de mida 96x96 píxels i de 3 canals (RGB).

Entrada i sortida:

La capa d'entrada està formada per les intensitats dels píxels de la imatge d'entrada, un volum de dimensions 96x96x3.

La capa de sortida és un volum de mida 1x1x70, i el valor d'activació de cada neurona indica la probabilitat de pertànyer a la respectiva classe (distribució probabilística sobre les classes). La suma dels 70 valors de la sortida és 1.

Arquitectura:

Per tal d'evitar l'overfitting, com que el conjunt d'aprenentatge és força petit, la Xarxa Neuronal Convolucional també ha de ser petita i no massa complexa. Tot i això, ha de tenir suficients capes de convolució per tal d'extreure característiques de la complexitat necessària per poder distingir cares de diferents persones. A continuació es descriuen les diferents capes i funcions que formen la CNN (en ordre d'actuació). La capa d'entrada no es mostra, i la capa de sortida és la que s'obté després d'aplicar la darrera funció (*Softmax*).

Nom	# Filtres	Mida filtre	Stride	Padding	Output Size
Convolution	32	6	1	2	95x95x32
BatchNorm					95x95x32
ReLU					95x95x32
MaxPooling		3	2		47x47x32
Convolution	32	5	1	2	47x47x32
BatchNorm					47x47x32
ReLU					47x47x32
MaxPooling		3	2		23x23x32
Convolution	64	5	1	2	23x23x64
BatchNorm					23x23x64
ReLU					23x23x64
AveragePooling		3	2		11x11x64
FullyConnected					1x1x70
Softmax					1x1x70

Taula 3.4: Definició de la Xarxa Neuronal Convolucional (I).

Funció de *loss*:

Aquesta és la funció que s'haurà de minimitzar durant l'entrenament. El seu objectiu és fer que per cada imatge d'entrada, la capa de sortida (distribució de probabilitats sobre les classes) tingui un 1 a la classe correcta, i 0 a la resta de valors. Per aconseguir-ho, s'utilitza la *Categorical Cross-Entropy Loss* [36], amb l'expressió 3.1. Suposant que la sortida de la CNN és y (sortida de la funció *softmax*), i y' és la sortida teòrica per la classe de la imatge d'entrada amb la que s'ha obtingut y, aleshores la *loss* per aquesta imatge d'entrenament és $H'_y(y)$.

$$H_{y'}(y) = -\sum_i y'_i * log(y_i) \tag{3.1}$$

Per exemple, si tinguéssim 2 classes d'etiquetes 0 i 1, i amb una imatge de la classe 0 la sortida y fos $[0.8, 0.2]$, la *Cross-Entropy Loss* d'aquest exemple concret individual seria $-log(0.8) = 0.097$, donat que y' seria $[1, 0]$. Únicament es té en compte la probabilitat de la classe teòrica, i no influeix la distribució de la resta de probabilitats en el càlcul (sí que ho fa en el procés de *back-propagation*).

Paràmetres i mètode d'entrenament:

Per entrenar aquesta xarxa s'ha utilitzat *back-propagation* juntament amb l'algorisme d'optimització *Stochastic Gradient Descent* (SGD) amb *momentum* (el *momentum* permet evitar els mínims locals de la *loss* i accelerar la convergència, tenint en compte els gradients de l'anterior minibatch) [37], [21]. Hi ha altres optimitzadors com Adam, però només s'utilitzarà si no hi ha convergència o aquesta triga molt. El *learning rate* és 0.001 i el *momentum* 0.9.

En total s'ha entrenat la xarxa durant 100 epochs. Cada epoch està format per 1120 minibatches, i cada minibatch està composat per 5 imatges. Els minibatches es formen a l'inici de cada epoch de forma aleatòria, així que un epoch representa una passada completa per tot el dataset d'aprenentatge (5600 imatges).

Per cada minibatch, es passen les 5 imatges per la Xarxa Neuronal Convolucional. Amb les respectives sortides es calcula la *cross-entropy loss* de cada exemple. Per cadascun d'ells, s'obtenen els gradients de la capa de sortida (derivada parcial de la *loss* respecte les activacions de la capa de sortida). A continuació, s'utilitza l'algorisme de *back-propagation* per calcular el gradient de cada paràmetre de la xarxa. Finalment, es fa la mitjana dels gradients de les 5 mostres, i s'actualitzen els paràmetres (una fracció dels gradients en negatiu) mitjançant SGD, intentant minimitzar la *loss* mitjana del minibatch.

Per tant, els paràmetres de la CNN s'actualitzaran cada 5 imatges. És una mida de batch petita, però no és massa rellevant ja que aquesta funció de *loss* és simple i molt fàcil de convergir.

3.4.1.2 Validació

Els paràmetres de la CNN canvien constantment, i s'ha d'avaluar en tot moment per aturar l'entrenament en el moment adient, evitant tant l'underfitting com l'overfitting. Per fer-ho, es fa un test de validació després de cada epoch, l'*accuracy* del qual serà un indicador del punt de l'entrenament en què ens trobem.

El procediment de validació és el següent: s'utilitza el conjunt d'imatges de validació preprocessat, és a dir, 350 cares alineades. Després de cada epoch, cada imatge del dataset es dona com a entrada a la Xarxa Neuronal entrenada. Segons la sortida (distribució probabilística), es prediu la seva classe. Comparant-la amb la classe real, es calcula la *accuracy* i s'obté la respectiva matriu de confusió.

Comentaris sobre la implementació: He utilitzat la llibreria Pytorch de Python per implementar tant l'entrenament com la validació [38], [39].

3.4.2 Resultats

Per cada epoch d'entrenament s'ha guardat la *loss* mitjana de tots els minibatches, és a dir, la *cross-entropy loss* mitjana de les 5600 imatges d'entrenament. També s'ha registrat la *accuracy* del test de validació. D'aquesta manera, es pot analitzar d'una banda si s'està entrenant correctament la xarxa (minimitzant la *loss*), i també per veure com es comporta la xarxa amb imatges amb les que no ha entrenat. Les dues gràfiques es mostren a les figures 3.24 i 3.25.

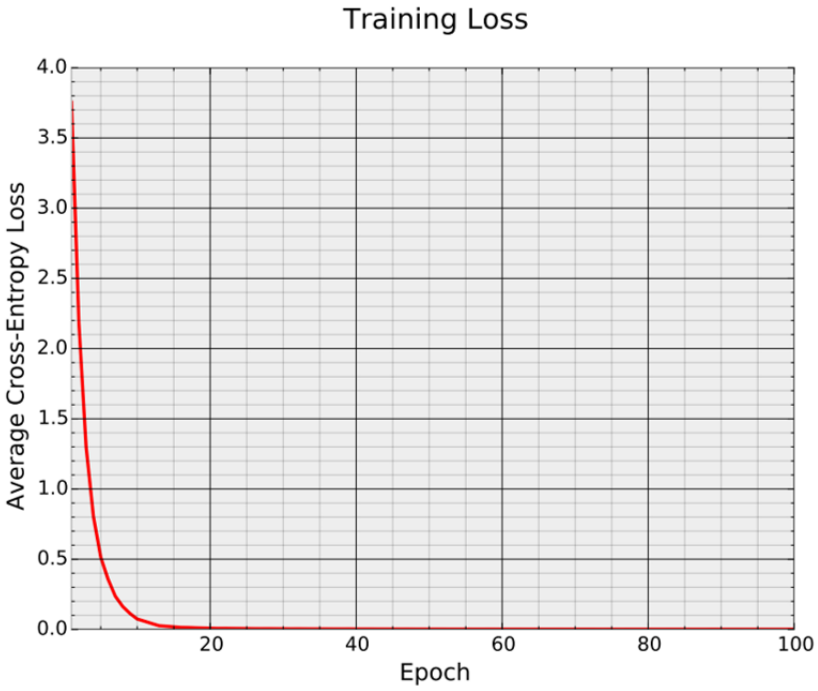

Figura 3.24: Training Loss per epoch (I).

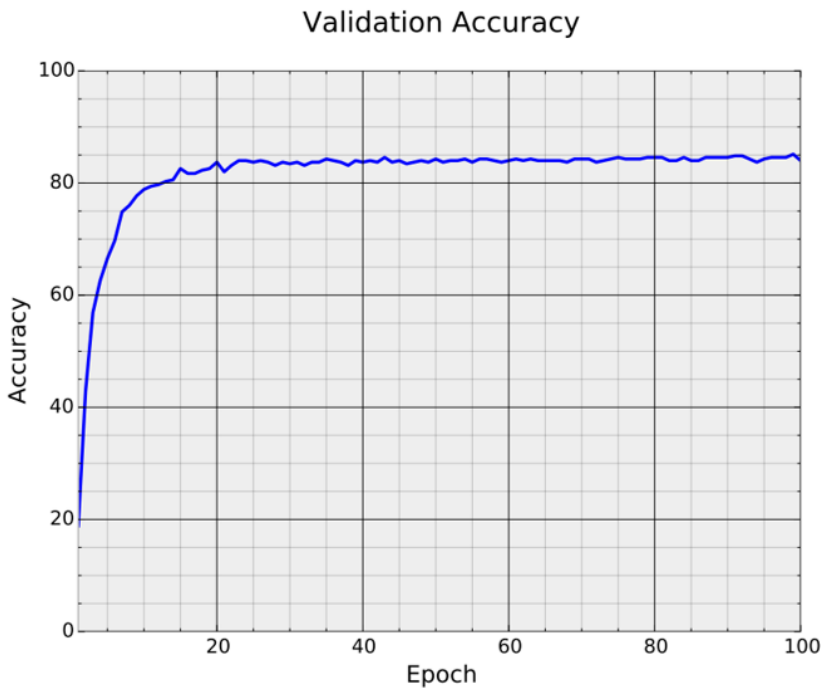

Figura 3.25: Validation Accuracy per epoch (I).

L'entrenament ha durat aproximadament uns 65 segons per epoch. La *validation accuracy* més alta ha estat de 84.29% i s'ha obtingut just després de l'epoch 35. La matriu de confusió corresponent es mostra a l'apèndix A.1.

3.4.3 Discussió

Els resultats obtinguts permeten extreure algunes conclusions.

En primer lloc, el model ha convergit suaument i ràpidament, i la *loss* ha passat a ser quasi 0 a partir de l'epoch 15 i de forma estable. Això és un indicatiu que la *training accuracy* és propera al 100%. Per tant, donats els bons resultats de convergència no ha estat necessari provar altres optimitzadors com Adam, el qual hagués afegit probablement més oscil·lacions.

En segon lloc, destacar que la *validation accuracy* màxima ha estat de 84.29%, i superior al 80% a partir de l'epoch 15. Són uns bons resultats, però és evident que hi ha overfitting, provocat tant per la reduïda mida del dataset d'aprenentatge (fins i tot després d'aplicar la tècnica de *data augmentation*) com per la funció de *loss* utilitzada (*cross-entropy*), la qual agrupa les imatges d'una mateixa classe en un *cluster* molt petit, sense comparar-se amb altres classes. Destacar que per corregir l'overfitting s'ha intentat afegir regularització mit-

jançant *weight decay*, capes de *dropout* i *batch normalization*. L'únic que ha estat fructífer és la *batch normalization*, augmentant la *validation accuracy* quasi un 10%, i és per això que s'ha utilitzat en aquesta arquitectura de CNN definitiva.

És possible que aquest mètode d'entrenament no sigui apropiat per un problema tan complex com la classificació de cares. És cert que ha demostrat una gran fiabilitat en la detecció de números escrits a mà, per exemple, però aprendre a diferenciar cares de persones diferents sent resistent a les variacions intrapersonals és un problema molt més difícil, que requereix tècniques més complexes. L'única opció que quedaria per obtenir millores notables seria augmentar el conjunt d'imatges d'aprenentatge, o bé utilitzar altres funcions de *loss* més innovadores com per exemple *Large Margin Softmax Loss* [40]. Això ho deixo en tot cas com a treball futur, ja que prefereixo centrar-me en tècniques d'aprenentatge radicalment diferents com les que es descriuen a continuació.

3.5 Classificació d'actors II: CNN entrenada per representar

La Xarxa Neuronal Convolucional de la opció I s'havia entrenat per ser capaç de classificar directament. Aquesta segona alternativa no és tan directa però ofereix moltes més possibilitats: consisteix a entrenar una CNN perquè generi una representació de la imatge d'entrada, és a dir, un vector de característiques [41]. És el que es coneix com *representation learning*, i es combinarà també amb una tècnica de *metric learning*. Per fer la classificació de cares, s'haurà d'entrenar a més un classificador (*Linear Support Vector Machine*) que, prenent com a entrada la representació obtinguda amb la CNN, predigui l'actor a qui correspon.

La figura 3.26 mostra esquemàticament un exemple del procés de predicció dissenyat, en cas que el problema a resoldre fos la classificació d'imatges entre gats i gossos i només s'extraguessin 4 característiques per imatge. La classe predita seria *gos*.

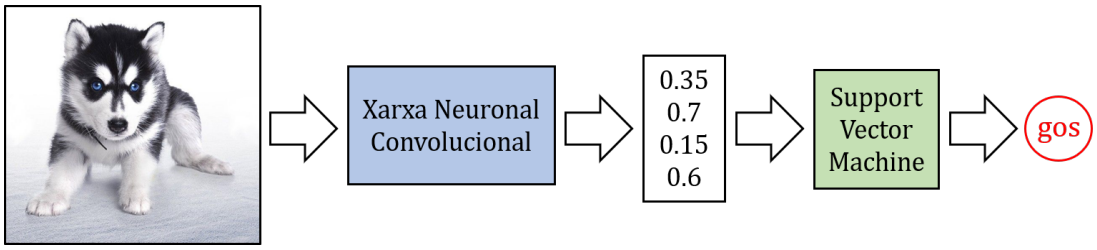

Figura 3.26: Exemple de classificació entre gats i gossos amb una CNN que representa i un SVM classificació.

A l'apartat del mètode es descriuen les característiques i algorismes d'entrenament per fer-ho possible.

3.5.1 Mètode

Aquesta segona opció és més complexa que la primera, i s'haurà de veure si així s'obtenen uns millors resultats. El mètode es basa en FaceNet [2] i es divideix en tres parts: entrenament de la Xarxa Neuronal Convolucional, entrenament del classificador, i validació.

En primer lloc, s'entrenarà una CNN des de zero perquè sigui capaç d'extreure 128 característiques de la imatge d'una cara, que estarà alineada i retallada. Aquest descriptor de característiques es referenciarà com a representació o *embedding*, i tindrà la propietat que l'arrel quadrada de la suma quadràtica dels seus elements doni 1 (veure exemple simplificat

de mida 4 a la figura 3.26). Amb això, sinònim a tenir longitud 1, s'aconsegueix limitar l'espai de representació. Conceptualment, la representació de cada cara correspondrà a un punt d'una hiperesfera unitat de 128 dimensions. L'objectiu serà que cares d'una mateixa classe tinguin un *embedding* similar, i cares de classes diferents tinguin *embeddings* diferents. Per fer-ho, s'utilitza el concepte de distància. La distància entre dues cares A i B indica el grau de similitud (inversament proporcional), i es calcula amb l'expressió 3.2 (el quadrat de la L2-norm de la diferència dels *embeddings*).

$$d = ||f(A) - f(B)||_2^2 \tag{3.2}$$

on $f(x)$ és l'*embedding* de la imatge x.

A l'apartat d'entrenament de la Xarxa s'explicarà com s'aconseguirà minimitzar la distància entre imatges d'un mateix actor i maximitzar a la vegada la distància respecte altres actors. A la figura 3.27 es mostra aquesta idea en el cas simplificat d'un *embedding* de mida 2.

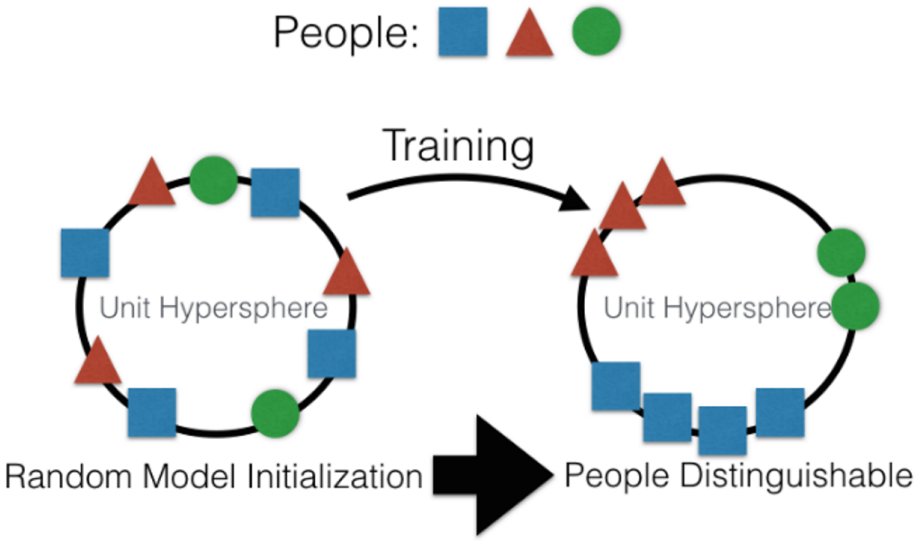

Figura 3.27: Representació dels *embeddings* de mida 2 abans i després de l'entrenament (hiperesfera de 2 dimensions). L'objectiu és que els *embeddings* d'una mateixa persona s'agrupin, i s'allunyin dels de persones diferents.

La representació haurà de ser resistent a variacions intrapersonals com són edat, maquillatge, expressions... i al mateix temps haurà de saber distingir entre les variacions interpersonals.

Amb la Xarxa Neuronal Convolucional només s'aconseguirà extreure un *embedding* per cada cara. S'haurà de crear per tant un classificador (*Linear Support Vector Machine*) que

s'entrenarà utilitzant els *embeddings* generats per la CNN. El procés es descriurà al segon apartat.

Finalment, s'explicarà el procés de validació d'aquest sistema de classificació basat en CNNs i SVMs.

3.5.1.1 Entrenament de la CNN

En els següents subapartats es descriuen les característiques d'aquesta Xarxa Neuronal Convolucional, així com les tècniques per entrenar-la.

Conjunt de dades d'aprenentatge:

El conjunt d'imatges d'aprenentatge és el mateix que el de la opció I. En total, consta de 5600 cares retallades i alineades de mida 96x96x3 píxels (3 canals RGB) cadascuna.

Entrada i sortida:

La capa d'entrada està formada per les intensitats dels píxels de la imatge d'entrada, un volum de dimensions 96x96x3.

La capa de sortida és un volum de mida 1x1x128, el qual representarà les 128 característiques de la cara d'entrada (*embedding*).

Arquitectura:

Per poder fer un anàlisi comparatiu d'aquesta opció de classificació respecte l'anterior (CNN entrenada per classificar), s'ha utilitzat una arquitectura quasi idèntica. Simplement s'ha eliminat la capa *fully-connected* de mida 70 i la funció de *softmax*, i s'han substituït per una capa *fully-connected* de 128 neurones i una posterior funció de normalització L2 (dividint cada element per la L2-norm del vector), per fer que la seva longitud sigui 1 (hiperesfera 128-dimensional).

Nom	# Filtres	Mida filtre	Stride	Padding	Output Size
Convolution	32	6	1	2	95x95x32
BatchNorm					95x95x32
ReLU					95x95x32
MaxPooling		3	2		47x47x32
Convolution	32	5	1	2	47x47x32
BatchNorm					47x47x32
ReLU					47x47x32
MaxPooling		3	2		23x23x32
Convolution	64	5	1	2	23x23x64
BatchNorm					23x23x64
ReLU					23x23x64
AveragePooling		3	2		11x11x64
FullyConnected					1x1x128
L2-Normalization					1x1x128

Taula 3.5: Definició de la Xarxa Neuronal Convolucional (II).

Funció de loss:

Com s'ha comentat prèviament, l'objectiu és aconseguir que *embeddings* d'una mateixa persona siguin més semblants que *embeddings* de persones diferents. Per aconseguir-ho, s'ha utilitzat el que es coneix com a *triplet loss* [2], una tècnica relativament moderna que utilitza l'expressió de distància. La *loss* no es calcula per cada imatge, sinó per cada tripleta de 3 imatges: 1 persona de referència (*anchor*), 1 persona de la mateixa classe que l'*anchor* (*positive*), i 1 persona d'una classe diferent (*negative*). Es genera l'*embedding* de cadascuna de les tres imatges passant-les per la CNN, i l'objectiu és que es compleixi la desigualtat 3.3.

$$||f(anchor) - f(positive)||_2^2 + \alpha \leq ||f(anchor) - f(negative)||_2^2 \tag{3.3}$$

on $f(x)$ és l'embedding de la imatge x i α és el marge mínim que es vol forçar entre positius i negatius respecte l'*anchor*.

Com es pot apreciar, l'objectiu no és fer que totes les imatges d'una mateixa classe estiguin el més juntes possible (conseqüència de l'entrenament amb *cross-entropy loss*), sinó que aquesta expressió força que, respecte l'*anchor*, la classe negativa estigui més lluny que la positiva, concretament una distància superior al marge α. Això permet una certa flexibilitat dins d'una mateixa classe sempre mantenint un marge de seguretat amb altres classes,

reduint l'efecte d'overfitting. La figura 3.28 representa esquemàticament l'entrenament amb *triplet loss*.

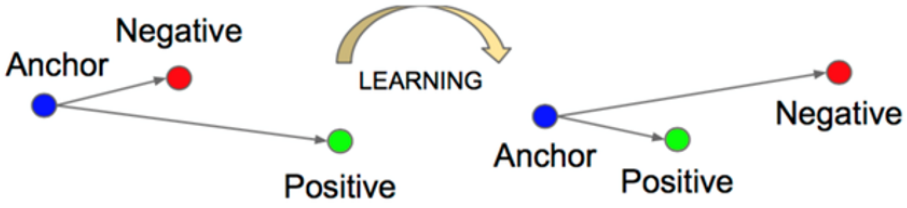

Figura 3.28: Esquema de l'entrenament amb *triplet loss*. S'ha d'aconseguir que la distància entre entre els *embeddings* de l'*anchor* i el *positive* (imatges d'una mateixa classe) sigui menor que la distància entre l'*anchor* i el *negative* (classe diferent).

Si es segueix el desenvolupament, s'obté la desigualtat 3.4.

$$||f(a) - f(p)||_2^2 - ||f(a) - f(n)||_2^2 + \alpha \leq 0 \tag{3.4}$$

on a és l'*anchor*, p és el *positive* i n és el *negative*.

Finalment, s'expressa la *loss* L de la tripleta d'imatges a, p i n com una funció (expressió 3.5).

$$L(a, p, n) = \max(||f(a) - f(p)||_2^2 - ||f(a) - f(n)||_2^2 + \alpha, 0) \tag{3.5}$$

L'objectiu de l'entrenament és fer que la *loss* de totes les tripletes sigui 0, ja que això significarà que la cara negativa està allunyada la suficient distància de l'*anchor* respecte la positiva. Això es pot escenificar amb la figura 3.29. Les representacions de l'*anchor* (a) i *positive* (p) són fixes, i està representada la distància respecte l'*anchor* (definició). Si es seleccionen *negatives* de la zona verda, la *triplet loss* serà 0. Si es seleccionen de la zona taronja, serà entre 0 i α, i de la zona vermella, superior a α. Si la representació del *positive* i *negative* és la mateixa o bé estan a la mateixa distància de l'*anchor*, la triplet loss valdrà α exactament.

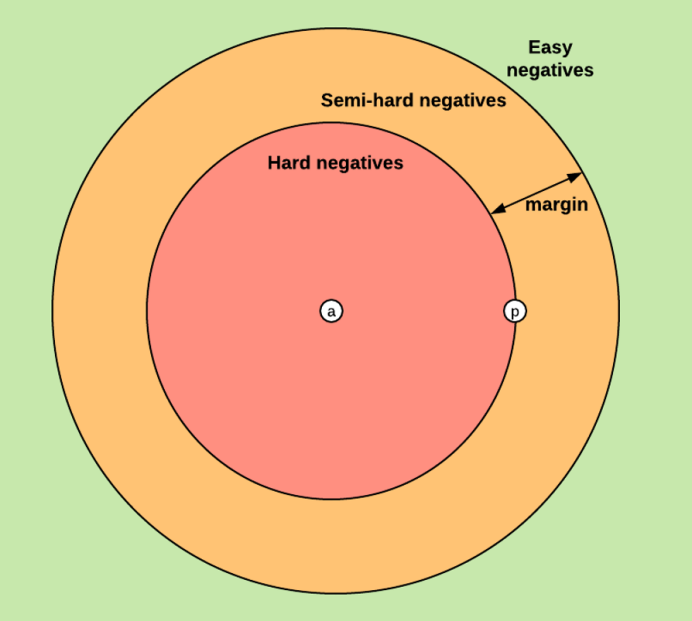

Figura 3.29: Tipus de *negatives* en funció de la distància respecte l'*anchor* i el *positive*.

Paràmetres i mètode d'entrenament:

El marge α del *triplet loss* serà 0.2. L'optimitzador que s'utilitzarà és Adam (Adaptative Moment Estimation) [21]. Es va començar a utilitzar SGD amb *momentum*, però era molt lent, probablement degut a la complexitat de la funció de *loss*, així que es va canviar per Adam, similar però normalment amb una convergència més ràpida. El coeficient d'aprenentatge (*learning rate*) és 0.001, i els paràmetres β_1, β_2 i ϵ valen 0.9, 0.999 i 1e-8. El número d'epochs totals serà 300 (el triple que abans ja que la convergència és més lenta), i s'utilitzaran tests de validació per comprovar el nivell d'entrenament després de cada epoch.

Cada epoch estarà format per 9 minibatches, i cada minibatch estarà constituït per un total de 300 imatges extretes aleatòriament del conjunt d'aprenentatge (15 actors diferents i 20 imatges per actor). La mida del minibatch ha de ser suficientment gran com per poder tenir suficients tripletes diferents amb les quals entrenar i accelerar així el procés, a diferència de l'opció anterior on no era tan determinant. La mida del minibatch ha de ser per tant gran, però hi ha un límit: les 300 imatges es passen a la CNN a la vegada, i consumeixen aproximadament 12 GB de memòria RAM. És un factor a tenir en compte.

Una altra diferència respecte la xarxa de l'opció I és que en aquest cas un epoch no és un passada completa de tot el dataset d'aprenentatge, sinó 2700 imatges amb algunes repeticions (una mateixa imatge pot formar part de més d'una tripleta). Com que la *loss* es calcula per tripletes, és difícil controlar que totes les imatges contribueixin a la *loss* final.

A més, la quantitat de minibatches per epoch només determina com de freqüentment es fa el test de validació. He decidit fer-lo bastant sovint per controlar en tot moment l'estat d'entrenament de la xarxa.

Per cada minibatch, el procediment és el següent [4]:

- Es calcula l'*embedding* de cadascuna de les imatges del minibatch passant-les com a entrada a la xarxa a la vegada.
- Es troben totes les parelles *anchor-positive* del minibatch, és a dir, imatges d'una mateixa classe. En total són 2850 parelles, ja que es generen combinacions i no permutacions (expressió 3.6).

$$15 \cdot \begin{pmatrix} 20 \\ 2 \end{pmatrix} = 2850 \tag{3.6}$$

- Per cada parella, es troben tots els *negatives* del minibatch que estiguin a una certa distància (marge α) del *positive* respecte l'*anchor*, tant per sobre com per sota (veure zona marcada en blau a la figura 3.30). Per fer-ho, s'utilitzen els *embeddings* generats per cadascuna de les 3 imatges a l'inici del minibatch (la *triplet loss* ha de valer entre 0 i 2α, ambdós exclosos). De tot el conjunt de *negatives* obtingut (concepte lleugerament diferent als *semi-hard negatives* de la figura 3.29), s'escull un únic *negative* aleatòriament. Si el conjunt és buit, no es té en compte la parella *anchor-positive* en qüestió. L'entrenament es fa d'aquesta manera ja que si s'utilitzessin *negatives* molt difícils o molt fàcils, podria provocar que s'arribés a un mínim local massa d'hora (model col·lapsat), o un entrenament massa lent respectivament. Aquestes seran les úniques tripletes (2850 com a màxim) que es tindran en compte a l'hora de calcular la *loss* mitjana total i actualitzar els paràmetres.

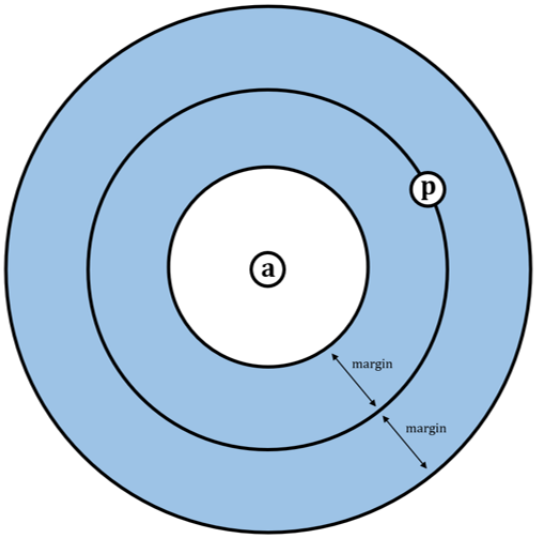

Figura 3.30: *Negatives* amb els quals es creen les tripletes. Han d'estar a una certa distància (marge α) del *positive* respecte l'*anchor*.

- Per cada tripleta, es calcula els gradient de la capa de sortida per cadascuna de les 3 imatges (derivada parcial de la *triplet loss* respecte els *embeddings* generats). S'aplica el procediment de *back-propagation*, i es calculen els gradients de la resta de paràmetres per cada imatge individual. Finalment, es calcula la mitjana de tots els gradients i es modifiquen els paràmetres de la CNN per tal de minimitzar la *triplet loss* total, mitjançant l'optimitzador Adam. Destacar que si no es troba cap tripleta en el minibatch (cap parella *anchor-positive* té un *negative* dins del marge), es salta al següent minibatch. Això sol succeir quan ja han passat molts epochs, i com que el model està més ben entrenat la majoria de classes estan ja agrupades i allunyades de la resta.

Comentaris sobre la implementació: No he utilitzat la llibreria Pytorch com abans ja que l'entrenament amb *triplet loss* és més complicat, sobre tot a causa de l'algorisme de selecció de tripletes. M'he basat en una implementació d'Openface [5], fent ús de Torch (plataforma de computació científica de gran eficiència que utilitza el llenguatge de programació LuaJIT) [42].

3.5.1.2 Entrenament del classificador

Ara que tenim una Xarxa Neuronal Convolucional que permet extreure característiques rellevants de la imatge d'una cara, s'ha de crear un classificador que donades les característiques d'una cara d'identitat desconeguda permeti predir a quin actor pertanyen amb major probabilitat.

Per fer-ho, en primer lloc s'han de generar els 5600 *embeddings* de les imatges d'aprenentatge mitjançant la CNN entrenada *per triplet loss* (s'emmagatzemen ordenats per classes, amb la corresponent etiqueta, en un fitxer CSV). A continuació, s'utilitzen els 5600 vectors de característiques i els corresponents identificadors de classe per crear un *Multi-class Linear Support Vector Machine* de 70 classes amb un paràmetre de regularització de 1.

Comentaris sobre la implementació: He utilitzat la llibreria Scikit-learn de Python per crear el classificador SVM (concretament la funció SVC) a partir dels *embeddings* generats amb la CNN [43].

3.5.1.3 Validació

Com amb la primera opció, s'utilitzarà el conjunt de 350 imatges de validació preprocessat (cares alineades retallades). El test de validació es farà després de cada epoch, i consistirà en el següent. Cadascuna de les imatges del dataset es passarà a la CNN com a entrada, s'obtindrà el seu *embedding*/representació, es donarà al classificador SVM com a entrada, i s'obtindrà la predicció de l'actor a qui correspon. Comparant la classe predita amb la classe veritable es calcularà l'*accuracy* total.

3.5.2 Resultats

Com anteriorment s'ha fet, per cada epoch d'entrenament s'ha guardat la *loss* mitjana de cada minibatch (*triplet loss* mitjana de les tripletes d'imatges seleccionades del minibatch). També s'ha registrat la *accuracy* del test de validació. Les figures 3.31 i 3.32 contenen la informació descrita.

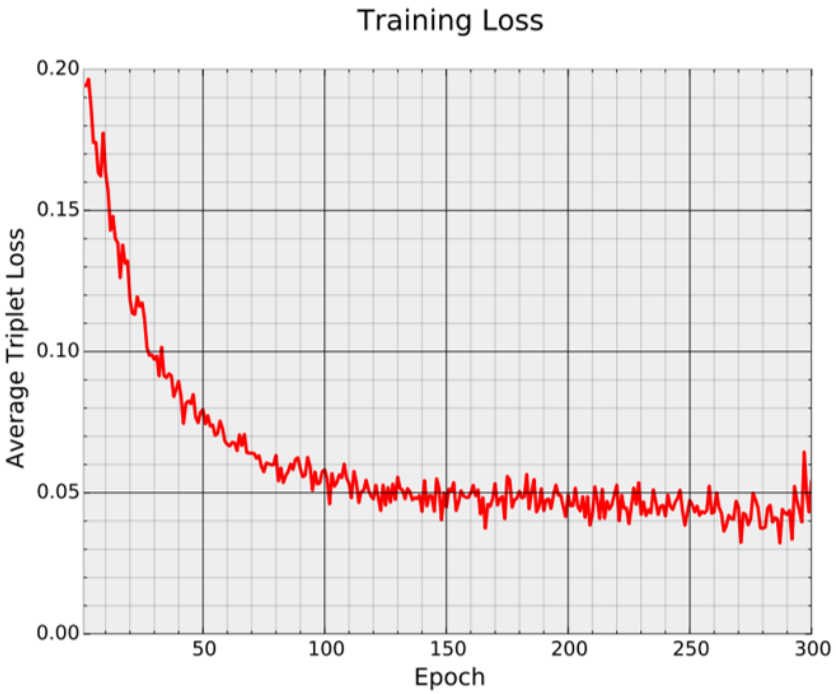

Figura 3.31: Training Loss per epoch (II).

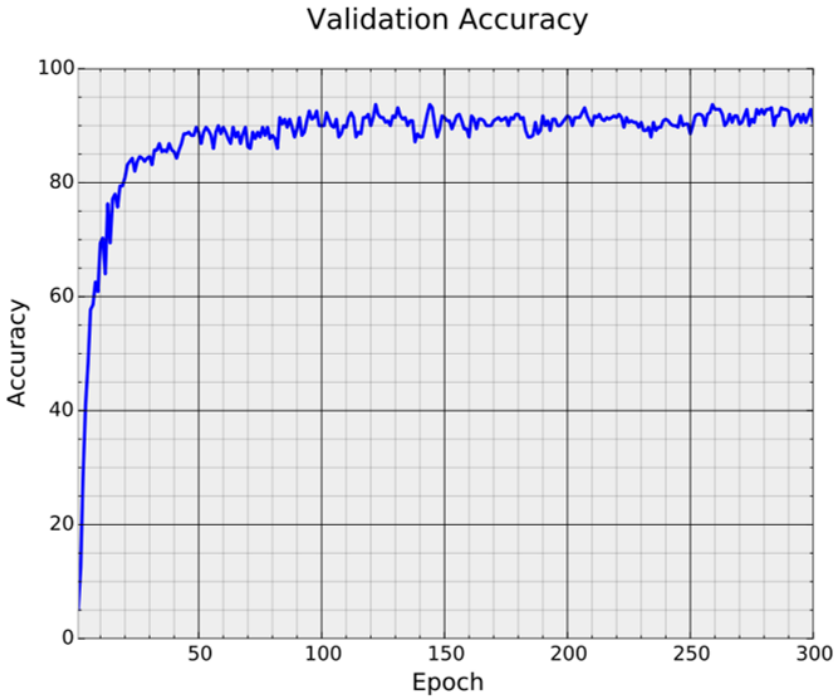

Figura 3.32: Validation Accuracy per epoch (II).

L'entrenament de la CNN ha durat uns 55 segons per epoch. D'altra banda, generar els 5600 *embeddings* i crear el classificador SVM ha trigat 2 minuts. La *validation accuracy* més alta ha estat de 93.43% i s'ha obtingut després de l'epoch 122. La matriu de confusió corresponent es mostra a l'apèndix A.2.

3.5.3 Discussió

Una diferència evident respecte el primer disseny és que hi ha molt més soroll i oscil·lacions, tant en la *loss* d'entrenament com en la *accuracy* de validació. Les fluctuacions en la *loss* es deuen a diverses causes: *la triplet loss* és una funció més complexa i difícil de minimitzar que la *cross-entropy loss*, i a més no totes les imatges d'aprenentatge contribueixen a l'actualització dels paràmetres de la xarxa a cada epoch, sinó 2700 com a màxim, així que depèn de l'aleatorietat de la selecció de les tripletes de cada minibatch. Respecte la irregularitat de la *validation accuracy* es deu principalment a que la xarxa està sota constants actualitzacions, i al tractar-se d'un dataset tan reduït (1 mostra = 0.29%), les afectacions són visualment molt notòries.

També és possible que amb l'optimitzador SGD la corba de *loss* fos més suau. Tanmateix, vaig fer una prova d'entrenament i després de 100 epochs la *training loss* era encara de 0.15, una evidència d'underfitting.

Tot i que la *triplet loss* mínima aconseguida sigui de 0.05 aproximadamentm, i no 0, això no implica que no hagi convergit. Com que aquesta funció es calcula tenint en compte únicament les tripletes amb una *loss* entre 0 i 0.4, si és 0.05 ens indica que la majoria de les tripletes que s'han trobat estan més a prop de ser fàcils que difícils. La *loss* mitjana únicament seria 0 si no s'hagués trobat cap tripleta que reunís les condicions, i en tal cas no s'actualitzarien els paràmetres i es passaria al següent minibatch.

També cal destacar que la *validation accuracy* màxima ha estat de 93.43%, i superior al 86% a partir de l'epoch 90. Per obtenir una *accuracy* de més de 80%, amb aquest mètode s'ha trigat 20 epochs (18 minuts), mentre que el primer va trigar 15 epochs (16 minuts). Comparativament, els dos mètodes són igual de ràpids a l'inici, però arriba un moment en què el primer s'estanca i en canvi el segon segueix millorant. La causa és l'ús de *triplet loss* en l'entrenament. Tenir en compte la relació amb altres classes i no únicament la pròpia fa que l'overfitting sigui quasi inexistent.

Hi hauria una altra alternativa d'entrenament que seria utilitzar *contrastive loss*, la qual és semblant al *triplet loss* però basat en parelles: acosta les representacions si són de la mateixa classe, i les allunya si són de diferents [41]. Tanmateix, l'he descartat ja que probablement sigui millor que *cross-entropy loss*, però no millor que *triplet loss* ja que la

distància és un concepte relatiu i no absolut, i com a tal és de gran ajuda fer l'entrenament amb tripletes d'imatges.

En resum, entrenar una CNN perquè classifiqui t'obliga normalment a utilitzar *cross-entropy loss*, la qual pot resultar massa simple per a resoldre alguns problemes. En canvi, entrenar una xarxa perquè representi dóna més possibilitats, permetent funcions *loss* més complexes com la *triplet loss*, que redueixin l'overfitting i donin millors resultats.

A més, també s'aconsegueix que la mida de la capa de sortida sigui fixa, i tan gran com el problema ho requereixi, fet que és un gran avantatge. Suposem per exemple que es modifiqués el número d'actors del sistema de reconeixement. L'arquitectura de la CNN de la opció II es mantindria fixa. En canvi, amb la opció I s'hauria de canviar la mida de la capa de sortida de la CNN (tantes neurones com classes hi hagués). Amb poques classes, aquest canvi no seria determinant. En canvi, si fossin moltes, suposaria tenir una capa de sortida (*fully-connected*) immensa, augmentant significativament el número de paràmetres i en conseqüència el risc d'overfitting.

3.6 Classificació d'actors III: *Transfer Learning*

Analitzant les opcions I i II s'ha arribat a la conclusió que la millor tècnica per resoldre la classificació de cares és entrenant una Xarxa Neuronal Convolucional amb *triplet loss* per extreure un conjunt de característiques (*embedding* o representació), i posteriorment utilitzar un classificador lineal per fer la predicció multi-classe. Hi ha una tercera opció, que es basa en la característica més important de les CNNs entrenades per representar: la seva arquitectura és independent del número de classes, així que aquestes poden ser variables.

Així com amb la opció I la *output layer* ha de tenir tantes neurones com classes hi hagi, la de la opció II té una mida fixa (128 característiques). Si l'entrenament es fa amb moltes imatges i moltes persones diferents es pot arribar a aconseguir que la Xarxa Neuronal Convolucional aprengui a generalitzar a classes que no hagi vist durant l'entrenament, i aprengui a extreure característiques rellevants no només per poder distingir les persones del seu dataset d'aprenentatge concret sinó de persones de tot el món. Aquest fet es podria aplicar al problema de la classificació d'actors en concret, si trobéssim una Xarxa Neuronal Convolucional extractora de característiques ja entrenada amb un dataset de cares molt extens.

Justament en això consisteix l'Aprenentatge per Transferència (*Transfer Learning* en anglès), una tècnica d'Aprenentatge Automàtic que consisteix a transferir el coneixement d'un problema a un de diferent però relacionat (generalment més específic) [44]. Aplicat al problema proposat en aquest treball, s'utilitzarien els coneixements de la classificació de cares (general) i s'aplicarien a la classificació de cares d'actors (específic). El *Transfer Learning* es sol utilitzar quan es disposa de poques dades d'aprenentatge i es vol evitar l'overfitting, i a més permet obtenir uns resultats molt bons amb un menor temps d'entrenament.

Hi ha dues maneres d'aplicar el *Transfer Learning* a aquest problema:

- Utilitzar la CNN preentrenada com a extractora de característiques.
- Reentrenar/afinar la CNN preentrenada (utilitzant el dataset d'aprenentatge), i utilitzar-la com a extractora de característiques.

Destacar que en tots dos casos s'hauria d'entrenar un nou classificador SVM a partir de les característiques extretes amb la nova CNN.

3.6.1 Mètode

Com s'ha comentat a la introducció, és important escollir una xarxa ja entrenada amb un dataset molt gran, i que estigui relacionada amb la tasca de classificació d'actors però que sigui més genèrica. Aquesta, es podrà mantenir fixa o bé reentrenar amb el mateix mètode que l'exposat a l'opció II. Finalment, s'entrenarà un classificador lineal SVM.

3.6.1.1 Selecció de la Xarxa Neuronal Convolucional preentrenada

El model de Xarxa Neuronal Convolucional que he escollit està entrenat per Openface (projecte de reconeixement facial de codi obert). Es tracta d'una CNN entrenada per representar que resol la tasca genèrica de la classificació de cares.

Concretament, el model s'anomena *nn4.small2.v1* [30], i ha estat entrenat amb 500 mil imatges (datasets Casia-WebFace i FaceScrub). Asseguren que el seu model aconsegueix una *accuracy* de 92.92% amb el dataset LFW [45]. Aquesta CNN utilitza una arquitectura molt complexa (figura 3.33), proporcional al número d'imatges del que disposa per entrenar, i es caracteritza per tenir molts mòduls *Inception*. No es mostren les funcions *BatchNormalization* ni *ReLU*, però s'apliquen després de cada convolució. És una versió reduïda de la CNN que utilitza el sistema de reconeixement facial FaceNet creat per Google, amb el que aconsegueix una *accuracy* de 99.64% a LFW, gràcies a tenir més de 200 milions d'imatges d'entrenament.

type	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj (p)	params	FLOPS
conv1 (7×7×3, 2)	112×112×64	1							9K	119M
max pool + norm	56×56×64	0						m 3×3, 2		
inception (2)	56×56×192	2		64	192				115K	360M
norm + max pool	28×28×192	0						m 3×3, 2		
inception (3a)	28×28×256	2	64	96	128	16	32	m, 32p	164K	128M
inception (3b)	28×28×320	2	64	96	128	32	64	L_2, 64p	228K	179M
inception (3c)	14×14×640	2	0	128	256,2	32	64,2	m 3×3,2	398K	108M
inception (4a)	14×14×640	2	256	96	192	32	64	L_2, 128p	545K	107M
inception (4b)	14×14×640	2	224	112	224	32	64	L_2, 128p	595K	117M
inception (4c)	14×14×640	2	192	128	256	32	64	L_2, 128p	654K	128M
inception (4d)	14×14×640	2	160	144	288	32	64	L_2, 128p	722K	142M
inception (4e)	7×7×1024	2	0	160	256,2	64	128,2	m 3×3,2	717K	56M
inception (5a)	7×7×1024	2	384	192	384	48	128	L_2, 128p	1.6M	78M
inception (5b)	7×7×1024	2	384	192	384	48	128	m, 128p	1.6M	78M
avg pool	1×1×1024	0								
fully conn	1×1×128	1							131K	0.1M
L2 normalization	1×1×128	0								
total									7.5M	1.6B

Figura 3.33: Definició de la Xarxa Neuronal Convolucional (III).

És el model perfecte per aplicar *Transfer Learning* a la tasca la classificació d'actors,

ja que resolen tasques quasi idèntiques a més d'altres coincidències necessàries per poder aplicar aquesta tècnica: l'entrada és la mateixa (cares de mida 96x96 alineades tenint en compte els punts externs dels ulls i la punta del nas), i la sortida també (representació de 128 característiques). Com a curiositat, el mètode d'entrenament també és el mateix (*triplet loss*), però això no és rellevant en la tècnica de *Transfer Learning*.

A continuació es descriuen les dues opcions que s'han implementat amb aquesta CNN preentrenada.

3.6.1.2 *Transfer Learning* I: Mantenir fixa la CNN preentrenada

Les dues tasques estan tan relacionades entre elles que és probable que utilitzant directament el model *nn4.small2.v1* com a extractor de característiques ja s'obtinguin uns bons resultats, veient que la *accuracy* que ha aconseguit amb un dataset de testing amb persones amb les que no ha sigut entrenat (LFW) és 92.92%.

Només caldrà generar els 5600 *embeddings* del nostre dataset d'aprenentatge utilitzant aquesta xarxa preentrenada, i crear-ne el classificador lineal SVM. Amb això s'estalvia haver d'entrenar una xarxa pròpia, una tasca que pot resultar difícil si es disposa d'un dataset reduït.

Es realitzarà un test de validació amb aquest sistema.

3.6.1.3 *Transfer Learning* II: Afinar la CNN preentrenada

La segona opció consisteix a reentrenar la xarxa *nn4.small2.v1*. La majoria de cares ja es classifiquen correctament sense necessitat de reentrenar. Tanmateix, és possible que les representacions de les cares d'alguns actors estiguin lleugerament solapades, i fer uns petits retocs a la xarxa preentrenada pot fer que els resultats millorin encara més. És el que es coneix com afinament o *fine-tuning* en anglès. Gràcies a això, es poden aconseguir tres millores de la *validation accuracy* respecte el número d'epochs d'entrenament: un inici més alt, una millora més ràpida, i una asímptota de convergència més alta (figura 3.34).

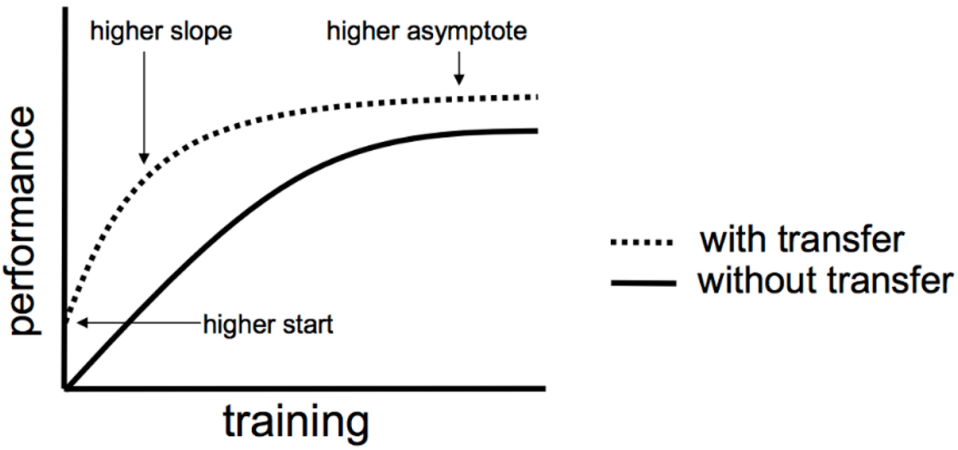

Figura 3.34: Gràfica del rendiment d'un model respecte el número d'epochs que ha sigut entrenat, que mostra els avantatges del *Transfer Learning*.

A més, al no entrenar-se la CNN des de zero, és de gran utilitat per reduir el temps d'entrenament i l'overfitting.

Entrant en els conceptes més tècnics relacionats amb l'entrenament, hi ha 2 opcions principals: congelar les primeres capes (extracció de característiques més genèriques) i entrenar únicament les darreres (més concretes), o desbloquejar-les totes i entrenar la xarxa sencera. La diferència principal entre els dos mètodes és el temps d'entrenament, menor en el primer cas. Tot i això, he decidit fer el *fine-tuning* amb la segona opció, ja que tinc la hipòtesi que puc obtenir millors resultats, i el temps d'entrenament no és un problema amb aquestes magnituds. L'algorisme d'entrenament pel *fine-tuning* utilitzat és idèntic a l'explicat a la secció 3.5.1.1, però amb un learning rate 10 vegades més baix (0.0001). S'entrenaran 100 epochs, amb el corresponent test de validació després de cadascun d'ells.

3.6.2 Resultats

El test de validació utilitzant el model preentrenat d'Openface (*nn4.small2.v1*) directament com a extractor de característiques ha donat una accuracy de 95.43% (veure matriu de confusió a l'apèndix A.3). D'altra banda, els resultats amb la xarxa afinada es mostren a les figures 3.35, 3.36 i 3.37.

Figura 3.35: Training Loss per epoch (III).

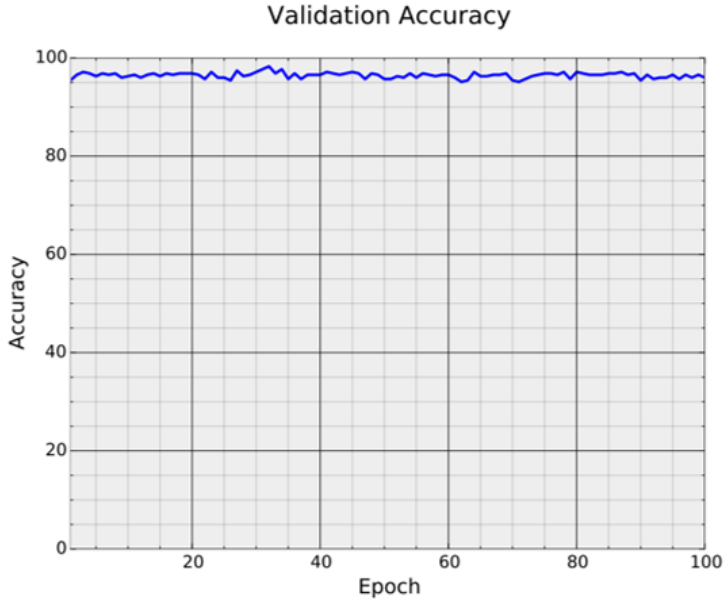

Figura 3.36: Validation Accuracy per epoch (III).

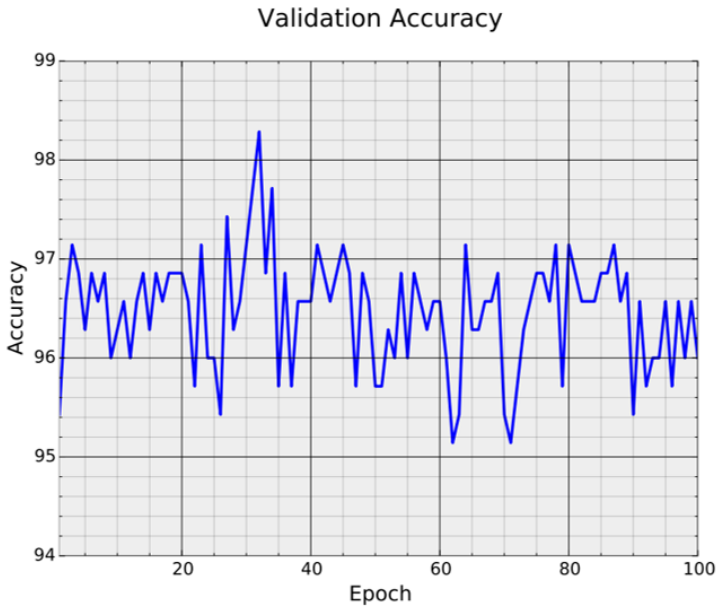

Figura 3.37: Validation Accuracy per epoch (III), ampliació.

El temps d'entrenament ha sigut de 130 segons per epoch, i la *validation accuracy* més alta s'ha aconseguit a l'epoch 32 (98.29%). La matriu de confusió es mostra a l'apèndix A.4.

3.6.3 Discussió

Amb les dues aplicacions del *Transfer Learning* s'han obtingut uns resultats millors que els aconseguits entrenant una CNN per representar des de zero (mètode de classificació II). Això és degut a que la tasca de classificació de cares s'adapta perfectament a la tasca de classificació de cares d'actors. A més, la Xarxa Neuronal preentrenada per reconèixer cares té una arquitectura molt complexa, i això sumat a l'immens dataset d'aprenentatge amb el que l'ha entrenat Openface permet extreure vectors de característiques molt més complexos i rellevants per cada individu.

Mantenint fixa la CNN preentrenada s'ha aconseguit una *validation accuracy* molt bona, però encara ha millorat més aplicant la tècnica d'afinament amb *triplet loss*, arribant fins al 98.29%. És cert que la *validation accuracy* fluctua, però això entra dins de la normalitat i, com a observació, durant la majoria d'epochs la *accuracy* està per sobre del 95.43% (*validation accuracy* amb el model preentrenat).

D'altra banda, les oscil·lacions que sí són excessives són les de la *loss* a partir de l'epoch 40, comparant-les amb les de l'opció de classificació II (figura 3.31), una clara senyal d'o-

verfitting. La xarxa té quasi 4 milions de paràmetres, així que el risc és molt alt amb un dataset d'aprenentatge tan petit. De fet, vaig intentar entrenar aquesta mateixa CNN des de zero, i vaig aconseguir una màxima *validation accuracy* de 82% després de 277 epochs (10 hores d'entrenament), un overfitting molt notori. Això és una mostra que el *Transfer Learning* ajuda a atenuar l'efecte d'overfitting, al menys durant els primers epochs. Per intentar corregir les oscil·lacions he provat de canviar l'optimitzador per SGD, però hi ha hagut underfitting.

En resum, el procés de *fine-tuning* sempre és de gran utilitat quan es disposa de poques dades per entrenar una xarxa. Tot i això, si la tasca per la que ha estat entrenada la xarxa preentrenada és tan similar a la del problema a resoldre com en aquest cas, pot no ser tan necessari l'afinament ja que el risc d'overfitting és massa alt pel mínim marge de millora, i en canvi més recomanable entrenar simplement el classificador lineal amb les característiques extretes per mitjà de la xarxa preentrenada.

3.7 Sistema de reconeixement d'actors en imatge

Una vegada es tenen tots els processos implementats, s'han de posar en comú per poder etiquetar els actors i actrius que apareguin en una imatge.

3.7.1 Mètode

L'entrada consisteix en una imatge en color, i la sortida és la mateixa imatge però amb unes *bounding boxes* superposades per cada cara, acompanyades per l'etiqueta de l'actor predit per cada cadascuna d'elles. Els passos són els següents:

- Detectar les cares mitjançant l'algorisme basat en els Histogrames de Gradients orientats, i guardar les dades de les *bounding boxes*.
- Per cada *bounding box*, utilitzar la tècnica d'alineament i retallat per tal d'obtenir una imatge de 96x96 píxels amb la cara totalment recta i centrada (punts externs dels ulls i nas en posicions fixes). D'aquesta manera, s'obtindran tantes imatges de 96x96px com cares es detectin a la imatge.
- Per cada imatge obtinguda amb el procés anterior, extreure les 128 característiques, utilitzant la Xarxa Neuronal entrenada que ha donat una *validation accuracy* de 98.29% (la de l'apartat 3.6.1.3, obtinguda aplicant la tècnica de *Transfer Learning*, afinant una CNN preentrenada que resol la tasca general del reconeixement facial).
- Passar les característiques extretes al classificador SVM ja entrenat amb totes les representacions del conjunt d'imatges d'aprenentatge, per obtenir una predicció de la classe a la que pertany cada cara.
- Superposar el nom de la classe predita a sota de la *bounding box* corresponent de la imatge original.

3.7.2 Resultats

La taula 3.6 mostra els temps de còmput aproximats per cada tasca del sistema reconeixement d'actors (aplicat en una imatge de mida 854x480px que conté 1 única cara). Es comenta també per cada tasca les seves dependències.

Tasca	Temps	Comentaris
Carregar imatge	3 ms	Depenent de la mida de la imatge
Detecció de cares	42 ms	Depenent de la mida de la imatge
Alineament i retallat	2 ms	Constant
Extracció de les 128 característiques (CNN)	33 ms	Depenent del número de cares
Predicció de l'actor (SVM)	1 ms	Depenent del número de cares
Total	**81 ms**	

Taula 3.6: Temps d'execució de les diferents tasques del sistema de reconeixement facial en imatge.

Respecte els resultats més visuals, les figures 3.38, 3.39 i 3.40 són un *collage* d'imatges on tots els actors s'han reconegut correctament. S'han agrupat per imatges amb 1 actor, imatges amb múltiples actors i imatges amb 1 actor en la seva època de joventut, respectivament.

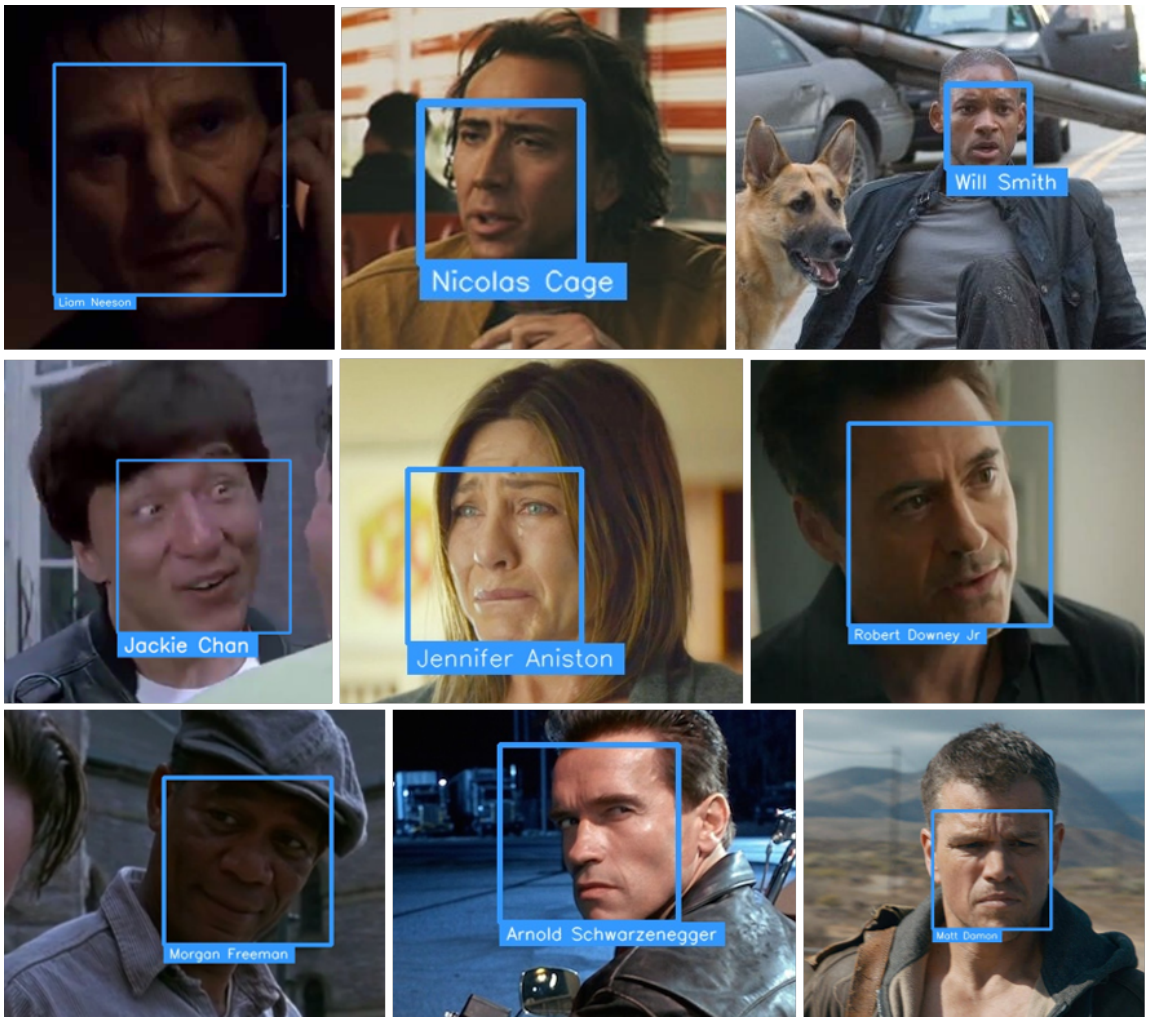

Figura 3.38: Sortides correctes del sistema de reconeixement d'actors en imatge (I).

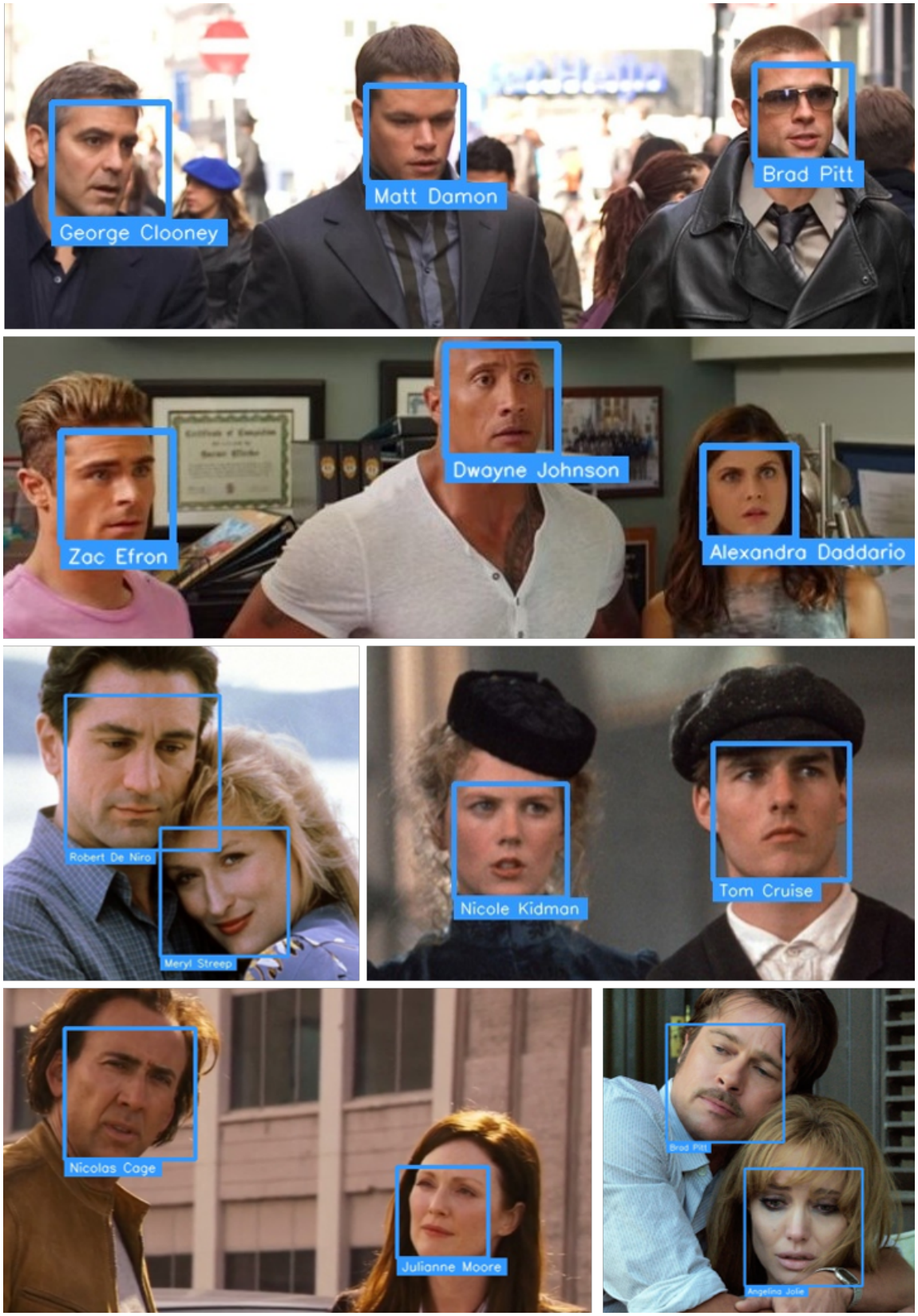

Figura 3.39: Sortides correctes del sistema de reconeixement d'actors en imatge (II).

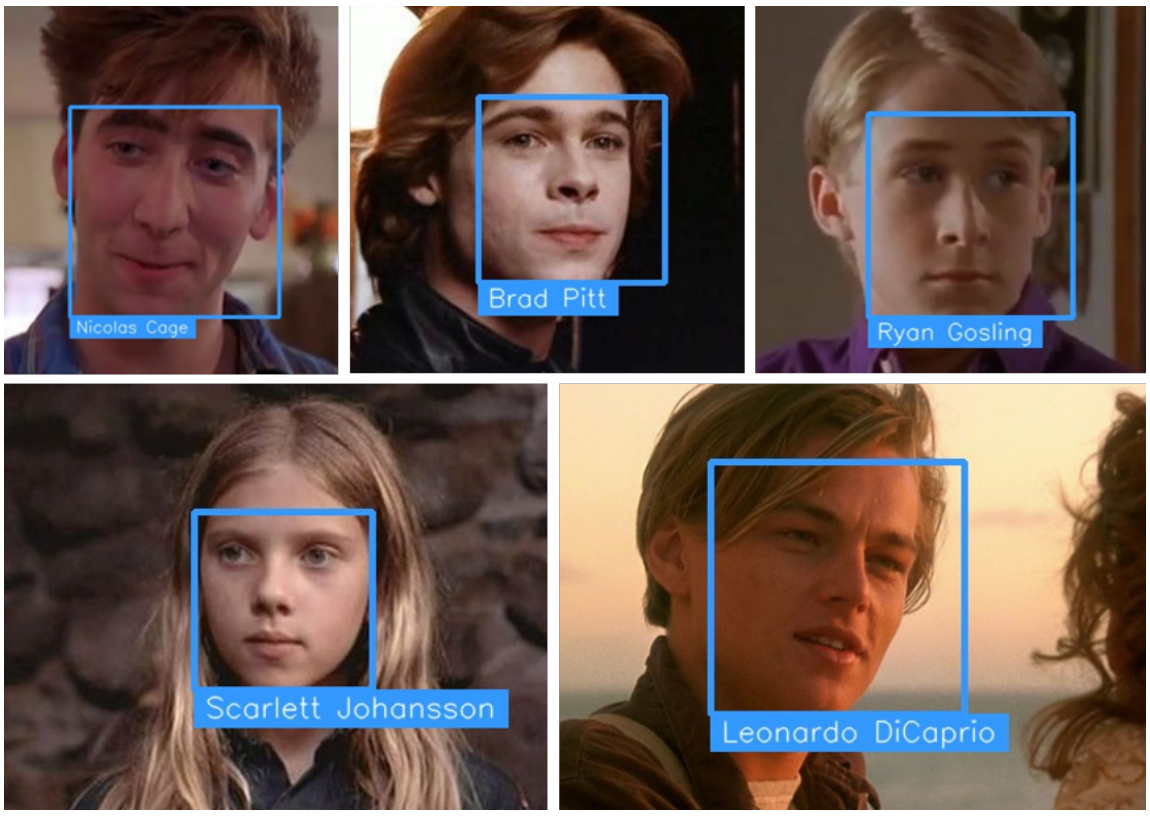

Figura 3.40: Sortides correctes del sistema de reconeixement d'actors en imatge (III).

D'altra banda, la figura 3.41 conté algunes de les imatges on algun actor no s'ha reconegut correctament.

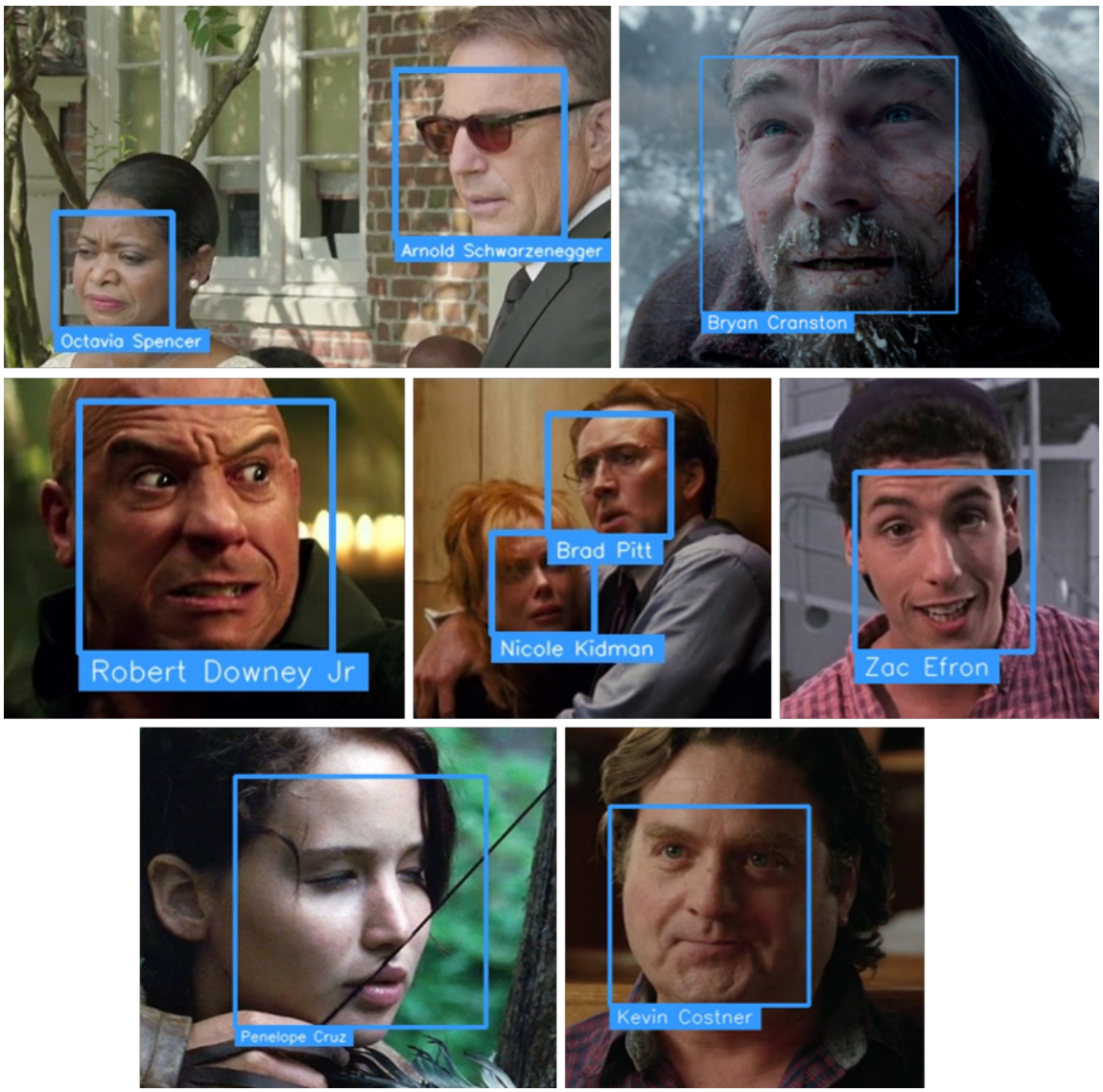

Figura 3.41: Sortides incorrectes del sistema de reconeixement d'actors en imatge.

3.7.3 Discussió

És en general un sistema de reconeixement facial força ràpid, penalitzat sobre tot per la detecció de cares i l'extracció de característiques. A més, té en general una gran fiabilitat, això sí amb algunes debilitats.

Després de fer moltes proves es poden extreure algunes conclusions. En primer lloc, el sistema és resistent a petits problemes d'il·luminació i d'oclusions. Si aquests són massa grans, el sistema de detecció ja no detecta les cares, indicant que ens trobem fora de l'abast

del projecte.

En segon lloc, resulta sorprenent que sigui capaç de reconèixer correctament els actors en les seves primeres actuacions (figura 3.40), ja que al conjunt d'aprenentatge s'havien utilitzat fotografies d'aquests actors en una època més moderna.

Passant a les fragilitats del sistema, observant la figura 3.41 se'n veuen diverses. Cares amb expressions facials estranyes, cares molt retocades (sang, neu...), actors amb o sense barba (el contrari del que sigui costum en ells), cares amb els ulls tancats i cares amb ulleres. Si un actor té una aparença que no presenta normalment, tampoc estarà present en el seu conjunt d'aprenentatge, i en conseqüència la predicció incorrecta anirà a favor d'un actor on aquesta característica hi tingui una major presència. Un exemple és en Kevin Costner, que al portar ulleres se l'ha confós amb Arnold Schwarzenegger, el conjunt d'aprenentatge del qual disposa de més imatges amb ulleres.

Tot i això, aquestes petites debilitats no són res comparades amb el gran problema d'aquest sistema: les imatges notòriament de perfil. Si una cara amb barba està mirant a la càmera, encara es poden extreure característiques rellevants del nas, els ulls, les celles, les proporcions... però no succeeix el mateix amb les de perfil. Com es veu a la figura 3.21, l'alineament funciona perfectament amb cares frontals i amb lleugeres rotacions. Tanmateix, amb les que estan tan de perfil que amb prou feines es veuen els dos ulls, la meitat de la imatge alineada serà *background*. En conseqüència, si l'actor no té gaires imatges de perfil al dataset d'aprenentatge, aleshores el confondrà amb algú que és possible que sigui molt diferent físicament, però que tingui més imatges de perfil. Tot i això, destacar que respectant les restriccions de l'abast del projecte, aquest problema no estaria inclòs, ja que només s'admeten cares lleugerament rotades i inclinades.

En resum, tot depèn del dataset d'aprenentatge: ha de ser el més gran, equilibrat i divers possible (expressions facials diferents, ulleres, barba, edat, rotacions...). D'aquesta manera, es poden entrenar representacions que siguin més resistents a diferències intrapersonals. També seria d'ajuda encara que més costós extreure les imatges d'aprenentatge de pel·lícules i no de festivals de cinema, ja que si no la majoria estaran somrient a càmera, una expressió facial no gaire freqüent a la majoria de pel·lícules.

3.8 Sistema de reconeixement d'actors en vídeo

Aquest és l'objectiu principal del problema proposat: etiquetar cada actor que aparegui en un vídeo.

3.8.1 Mètode

Una vegada explicat com funciona el sistema de reconeixement en imatge, el sistema en vídeo no té cap complicació:

- Aplicar el reconeixement d'actors en imatge per editar cada fotograma del vídeo.

3.8.2 Resultats

Alguns exemples de vídeos amb el reconeixement d'actors aplicat es podran veure a la fase de demostració de la defensa del TFG.

3.8.3 Discussió

Segons la taula 3.6, en un vídeo de 854x480px es podria aplicar el sistema de reconeixement en temps real a una freqüència de poc més de 12 fps (fotogrames per segon). Si el vídeo fos HD (1280x720px), aleshores seria de 7fps, principalment degut al temps de detecció de cares. No és una gran fluïdesa, però acceptable.

Tot i això, es podria aplicar el sistema de reconeixement a una escena de 30 fps per exemple. Hi hauria 2 opcions: la primera, ignorar la meitat dels fotogrames i mantenir la predicció i posició de les *bounding boxes* del fotograma anterior. La segona, utilitzar la tècnica del paral·lelisme, amb la qual s'obtindria igualment el resultat de cada fotograma amb un retard de 81 ms (en el cas de 480p), però al no ser iteratiu quan acabés el primer ja es tindria llest també el segon, i així successivament. Tot i això, queda fora de l'abast del projecte i es proposa com a treball futur.

Deixant de banda el temps d'execució i centrant l'atenció en la fiabilitat del sistema de reconeixement en vídeo, amb totes les proves que he fet, dedueixo que la *accuracy* real del sistema de reconeixement respecte les cares detectades és superior al 90%, però inferior a la de validació. Això ja era previsible, ja que aquesta només era orientativa per seleccionar el model i tècnica d'entrenament més adient per resoldre aquest problema, i no era un indicatiu de la fiabilitat real del sistema. També s'ha pogut sobrevalorat la *accuracy* per possibles imatges repetides amb el conjunt d'aprenentatge. La *accuracy* real no s'ha calculat amb

precisió per problemes de temps, ja que perquè la mesura fos objectiva requeriria un procés de *cross-validation* i un dataset de proves molt més extens i extret únicament de fotogrames de pel·lícules, i a més de gran diversitat cobrint totes les possibilitats d'expressions facials i aparença física. Únicament se n'ha fet un acotament a partir d'unes 20 escenes cinematogràfiques analitzades.

El problema que s'observa als vídeos és que tot i tenir una fiabilitat molt alta (posem 92%), en un vídeo de 1 minut a 30 fps (1800 fotogrames) hi hauria 144 prediccions errònies. Aquestes imprecisions aïllades són molt incòmodes a la vista, però es podrien neutralitzar fent un seguiment de les anteriors prediccions (treball futur).

Capítol 4

Planificació temporal

El projecte va iniciar-se el 23 de febrer de 2018, i s'ha acabat el 19 d'octubre de 2018, una setmana abans de la lectura del TFG (aquesta setmana es dedicarà a preparar la defensa del treball). Ha tingut una duració de quasi 8 mesos.

A continuació es descriuen les tasques principals d'aquest projecte així com la seva organització en l'espai temporal. També es comentaran desviacions respecte la planificació inicial.

4.1 Descripció de les tasques

Recordar que el projecte es divideix en una sèrie de mòduls d'implementació principals (el segon té 2 variants i el quart en té 4): 1) la creació d'un conjunt d'imatges d'aprenentatge i validació, 2) la detecció de cares, 3) l'alineament i retallat de cares (preprocessament), 4) la classificació d'una cara a la classe corresponent, 5) el sistema de reconeixement en imatge i 6) el sistema de reconeixement en vídeo.

A continuació es descriu breument cada tasca d'implementació, així les que permeten gestionar-les.

4.1.1 Fase inicial

Aquesta primera fase s'assoleix principalment a l'assignatura de Gestió de Projectes, i consisteix en una sèrie de lliuraments que permeten definir tots els aspectes relacionats amb la planificació del projecte: definició de l'abast i contextualització, planificació temporal,

anàlisi de la gestió econòmica i de sostenibilitat, una presentació preliminar, un plec de condicions relatiu a l'especialitat, i un document i presentació finals que ho recopilaran tot.

La fase inicial finalitza entre el 16 i el 20 d'abril, que és el període destinat a les presentacions finals de la fita inicial. Per tant, el temps aproximat per aquesta primera fase és de quasi de 2 mesos.

4.1.2 Anàlisi de les tècniques existents

És de gran importància analitzar l'estat de l'art relacionat amb el reconeixement facial. És una tasca present sobre tot a la fase inicial, però que també s'ha realitzat al llarg de tot el treball, trobant noves tècniques que a l'inici es desconeixien.

4.1.3 Definició del software

Basant-se en l'anàlisi de les tècniques existents s'ha de decidir els recursos software que s'utilitzarà. Com l'anterior tasca, aquesta forma part de la fase inicial, però pot tenir presència també al llarg del treball.

4.1.4 Configuració de l'entorn de treball

Instal·lar tot el software necessari per implementar el projecte. Aquesta tasca estarà present també al llarg de tot el treball, en paral·lel a les modificacions en la definició del software.

4.1.5 Creació d'un conjunt d'imatges d'aprenentatge i validació

Per desenvolupar el sistema de reconeixement d'actors és important disposar de dos conjunts d'imatges d'actors: un per l'aprenentatge i l'altre per la validació. S'haurà de buscar si ja n'hi ha de creats o si s'han de crear específicament.

4.1.6 Creació d'un detector de cares amb l'algorisme de Viola-Jones

Aquesta és la primera variant del mòdul de detecció de cares. Consisteix a aplicar l'algorisme de Viola-Jones per detectar totes les cares que apareguin en una imatge.

4.1.7 Creació d'un detector de cares amb Histogrames de Gradients orientats

És la segona variant del mòdul de detecció de cares. En aquest cas es farà ús dels Histogrames de Gradients orientats, i un classificador SVM.

4.1.8 Creació de la funció d'alineament i retallat de cares

Aquesta tasca (tercera variant) té com a objectiu preprocessar les cares detectades, per tal de corregir petites rotacions. A més, totes les imatges es retallaran a la mateixa mida per facilitar les fases posteriors.

4.1.9 Creació del classificador d'actors I entrenant una CNN que classifiqui

Aquesta quarta variant consisteix a implementar una CNN que faci la funció de classificador, amb el qual predir a quin actor correspon una determinada cara. Tasques que s'inclouen dins d'aquesta són:

- Entrenament d'una CNN amb *cross-entropy loss* que permeti predir l'actor a qui correspon la imatge d'entrada (una cara).
- Validació del classificador.

4.1.10 Creació del classificador d'actors II entrenant una CNN que representi

Aquesta tasca és la segona alternativa del mòdul i consisteix a crear el classificador d'actors a partir d'una CNN entrenada per extreure característiques d'una cara d'entrada, i una SVM per classificar-les a la classe corresponent. Tasques que s'inclouen dins d'aquesta són:

- Entrenament d'una CNN amb *triplet loss* que permeti extreure 128 característiques rellevants de la imatge d'entrada (una cara).
- Entrenament d'un classificador lineal SVM per poder predir l'actor a qui corresponen les característiques extretes amb la CNN.
- Validació del classificador.

4.1.11 Creació del classificador d'actors III amb *Transfer Learning* I

Aquesta tasca consisteix a crear un classificador utilitzant intacta una CNN ja entrenada per resoldre una tasca similar. Tasques que s'inclouen són:

- Obtenció d'una CNN extractora de característiques que ja estigui entrenada per resoldre un problema similar.
- Entrenament d'un classificador SVM amb la CNN preentrenada.
- Validació del classificador obtingut.

4.1.12 Creació del classificador d'actors III amb *Transfer Learning* II

Aquesta tasca consisteix a crear un classificador utilitzant una CNN creada per mitjà de l'afinament (*fine-tuning*) d'una CNN ja entrenada per resoldre una tasca similar. Tasques que s'inclouen són:

- Obtenció d'una CNN extractora de característiques que ja estigui entrenada per resoldre un problema similar.
- Entrenament de la CNN des de zero per extreure algunes conclusions sobre l'efecte del número de paràmetres en el risc d'overfitting.
- Reentrenament de la CNN preentrenada (*fine-tuning*).
- Entrenament d'un classificador SVM amb la CNN preentrenada afinada.
- Validació del classificador obtingut.

4.1.13 Creació del sistema de reconeixement d'actors en imatge

Aquesta tasca ha de permetre etiquetar el nom de tots els actors que apareixen en una imatge. Per fer-ho, es basa en les tasques anteriorment descrites:

- Detecció de cares amb l'algorisme més fiable.
- Alineament i retallat de les cares detectades.
- Classificació en actors de les cares alineades amb l'algorisme més fiable.

- Modificació de la imatge original amb les *bounding boxes* i etiquetes corresponents.

4.1.14 Creació del sistema de reconeixement d'actors en vídeo

Aquesta tasca és la que compleix l'objectiu principal, i consisteix a etiquetar el nom de tots els actors que apareixen en un vídeo. Per fer-ho, es basa en la tasca 4.1.13:

- Modificació de cada fotograma mitjançant l'execució del sistema de reconeixement d'actors en imatge.

4.1.15 Documentació de la fase final

Aquesta última fase consisteix a redactar la memòria del projecte. És una tasca present al llarg de tot el projecte, fins i tot durant la fase inicial.

Destacar que com s'ha definit a la metodologia, totes les tasques d'implementació estaran dividides en quatre fases iteratives: anàlisi, disseny, implementació i proves.

4.2 Duració aproximada

A la taula 4.1, es defineix una estimació de la repartició del temps per cadascuna de les tasques del projecte.

Tasca	**Duració aproximada (hores)**
Fase inicial	40
Anàlisi de les tècniques existents	35
Definició del software	6
Configuració de l'entorn de treball	25
Creació d'un conjunt d'imatges d'aprenentatge i validació	55
Detecció de cares amb l'algorisme de Viola-Jones	25
Detecció de cares amb Histogrames de Gradients orientats	35
Alineament i retallat de cares	8
Classificació d'actors I entrenant una CNN que classifiqui	40
Classificació d'actors II entrenant una CNN que representi	70
Classificació d'actors III amb *Transfer Learning* I	7
Classificació d'actors III amb *Transfer Learning* II	53
Reconeixement d'actors en imatge	15
Reconeixement d'actors en vídeo	9
Documentació de la fase final	75
Total	**498 h**

Taula 4.1: Distribució temporal del projecte, amb el temps estimat per cada tasca.

El diagrama de Gantt amb la repartició de tasques definitiva es mostra a la figura 4.1 (dividida en 3 parts per motius d'espai). El color negre en lloc de blau indica un període en què al no disposar d'ordinador, vaig dedicar-lo íntegrament a documentar-me.

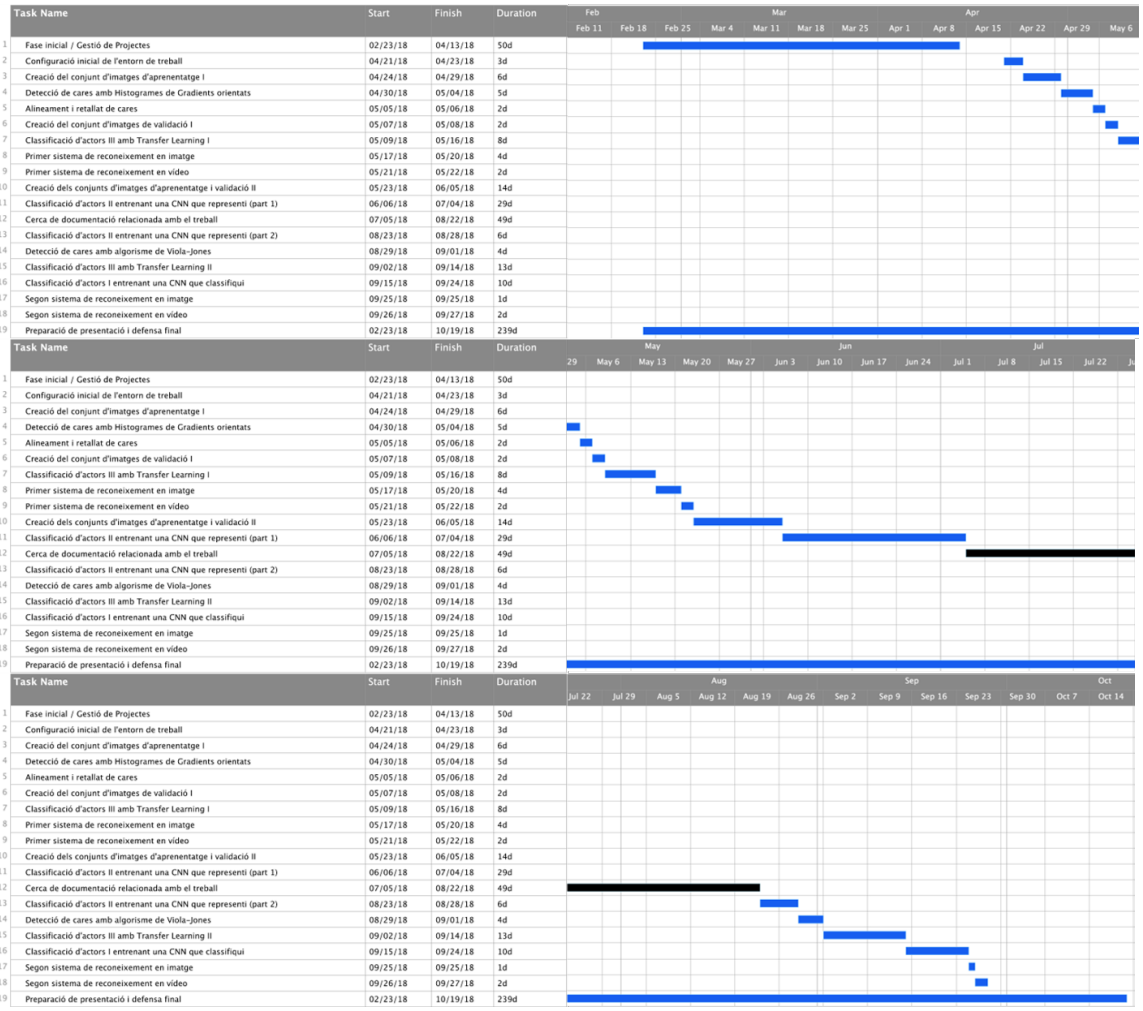

Figura 4.1: Diagrama de Gantt definitiu del projecte que representa la distribució temporal de les tasques.

4.3 Desviacions respecte la planificació inicial

Inicialment, estava previst presentar el TFG el torn de juny 2018. Com mostra el diagrama de Gantt (veure figura 4.1), el 22 de maig ja tenia una implementació funcional del reconeixement d'actors en imatge i vídeo, complint així el primer objectiu principal. Tanmateix, només havia utilitzat un algorisme per la detecció i una CNN per la classificació, no complint així el segon objectiu (analitzar diferents tècniques per obtenir un sistema el més fiable possible), així que vaig decidir endarrerir la defensa al següent torn (octubre 2018). Això m'ha permès multiplicar per 6 la quantitat d'imatges d'aprenentatge i validació (de 50 a 70 actors, i de 10 imatges per actor a 45 imatges per actor), i implementar l'algorisme

alternatiu de detecció de cares i tres tècniques addicionals per la tasca de classificació. Els objectius els he mantingut, i gràcies a presentar en un torn posterior m'ha permès satisfer-los en major mesura.

La principal causa d'aquesta desviació no va ser les hores previstes per cada tasca, la qual ha sigut força acurada, sinó l'estimació del temps de dedicació setmanal. Durant la fase inicial vaig preveure que de mitjana es podrien destinar 30 hores setmanals al desenvolupament del projecte. Tanmateix, a causa dels projectes d'altres assignatures això ha estat impossible, sent 20 hores/setmana el màxim que hi he pogut invertir fins el mes de juliol, i 30 hores/setmana la mitjana a partir del mes de setembre. Durant gran part de l'estiu no s'ha disposat d'ordinador i només s'ha pogut fer la tasca de recerca d'informació relativa al treball.

Capítol 5

Informe de sostenibilitat

5.1 Autoavaluació sobre la sostenibilitat

Fins ara només havia analitzat la sostenibilitat de forma superficial, i al llarg de l'autoavaluació m'he adonat que són pocs els projectes que la tenen present. És imprescindible que, al pensar en un projecte, s'intenti aplicar la innovació i la creativitat per generar idees que puguin contribuir a que sigui més sostenible. En un principi, quan sentia parlar de sostenibilitat pensava únicament en possibles afectacions al medi ambient, és a dir, consum d'energia o ús de paper. En canvi, no pensava en les altres dues dimensions: la social i l'econòmica, que estan relacionades amb com de rentable és un projecte.

Hi ha tres idees que haurien de ser fonamentals en qualsevol projecte. En primer lloc, un s'ha d'assegurar que sigui factible, disposant dels recursos humans i coneixement necessaris per la seva finalització, i és que abandonar un projecte a meitat de trajecte suposa un desaprofitament dels recursos molt important, tant humans com materials. I això ens porta al segon punt, la viabilitat econòmica d'un projecte: és imprescindible disposar dels recursos materials adequats ja que, altrament, el projecte no podrà ser finalitzat. A més, cal destacar que un treball col·laboratiu eficient pot contribuir a un millor aprofitament dels recursos. Finalment, al dissenyar un projecte és necessari analitzar com serà rebut per la societat, i preguntar-se si és realment necessari.

Per tant, per tal de deduir si un projecte de Tecnologies de la Informació i la Comunicació és sostenible, s'han d'analitzar tres aspectes bàsics: els indicadors d'impacte ambiental, la contribució en la societat (justícia social, equitat, transparència, diversitat, seguretat, accessibilitat), i la gestió econòmica (amortitzacions, planificació, costos fixes i variables). Per fer-ho, no s'ha de pensar només en les conseqüències immediates, sinó també en les indirectes que poden sorgir a partir d'aquestes. En definitiva, l'objectiu hauria de ser

dissenyar un projecte que contribuís al bé comú de la societat, respectant al màxim els principis deontològics relacionats amb la sostenibilitat en un projecte TIC.

A continuació es fa un anàlisi de la sostenibilitat d'aquest projecte des dels tres punts de vista.

5.2 Dimensió ambiental

5.2.1 Projecte posat en producció

La realització d'aquest projecte no ha tingut un gran impacte en el medi ambient, ja que al estar principalment relacionat amb el software, les úniques afectacions són les causades per les despeses indirectes (consum de paper i electricitat). Els productes hardware no es comptabilitzen ja que no s'han comprat expressament pel desenvolupament del projecte. He quantificat el consum total en 306 kWh (300 kWh d'electricitat i 6 kWh per fabricar 500 fulls de paper). És un consum mínim tenint en compte que correspon a un període de 8 mesos. Per reduir-lo, he evitat deixar encès l'ordinador quan no s'utilitzava, i n'he apagat la pantalla durant l'entrenament de les Xarxes Neuronals. També he procurat tenir una brillantor de pantalla adequada tant a l'ordinador com al telèfon mòbil. Finalment, també he intentat en la mesura del possible reutilitzar software ja implementat per llibreries públiques o sistemes de codi obert, ja que així s'estalvia l'energia destinada al procés d'entrenament (un exemple és l'algorisme de detecció de cares, amb un classificador ja entrenat). No he quantificat l'estalvi total, però estimo que ha sigut d'un 10% aproximadament.

Si tornés a realitzar el projecte, probablement necessitaria fer menys ús de l'ordinador, ja que al tenir més coneixement evitaria la necessitat de dedicar hores a adaptar-me al software i implementar algorismes que no donen bons resultats. Amb això s'aconseguiria reduir el consum d'electricitat.

5.2.2 Vida útil

Quan el producte estigui posat en marxa, l'únic recurs que s'utilitzarà és l'ordinador amb el qual cada usuari executarà el sistema de reconeixement d'actors, causant un consum mínim. Si no utilitzessin aquest sistema de reconeixement n'utilitzarien un altre, així que globalment l'impacte ambiental d'aquest projecte és mínim o neutre, ja que ni l'incrementa ni el disminueix.

5.2.3 Riscs

L'única eventualitat que podria causar un impacte ambiental més negatiu del previst seria que s'espatllés algun component de hardware i s'hagués de substituir per un de nou, fet que augmentaria el consum i la petjada ecològica del projecte.

5.3 Dimensió econòmica

5.3.1 Projecte posat en producció

És important fer una estimació del pressupost total per poder extreure una sèrie de conclusions sobre el desenvolupament del projecte, considerant els recursos humans, de hardware, de software i les despeses indirectes. Destacar que s'han produït modificacions respecte la previsió inicial, ja que el número total d'hores i dies dedicats al projecte ha canviat. A continuació es defineix la nova estimació de costos tenint en compte el diagrama de Gantt definitiu i el repartiment de temps entre les tasques de la secció 4.2, així com els recursos descrits a la secció 1.10.

Començant amb el pressupost de recursos humans, com a desenvolupador del projecte he dut a terme les tasques adjudicades a un cap de projecte, un dissenyador, un programador i un tester. El cost total es veu reflectit a la taula 5.1, tenint en compte uns preus per hora aproximats.

Rol	**Duració (h)**	**Preu per hora (€/h)**	**Preu total (€)**
Cap de projecte	135	20	2700
Dissenyador	68	15	1020
Programador	260	15	3900
Tester	35	10	350
Total	**498 h**		**7970 €**

Taula 5.1: Pressupost de recursos humans.

En relació al hardware, la taula 5.2 defineix el cost original de cada producte i la corresponent amortització, tenint en compte la nova duració del projecte i la vida útil aproximada de cada producte. Destacar que s'ha inclòs un nou producte respecte la planificació inicial (un telèfon mòbil, utilitzat sobre tot en la tasca de recerca d'informació).

Producte	Preu (€)	Vida útil aproximada (anys)	Amortització (€)
Ordinador de sobretaula	1036	6	115.1
Telèfon mòbil	174	3	38.7
Total	**1210 €**		**153.8 €**

Taula 5.2: Pressupost de hardware.

Respecte el software, la majoria de productes que he utilitzat són gratuïts. He tingut en compte que la vida útil dels productes de Microsoft Office és lleugerament inferior a la d'un l'ordinador de sobretaula, ja que és el temps aproximat amb què Microsoft deixa d'oferir suport per aquesta versió. L'anàlisi es mostra a la taula 5.3.

Producte	Preu (€)	Vida útil aproximada (anys)	Amortització (€)
SO Ubuntu 16.04	0	-	0
Python, Torch, etc.	0	-	0
Microsoft Office 2016	150	4	25.0
Total	**150 €**		**25.0 €**

Taula 5.3: Pressupost de software.

Quant a les despeses indirectes, és a dir totes aquelles que no estan immediatament relacionades amb el projecte però sí indirectament, com les despeses en electricitat (consum de l'ordinador i del telèfon mòbil) o de paper (impressions del treball), el pressupost aproximat es mostra a la taula 5.4. El consum d'electricitat ha incrementat lleugerament respecte la planificació inicial, ja que s'han destinat més hores de les previstes al projecte.

Producte	Unitats	Preu per unitat	Preu total (€)
Paper	1 paquet 500 fulls	8 €/paquet	8
Electricitat	300 kWh	0.145 €/kWh	43.5
Total			**51.5 €**

Taula 5.4: Pressupost de despeses indirectes.

Si es sumen tots els costos (veure taula 5.5), s'obté un pressupost total de 8199.5 €. És una quantitat 635 € superior a la prevista a la planificació inicial, una diferència que ha pogut ser coberta amb el 56% del fons de contingència.

Tipus	**Preu total (€)**
Recursos humans	7970.0
Hardware	153.8
Software	25.0
Despeses indirectes	51.5
Total	**8199.5 €**

Taula 5.5: Pressupost total del projecte.

Per intentar reduir el pressupost total he aplicat la mateixa tècnica que en la dimensió ambiental: consumir la menor electricitat possible (apagant l'ordinador quan està en desús, apagar la pantalla durant l'entrenament, i abaixar la brillantor de pantalla).

Gràcies a haver fet una estimació acurada del número d'hores destinades al projecte s'ha aconseguit que el cost final s'hagi ajustat prou bé a la previsió inicial. El número d'hores només ha augmentat en un 6%, tot i haver-se duplicat el període de desenvolupament (de 4 a 8 mesos). Això ha permès que la diferència de cost hagi pogut ser coberta pel fons de contingència.

5.3.2 Vida útil

El cost en la fase de vida útil d'aquest projecte és nul. Una vegada el sistema de reconeixement està creat, no es necessita fer cap actualització ni reparació, i el cost de manteniment és zero (només seria una mica superior si es decidís fer-ne una aplicació web). És impossible reduir aquest cost.

5.3.3 Riscs

L'únic que podria evitar que el projecte fos viable seria que l'ordinador de sobretaula amb el que es desenvolupa s'espatllés, i s'haguessin de comprar nous components. Tanmateix, l'ordinador encara està en garantia, així que aquesta eventualitat no afectaria a nivell econòmic.

5.4 Dimensió social

5.4.1 Projecte posat en producció

A nivell personal, la realització d'aquest projecte m'ha fet reflexionar sobre els algunes qüestions dels sistemes de reconeixement facial en general. Aquests encara són imperfectes i senzills de vulnerar, i abans de poder aplicar-los en un futur a aplicacions que ara semblen de ciència ficció, s'ha d'aconseguir que siguin infal·libles.

D'altra banda, el reconeixement facial és de gran utilitat en la seguretat (control de sospitosos en un event multitudinari per exemple), i em pregunto si això pot suposar a la llarga una pèrdua del dret a la privacitat. A més, aquests sistemes necessiten ser entrenats amb moltes imatges, fet que pot arribar a suposar una vulneració en termes de drets d'imatge.

5.4.2 Vida útil

El projecte ha solucionat el problema plantejat, complint tots els objectius de forma satisfactòria. Se'n podran beneficiar tant usuaris individuals com empreses de la manera que descriu la secció 1.5.3, sent una aplicació d'oci i a la vegada funcional que no demana cap mena d'informació personal a l'usuari. No hi ha cap col·lectiu per tant que es pugui veure perjudicat per aquest projecte, ja que és una aplicació que permet reconèixer a un conjunt d'actors (personatges totalment públics) en una imatge extreta d'una pel·lícula o sèrie. Diferent seria si hagués creat un sistema de reconeixement facial que permetés reconèixer totes les persones del món (Google i Facebook tindrien recursos per fer-ho), ja que això suposaria una amenaça a la privacitat de la societat, i estaria fent un ús no consentit de les imatges per l'entrenament d'un sistema de reconeixement.

5.4.3 Riscs

Un possible risc seria que a algun paparazzi amb molts recursos se li acudís aplicar el sistema de reconeixement d'actors per trobar celebritats al carrer, fora del context pel que s'ha implementat (per reconèixer-los en escenes de pel·lícules o sèries). Aquest seria un ús poc ètic que atemptaria contra la intimitat dels actors.

Capítol 6

Conclusions

Tots els objectius proposats s'han complert satisfactòriament. S'ha obtingut un sistema de reconeixement de 70 actors en imatge i vídeo molt fiable, després d'haver analitzat quatre alternatives de classificació d'actors diferents, i dues de detecció de cares.

Més específicament, el millor sistema de reconeixement que s'ha obtingut té les següents característiques: la detecció de cares es fa mitjançant Histogrames de Gradients orientats. L'alineament i retallat es fa amb un conjunt d'arbres de regressió. Finalment, es classifica a la classe corresponent mitjançant una Xarxa Neuronal Convolucional entrenada per extreure 128 característiques i un classificador SVM que les utilitza per fer una predicció. Aquesta CNN s'ha obtingut afinant una xarxa ja entrenada per Openface que resol el problema genèric del reconeixement facial (amb 500 mil imatges), aplicant una tècnica que s'anomena *Transfer Learning*. La *accuracy* de validació màxima ha estat de 98.29%, més alta que la *accuracy* real, la qual no s'ha pogut calcular però es preveu que sigui superior al 90%. A més de fiable també és un sistema ràpid, i sense paral·lelisme permetria processar una escena de 480p i 12 fps en temps real, una fluidesa força digna.

Analitzant els processos més importants per separat, l'algorisme de detecció amb Histogrames de Gradients orientats és més fiable que el de Viola-Jones, i s'ha utilitzat en el sistema tot i ser més lent per evitar que la detecció facial sigui un coll d'ampolla en termes de fiabilitat.

D'altra banda, respecte la fase de classificació/predicció, la Xarxa Neuronal Convolucional que millor ha funcionat és l'entrenada per representar i no per classificar, ja que al aprendre per tripletes (amb *triplet loss*, *back-propagation* i optimitzador Adam), aconsegueix separar les representacions de classes diferents i a la vegada acostar les d'una mateixa classe, sense concentrar-les en excés, reduint així el risc d'overfitting. A més, té l'avantatge que el número de classes és variable gràcies a tenir una *output layer* de mida fixa.

Fent un anàlisi més subjectiu dels resultats obtinguts, destacar que el sistema funciona en general molt positivament, i és resistent a l'edat, petites oclusions i irregularitats en la il·luminació. Tot i això, la major debilitat són les cares de perfil i els accessoris facials (ulleres, barba...). És difícil de solucionar al 100%, però podria ser corregit en major mesura amb un conjunt de dades d'aprenentatge més gran, divers i equilibrat, com en el que es basen les grans empreses com Google i Facebook en els seus sistemes de reconeixement facial.

Passant a la planificació temporal, el dia inicialment previst per acabar el treball es tenia una implementació funcional (la de l'apartat 3.6.1.3). Tanmateix, es va decidir endarrerir la data de la defensa per així poder dedicar aquest temps a implementar 1 sistema de detecció i 3 sistemes de classificació addicionals.

A mode de reflexió personal, l'Aprenentatge Automàtic i sobre tot l'Aprenentatge Profund estan guanyant terreny dins el camp de la Visió per Computador, i la majoria de tècniques més tradicionals van sent eclipsades per les més modernes. Abans de començar el treball, desconeixia el funcionament de les Xarxes Neuronals Convolucionals, únicament n'havia sentit a parlar a classe o llegit en algun article. El desenvolupament d'aquest projecte m'ha permès adquirir un coneixement molt valuós i amb grans possibilitats, que és sens dubte el futur de la Intel·ligència Artificial.

Capítol 7

Treball futur

El marge de millora d'aquest projecte és força gran. En primer lloc, en relació a l'abast, es podria augmentar el número d'actors que el sistema fos capaç de reconèixer, i també millorar la interfície gràfica per facilitar l'execució del sistema de reconeixement per part de l'usuari.

Posant el focus a millorar la fiabilitat del sistema, es podria augmentar el dataset d'entrenament amb més imatges per actor, més variades i de major resolució per obtenir uns millors resultats. Abans, però, serien prioritàries altres qüestions, com són superar el major punt dèbil: el reconeixement de cares de perfil.

El procés d'alineament és molt ràpid, però només és útil amb cares que estan lleugerament rotades, ja que les que estan significativament de perfil són difícils d'alinear, al poder aprofitar únicament una meitat de la cara. Per solucionar-ho, es podria optar per un algorisme de rotació 3D, basat en *Deep Learning* [31]. Tanmateix, és un tema que encara està en desenvolupament, i a més no és tan ràpid com el que s'utilitza en aquest problema (el temps d'execució és clau en el reconeixement en temps real). Una forma alternativa de millorar els resultats amb les cares de perfil podria ser augmentar la seva presència en el conjunt d'aprenentatge, o bé crear dos sistemes de reconeixement diferents, un per cares frontals/semifrontals i un per cares de perfil (hi ha detectors de cares de perfil).

En relació al reconeixement facial aplicat a un vídeo, ja s'ha comentat que es podria augmentar la freqüència de reconeixement mitjançant paral·lelisme. També es podria accelerar el procés aplicant un algorisme de *face tracking* basat en *mean-shift* per exemple. Això evitaria haver d'aplicar la detecció facial a cada fotograma, un algorisme temporalment costós. D'altra banda, per evitar les prediccions errònies aïllades, es podria fer un seguiment de les prediccions fetes fins el fotograma actual, i quan n'hi hagués una diferent de l'anterior, analitzar la seva diferència respecte aquesta, i si fos relativament petita, ignorar la predicció

errònia. Una forma alternativa seria fer 2 passades al vídeo: reconeixement a la primera i correcció de prediccions incorrectes aïllades a la segona.

Finalment, destacar un factor que no seria determinant en la fiabilitat del sistema, però sí en el temps d'entrenament. Entrenar les Xarxes Neuronals Convolucionals amb la GPU en lloc de la CPU (amb la plataforma CUDA) podria accelerar el procés d'entrenament unes 10 vegades [46]. En aquest projecte he entrenat les CNNs amb la CPU i una RAM de 16 GB, principalment perquè els datasets d'aprenentatge són relativament petits, les xarxes no són excessivament grans, i a més la mida del batch ha de ser molt gran en l'entrenament amb *triplet loss* (consum aproximat de 12 GB de memòria), excedint així la memòria de la GPU (3 GB en el cas concret del meu ordinador de sobretaula). Quedaria com a treball futur utilitzar GPUs per accelerar el procés. Per fer-ho, es podria reduir la mida del batch, tot i que caldria veure si això afectaria a la fiabilitat del sistema al poder crear menys tripletes. Una altra opció seria comprar una GPU amb més memòria, una possibilitat molt remota per l'increment en el pressupost que suposaria.

Índex de figures

Índex de taules

Bibliografia

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015. [Online]. Available: https://www.evl.uic.edu/creativecoding/courses/cs523/slides/week3/DeepLearning_LeCun.pdf.

[2] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823, 2015. [Online]. Available: https://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Schroff_FaceNet_A_Unified_2015_CVPR_paper.pdf.

[3] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1701–1708, 2014. [Online]. Available: https://www.cv-foundation.org/openaccess/content_cvpr_2014/papers/Taigman_DeepFace_Closing_the_2014_CVPR_paper.pdf.

[4] B. Amos, B. Ludwiczuk, M. Satyanarayanan, *et al.*, "Openface: A general-purpose face recognition library with mobile applications," *CMU School of Computer Science*, 2016. [Online]. Available: http://reports-archive.adm.cs.cmu.edu/anon/anon/usr0/ftp/2016/CMU-CS-16-118.pdf.

[5] B. Amos, B. Ludwiczuk, M. Satyanarayanan, *et al.*, "Openface," *GitHub repository*. [Online]. Available: https://github.com/cmusatyalab/openface. [Accessed Oct. 7, 2018].

[6] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1, pp. I–I, IEEE, 2001. [Online]. Available: https://www.researchgate.net/profile/Michael_Jones20/publication/3940582_Rapid_Object_Detection_using_a_Boosted_Cascade_of_Simple_Features/links/0f31753b419c639337000000.pdf.

[7] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, pp. 886–893, IEEE, 2005. [Online]. Available: https://hal.inria.fr/docs/00/54/85/12/PDF/hog_cvpr2005.pdf.

[8] V. Kazemi and J. Sullivan, "One millisecond face alignment with an ensemble of regression trees," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1867–1874, 2014. [Online]. Available: https://www.cv-foundation.org/openaccess/content_cvpr_2014/papers/Kazemi_One_Millisecond_Face_2014_CVPR_paper.pdf.

[9] J. Huang, "Accelerating ai with gpus: A new computing model," Jan. 12, 2016. [Online]. Available: https://blogs.nvidia.com/blog/2016/01/12/accelerating-ai-artificial-intelligence-gpus/. [Accessed Jul. 19, 2018].

[10] T. Dettmers, "Deep learning in a nutshell: Core concepts," Nov. 3, 2015. [Online]. Available: https://devblogs.nvidia.com/deep-learning-nutshell-core-concepts/. [Accessed Apr. 7, 2018].

[11] M. Malik, "Basics of neural network," Apr. 3, 2018. [Online]. Available: https://becominghuman.ai/basics-of-neural-network-bef2ba97d2cf. [Accessed Apr. 7, 2018].

[12] A. Ng, "Support vector machines," *Lecture notes*. [Online]. Available: http://cs229.stanford.edu/notes/cs229-notes3.pdf.

[13] J. Huang, "What the waterfall project management methodology can (and can't) do for you," Aug. 23, 2017. [Online]. Available: https://www.lucidchart.com/blog/waterfall-project-management-methodology. [Accessed Mar. 4, 2018].

[14] P. Kovesi, "Decision trees, random forests and random ferns," [Online]. Available: http://cvpr11.cecs.anu.edu.au/files/RandomForests.pdf. [Accessed Mar. 18, 2018].

[15] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org.

[16] S. University, "Convolutional neural networks (cnns / convnets)," [Online]. Available: http://cs231n.github.io/convolutional-networks/. [Accessed Mar. 14, 2018].

[17] J. Yang, "Relu and softmax activation functions," [Online]. Available: https://github.com/Kulbear/deep-learning-nano-foundation/wiki/ReLU-and-Softmax-Activation-Functions. [Accessed Apr. 25, 2018].

[18] M. Lin, Q. Chen, and S. Yan, "Network in network," 2013. [Online]. Available: https://arxiv.org/pdf/1312.4400.

[19] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015. [Online]. Available: https://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Szegedy_Going_Deeper_With_2015_CVPR_paper.pdf.

[20] G. Sanderson, "Neural networks," *YouTube playlist*. [Online]. Available: https://www.youtube.com/watch?v=aircAruvnKk&list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi.

[21] S. Ruder, "An overview of gradient descent optimization algorithms," Jan. 19, 2016. [Online]. Available: http://ruder.io/optimizing-gradient-descent/. [Accessed Jun. 22, 2018].

[22] T. Gupta, "Deep learning: Overfitting," [Online]. Available: https://towardsdatascience.com/deep-learning-overfitting-846bf5b35e24. [Accessed Jul. 2, 2018].

[23] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014. [Online]. Available: http://www.jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf.

[24] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015. [Online]. Available: https://arxiv.org/pdf/1502.03167.pdf.

[25] "Google images," https://images.google.com/.

[26] "Face detection using haar cascades," *OpenCV documentation*. [Online]. Available: https://docs.opencv.org/3.4.2/d7/d8b/tutorial_py_face_detection.html. [Accessed May. 13, 2018].

[27] A. Rosebrock, "Histogram of oriented gradients and object detection," Nov. 10, 2014. [Online]. Available: https://www.pyimagesearch.com/2014/11/10/histogram-oriented-gradients-object-detection/. [Accessed Sept. 12, 2018].

[28] "Face detection," *Dlib documentation*. [Online]. Available: http://dlib.net/face_detector.py.html. [Accessed Feb. 27, 2018].

[29] "Face landmark detection," *Dlib documentation*. [Online]. Available: http://dlib.net/face_landmark_detection.py.html. [Accessed Mar. 21, 2018].

[30] "Models and accuracies," *Openface documentation*. [Online]. Available: https://cmusatyalab.github.io/openface/models-and-accuracies/. [Accessed Mar. 8, 2018].

[31] J. R. A. Moniz, C. Beckham, S. Rajotte, S. Honari, and C. Pal, "Unsupervised depth estimation, 3d face rotation and replacement," 2018. [Online]. Available: https://arxiv.org/pdf/1803.09202.pdf.

[32] "Face alignment," *Openface documentation*. [Online]. Available: https://openface-api.readthedocs.io/en/latest/_modules/openface/align_dlib.html. [Accessed Mar. 22, 2018].

[33] "Geometric image transformations," *OpenCV documentation*. [Online]. Available: https://docs.opencv.org/2.4/modules/imgproc/doc/geometric_transformations.html. [Accessed Mar. 22, 2018].

[34] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[35] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*, pp. 818–833, Springer, 2014. [Online]. Available: https://arxiv.org/pdf/1311.2901.

[36] "Loss functions," *ML Cheatsheet*. [Online]. Available: https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html. [Accessed Mar. 10, 2018].

[37] V. Bishaev, "Stochastic gradient descent with momentum," Dec. 4, 2017. [Online]. Available: https://towardsdatascience.com/stochastic-gradient-descent-with-momentum-a84097641a5d. [Accessed Sept. 16, 2018].

[38] "Training a classifier," *Pytorch documentation*. [Online]. Available: https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html. [Accessed Mar. 10, 2018].

[39] Pytorch, "Pytorch documentation," [Online]. Available: https://pytorch.org/docs/stable/index.html. [Accessed Oct. 2, 2018].

[40] W. Liu, Y. Wen, Z. Yu, and M. Yang, "Large-margin softmax loss for convolutional neural networks.," in *ICML*, pp. 507–516, 2016. [Online]. Available: http://www.jmlr.org/proceedings/papers/v48/liud16.pdf.

[41] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *null*, pp. 1735–1742, IEEE, 2006. [Online]. Available: https://www.researchgate.net/profile/Yann_Lecun/publication/4246277_Dimensionality_Reduction_by_Learning_an_Invariant_Mapping/links/00b7d514af9f25ecca000000/Dimensionality-Reduction-by-Learning-an-Invariant-Mapping.pdf.

[42] Torch, "Torch neural network package," *GitHub repository*. [Online]. Available: https://github.com/torch/nn. [Accessed Sept. 28, 2018].

[43] "Support vector machines," *Scikit-Learn documentation*. [Online]. Available: http://scikit-learn.org/stable/modules/svm.html. [Accessed Apr. 3, 2018].

[44] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," in *Advances in neural information processing systems*, pp. 3320–3328, 2014. [Online]. Available: http://papers.nips.cc/paper/5347-how-transferable-are-features-in-deep-neural-networks.pdf.

[45] G. B. Huang, M. Mattar, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database forstudying face recognition in unconstrained environments," in *Workshop on faces in'Real-Life'Images: detection, alignment, and recognition*, 2008. [Online]. Available: https://hal.inria.fr/docs/00/32/19/23/PDF/Huang_long_eccv2008-lfw.pdf.

[46] A. Alonso, "Gpu computer: How much faster is it for deep learning?," [Online]. Available: https://www.linkedin.com/pulse/gpu-computer-how-much-faster-deep-learning-alejandro-alonso. [Accessed Mar. 23, 2018].

Apèndix A

Matrius de confusió

Una matriu de confusió permet visualitzar el rendiment d'un algorisme de classificació, en aquest cas concret la classificació d'actors. Es tracta d'una matriu quadrada, on les files representen les classes reals i les columnes les predites, o viceversa. Seria per tant una matriu de mida 70x70, però per qüestions d'espai he decidit representar-la d'una forma alternativa: per cada actor (classe real), es representa per files l'exactitud de les prediccions realitzades durant el test de validació, i les prediccions errònies si n'hi ha (confusions).

A.1 Classificació d'actors I (84.29%)

Real Class	Prediction Accuracy	Bad predictions
Aaron Paul	80%	Zach Galifianakis
Adam Sandler	60%	Bradley Cooper, Ryan Gosling
Alexandra Daddario	80%	Ellen Page
Amy Adams	100%	
Angelina Jolie	80%	Amy Adams
Arnold Schwarzenegger	100%	
Ben Affleck	60%	James Franco, Robert Downey Jr
Benedict Cumberbatch	80%	Ben Stiller
Ben Stiller	80%	Tom Cruise
Bradley Cooper	100%	
Brad Pitt	80%	Dwayne Johnson
Bruce Willis	80%	Jason Statham
Bryan Cranston	100%	
Cameron Diaz	80%	Kristen Bell
Cara Delevigne	100%	
Channing Tatum	80%	Arnold Schwarzenegger
Christopher Walken	100%	
Christoph Waltz	80%	Daniel Radcliffe
Claire Danes	40%	Scarlett Johansson, Kate Mara, Joseph Gordon-Levitt
Dakota Johnson	60%	Angelina Jolie, Margot Robbie
Daniel Craig	100%	
Daniel Radcliffe	100%	
Denzel Washington	80%	Jason Statham
Dwayne Johnson	100%	
Ellen Page	80%	Angelina Jolie
Emma Stone	100%	
Emma Watson	100%	
George Clooney	80%	Vin Diesel
Haley Joel Osment	100%	
Harrison Ford	40%	Brad Pitt, Kevin Spacey, Viggo Mortensen
Jackie Chan	80%	Tom Hanks
Jake Gyllenhaal	100%	
James Franco	80%	Jake Gyllenhaal
January Jones	60%	Kristen Bell, Jennifer Lawrence
Jason Statham	40%	Denzel Washington, Dwayne Johnson, Christoph Waltz
Jennifer Aniston	100%	
Jennifer Lawrence	80%	Kristen Bell
Joseph Gordon-Levitt	100%	
Julianne Moore	80%	Amy Adams
Julia Roberts	100%	
Kate Mara	80%	Julianne Moore
Kevin Costner	80%	Bruce Willis
Kevin Spacey	80%	Vin Diesel
Kristen Bell	80%	Jake Gyllenhaal
Laurence Fishburne	80%	Will Smith
Leonardo DiCaprio	100%	
Liam Neeson	80%	Zach Galifianakis
Margot Robbie	100%	
Mark Ruffalo	80%	Adam Sandler
Matt Damon	100%	
Meryl Streep	80%	Christoph Waltz
Miles Teller	100%	
Morgan Freeman	80%	Denzel Washington
Nicolas Cage	100%	
Nicole Kidman	80%	Amy Adams
Octavia Spencer	100%	
Penelope Cruz	100%	
Robert De Niro	100%	
Robert Downey Jr	80%	Leonardo DiCaprio
Ryan Gosling	60%	Aaron Paul, Harrison Ford
Sandra Bullock	100%	
Scarlett Johansson	80%	Dakota Johnson
Sylvester Stallone	80%	Sandra Bullock
Tom Cruise	100%	
Tom Hanks	100%	
Viggo Mortensen	100%	
Vin Diesel	40%	Sylvester Stallone, Cara Delevingne, Tom Hanks
Will Smith	60%	Dwayne Johnson, Jennifer Lawrence
Zac Efron	80%	Ben Stiller
Zach Galifianakis	100%	
Total	**84.28571%**	

Taula A.1: Matriu de confusió de la classificació d'actors (I).

A.2 Classificació d'actors II (93.43%)

Real Class	Prediction Accuracy	Bad predictions
Aaron Paul	100%	
Adam Sandler	80%	Bradley Cooper
Alexandra Daddario	80%	Emma Stone
Amy Adams	100%	
Angelina Jolie	80%	Amy Adams
Arnold Schwarzenegger	100%	
Ben Affleck	100%	
Benedict Cumberbatch	100%	
Ben Stiller	100%	
Bradley Cooper	100%	
Brad Pitt	80%	Robert Downey Jr
Bruce Willis	100%	
Bryan Cranston	100%	
Cameron Diaz	100%	
Cara Delevigne	100%	
Channing Tatum	60%	Tom Hanks, Kevin Costner
Christopher Walken	100%	
Christoph Waltz	80%	Daniel Radcliffe
Claire Danes	80%	Scarlett Johansson
Dakota Johnson	80%	Penelope Cruz
Daniel Craig	100%	
Daniel Radcliffe	100%	
Denzel Washington	80%	Matt Damon
Dwayne Johnson	100%	
Ellen Page	100%	
Emma Stone	80%	Cara Delevingne
Emma Watson	100%	
George Clooney	80%	Mark Ruffalo
Haley Joel Osment	100%	
Harrison Ford	80%	Daniel Craig
Jackie Chan	100%	
Jake Gyllenhaal	80%	Tom Cruise
James Franco	80%	Benedict Cumberbatch
January Jones	80%	Scarlett Johansson
Jason Statham	80%	Daniel Craig
Jennifer Aniston	100%	
Jennifer Lawrence	100%	
Joseph Gordon-Levitt	100%	
Julianne Moore	100%	
Julia Roberts	100%	
Kate Mara	100%	
Kevin Costner	100%	
Kevin Spacey	100%	
Kristen Bell	100%	
Laurence Fishburne	100%	
Leonardo DiCaprio	100%	
Liam Neeson	80%	Bryan Cranston
Margot Robbie	100%	
Mark Ruffalo	80%	Sylvester Stallone
Matt Damon	100%	
Meryl Streep	100%	
Miles Teller	100%	
Morgan Freeman	100%	
Nicolas Cage	100%	
Nicole Kidman	100%	
Octavia Spencer	100%	
Penelope Cruz	100%	
Robert De Niro	100%	
Robert Downey Jr	100%	
Ryan Gosling	80%	Nicolas Cage
Sandra Bullock	100%	
Scarlett Johansson	100%	
Sylvester Stallone	100%	
Tom Cruise	100%	
Tom Hanks	100%	
Viggo Mortensen	100%	
Vin Diesel	80%	Meryl Streep
Will Smith	80%	Octavia Spencer
Zac Efron	80%	Tom Cruise
Zach Galifianakis	100%	
Total	**93.42857%**	

Taula A.2: Matriu de confusió de la classificació d'actors (II).

A.3 Classificació d'actors III.I (95.43%)

Real Class	Prediction Accuracy	Bad predictions
Aaron Paul	100%	
Adam Sandler	100%	
Alexandra Daddario	100%	
Amy Adams	100%	
Angelina Jolie	100%	
Arnold Schwarzenegger	100%	
Ben Affleck	100%	
Benedict Cumberbatch	100%	
Ben Stiller	100%	
Bradley Cooper	100%	
Brad Pitt	80%	Zach Galifianakis
Bruce Willis	80%	Bradley Cooper
Bryan Cranston	100%	
Cameron Diaz	100%	
Cara Delevigne	100%	
Channing Tatum	80%	Joseph Gordon-Levitt
Christopher Walken	100%	
Christoph Waltz	80%	Joseph Gordon-Levitt
Claire Danes	100%	
Dakota Johnson	100%	
Daniel Craig	100%	
Daniel Radcliffe	100%	
Denzel Washington	80%	Will Smith
Dwayne Johnson	100%	
Ellen Page	100%	
Emma Stone	80%	Margot Robbie
Emma Watson	100%	
George Clooney	100%	
Haley Joel Osment	100%	
Harrison Ford	100%	
Jackie Chan	100%	
Jake Gyllenhaal	80%	Tom Cruise
James Franco	100%	
January Jones	60%	Kristen Bell, Jennifer Lawrence
Jason Statham	100%	
Jennifer Aniston	100%	
Jennifer Lawrence	80%	Cameron Diaz
Joseph Gordon-Levitt	100%	
Julianne Moore	100%	
Julia Roberts	100%	
Kate Mara	100%	
Kevin Costner	100%	
Kevin Spacey	60%	Bryan Cranston, Tom Hanks
Kristen Bell	100%	
Laurence Fishburne	100%	
Leonardo DiCaprio	100%	
Liam Neeson	80%	Bradley Cooper
Margot Robbie	60%	Julianne Moore, Jennifer Lawrence
Mark Ruffalo	100%	
Matt Damon	100%	
Meryl Streep	100%	
Miles Teller	100%	
Morgan Freeman	100%	
Nicolas Cage	100%	
Nicole Kidman	100%	
Octavia Spencer	100%	
Penelope Cruz	100%	
Robert De Niro	100%	
Robert Downey Jr	100%	
Ryan Gosling	100%	
Sandra Bullock	100%	
Scarlett Johansson	100%	
Sylvester Stallone	100%	
Tom Cruise	100%	
Tom Hanks	100%	
Viggo Mortensen	100%	
Vin Diesel	100%	
Will Smith	100%	
Zac Efron	80%	Tom Cruise
Zach Galifianakis	100%	
Total	**95.42857%**	

Taula A.3: Matriu de confusió de la classificació d'actors (III.I).

A.4 Classificació d'actors III.II (98.29%)

Real Class	Prediction Accuracy	Bad predictions
Aaron Paul	100%	
Adam Sandler	100%	
Alexandra Daddario	100%	
Amy Adams	100%	
Angelina Jolie	100%	
Arnold Schwarzenegger	100%	
Ben Affleck	100%	
Benedict Cumberbatch	100%	
Ben Stiller	100%	
Bradley Cooper	100%	
Brad Pitt	80%	Bruce Willis
Bruce Willis	100%	
Bryan Cranston	100%	
Cameron Diaz	100%	
Cara Delevigne	100%	
Channing Tatum	100%	
Christopher Walken	100%	
Christoph Waltz	80%	Miles Teller
Claire Danes	100%	
Dakota Johnson	100%	
Daniel Craig	100%	
Daniel Radcliffe	100%	
Denzel Washington	80%	Robert Downey Jr
Dwayne Johnson	100%	
Ellen Page	100%	
Emma Stone	100%	
Emma Watson	100%	
George Clooney	100%	
Haley Joel Osment	100%	
Harrison Ford	100%	
Jackie Chan	100%	
Jake Gyllenhaal	100%	
James Franco	100%	
January Jones	80%	Margot Robbie
Jason Statham	100%	
Jennifer Aniston	100%	
Jennifer Lawrence	100%	
Joseph Gordon-Levitt	100%	
Julianne Moore	100%	
Julia Roberts	100%	
Kate Mara	100%	
Kevin Costner	100%	
Kevin Spacey	100%	
Kristen Bell	100%	
Laurence Fishburne	100%	
Leonardo DiCaprio	100%	
Liam Neeson	80%	Jason Statham
Margot Robbie	100%	
Mark Ruffalo	100%	
Matt Damon	100%	
Meryl Streep	100%	
Miles Teller	100%	
Morgan Freeman	100%	
Nicolas Cage	100%	
Nicole Kidman	100%	
Octavia Spencer	100%	
Penelope Cruz	100%	
Robert De Niro	100%	
Robert Downey Jr	100%	
Ryan Gosling	100%	
Sandra Bullock	100%	
Scarlett Johansson	100%	
Sylvester Stallone	100%	
Tom Cruise	100%	
Tom Hanks	100%	
Viggo Mortensen	100%	
Vin Diesel	80%	Mark Ruffalo
Will Smith	100%	
Zac Efron	100%	
Zach Galifianakis	100%	
Total	**98.28571%**	

Taula A.4: Matriu de confusió de la classificació d'actors (III.II).

Apèndix B

Glossari

Característica de Haar Característica molt utilitzada en la detecció d'objectes, que s'obté mitjançant la diferència d'intensitat de píxels.

Viola-Jones Algorisme de detecció de cares basat en les característiques de Haar.

Histograma de Gradients orientats Descriptor de característiques molt utilitzat en la detecció d'objectes, que s'obté comptant les ocurrències de la orientació del gradient en petites porcions d'una imatge.

Fern Arbre de regressió binari en què es fa el mateix test binari a tots els nodes del mateix nivell.

Support Vector Machine (SVM) Algorisme d'aprenentatge automàtic supervisat que té com a objectiu la tasca de classificació.

Xarxa Neuronal Profunda Model que té com a objectiu el reconeixement de patrons. Està format per una capa d'entrada, una de sortida, i vàries d'ocultes. Cadascuna d'elles es compon per neurones, les quals estan connectades amb altres capes a través de pesos.

Xarxa Neuronal Convolucional (CNN) Xarxa Neuronal Profunda que permet simular algunes de les accions del còrtex visual humà. Es caracteritza per tenir capes convolucionals, i una imatge a la capa d'entrada.

Rectified Linear Unit (ReLU) Funció d'activació $f(x) = max(0, x)$, que permet que una CNN aprengui a extreure característiques més complexes i s'entreni més de pressa.

Softmax Funció d'activació aplicada a la capa de sortida d'una CNN per convertir-la en una distribució probabilística sobre classes.

Loss Error d'una CNN.

Cross-Entropy Loss *Loss* calculada a partir dels valors de la capa de sortida d'una CNN per un únic exemple d'entrada.

Triplet Loss *Loss* calculada a partir dels valors de la capa de sortida d'una CNN per tres exemples d'entrada (2 de la mateixa classe i 1 de diferent).

Back-propagation Tècnica utilitzada per entrenar una CNN, que permet calcular el gradient per cadascun dels paràmetres respecte la *Loss*.

Stochastic Gradient Descent (SGD) Optimitzador que canvia els paràmetres d'una CNN en funció dels gradients obtinguts per *back-propagation*, per tal de minimitzar la *Loss* total. En aquest treball, quan menciono SGD em refereixo a *Minibatch Gradient Descent*.

Adaptative Moment Estimation (Adam) Algorisme d'optimització que canvia els paràmetres d'una CNN en funció dels gradients obtinguts per *back-propagation*, per tal de minimitzar la *Loss* total. Es caracteritza per *learning rates* adaptatius per cada paràmetre.

Learning rate Ritme al qual es modifiquen els paràmetres de la CNN.

Embedding Vector de 128 característiques que extret de la imatge d'una cara, que representa un punt d'una hiperesfera unitat de 128 dimensions.

Batch size Determina el número de mostres del conjunt d'aprenentatge que passaran per la CNN abans d'actualitzar els paràmetres. Si un batch té una mida entre 1 i la mida del conjunt d'aprenentatge (ambdós exclosos), s'anomena minibatch.

Epoch size La mida d'un epoch determina el número de minibatches que s'executaran abans de fer un test de validació.

Training Loss Resultat de la funció *Loss* obtingut amb les imatges del conjunt d'aprenentatge.

Validation Accuracy Percentatge de mostres del conjunt d'imatges de validació que han estat classificades correctament.

Overfitting Succeeix quan la CNN classifica molt satisfactòriament les imatges d'aprenentatge, però no sap generalitzar a imatges que no ha vist durant l'entrenament, donant uns resultats de validació pobres.

Dropout Tècnica de regularització que, apagant algunes neurones durant l'entrenament, és capaç de reduir l'overfitting d'una CNN.

Batch Normalization Tècnica de normalització que a la vegada té efectes de regularització, de gran utilitat per combatre l'overfitting.

Labeled Faces in the Wild (LFW) Conjunt d'imatges de cares dissenyat per validar sistemes de reconeixement facial.