Universitat de Girona
Escola Politècnica Superior

Projecte fi de grau

Estudi: Grau Enginyeria Informàtica

Títol: Desenvolupament d'un sistema de reconeixement facial a partir d'un conjunt reduït d'imatges

Document: Memòria

Alumne: Sergi Felip Ribas

Tutor: Dr. Rafael Garcia Campos
Departament: Arquitectura i tecnologia de computadors
Àrea: Arquitectura i tecnologia de computadors

Convocatòria (mes / any): Juny 2022

Universitat de Girona
Escola Politècnica Superior

PROJECTE FI DE GRAU

Desenvolupament d'un sistema de reconeixement facial a partir d'un conjunt reduït d'imatges

Autor:
Sergi Felip Ribas

Juny 2022

Grau en Enginyeria Informàtica

Tutor:
Dr. Rafael Garcia Campos

Resum

En plena era de la informació on el Big Data és mostra en un ascens imparable, l'avenç tecnològic s'ha vist incrementat en potència de càlcul per processar la gran quantitat de dades existent. Aquesta nova era tecnològica ha comportat que mètodes abans obsolets per falta de capacitat de còmput hagin guanyat actualitat en recerca i desenvolupament. En conseqüència la tecnologia està cada cop més present al nostre dia a dia. Usuaris i institucions generen un gran nombre de dades sotmeses cada cop a més amenaces. Una de les solucions per protegir les dades són els sistemes de control d'accés. Els sistemes de control d'accés en conjunt amb la biometria permeten crear sistemes de reconeixement dactilar, reconeixement de veu o reconeixement facial amb els quals protegir espais i dades.

Aquest projecte se centra en crear un sistema de control d'accés biomètric via reconeixement facial a partir de tècniques d'aprenentatge profund on les dades en forma de Big Data s'utilitzen per crear algoritmes capaços d'aprendre i predir noves dades mai vistes pel sistema.

Els esforços se centren en l'aprenentatge de les tècniques existents, la selecció d'una arquitectura d'aprenentatge profund òptima, la parametrització i entrenament del mateix model i l'ús adequat d'aquest model per resoldre el problema del reconeixement facial a partir d'un conjunt d'imatges reduït com podria ser pel cas d'accés a empresa.

Agraïments

Vull agrair a totes aquelles persones que han contribuït a la realització d'aquest projecte ja sigui de forma directa o indirecta. En primer lloc, al meu tutor de projecte Rafael Garcia, per introduir-me al món laboral, formar-me i guiar-me en aquest projecte. En segon lloc aquells amics i companys de feina que s'han ofert a col·laborar al projecte ajudant-me tant en la fase d'aprenentatge del marc teòric com en la recollida de dades i posada en marxa del sistema en el marc pràctic. Finalment agrair a la meva família, amics i companys que m'han fet arribar fins aquí i sense els quals aquest projecte no seria possible.

Gracies a tots ells.

Sergi.

Índex

Índex d'Il·lustracions

CAPÍTOL 1

Introducció

La societat conviu en un flux massiu i constant de dades on la privacitat d'aquestes juga un paper cada cop més important. La seguretat de dades i el control d'accés tenen un estret vincle. Moltes de les dades que compartim a través d'Internet deixen de ser completament nostres i permeten extreure, via anàlisi de dades, molta més informació de la que ens pensem que estem proporcionant.
Durant molt de temps s'ha pensat que la manera de protegir la informació mitjançant contrasenyes era suficient. Les contrasenyes però, no han estat suficients per parar els nombrosos tipus d'atacs i tècniques d'intrusió que deixen en evidència la limitació d'aquesta tècnica [1]. És en aquest marc on el control d'accés pren més importància.

El control d'accés és pot definir com un mecanisme o dispositiu que autoritza l'entrada a determinades instal·lacions, dispositius, serveis o dades. Amb el pas del temps i d'acord amb les noves necessitats, els sistemes de control d'accés han anat evolucionant tecnològicament des dels sistemes més rudimentaris coms els murs medievals fins als avançats sistemes de reconeixement facial integrant conceptes de visió per computador i intel·ligència artificial.

Per a qualsevol persona o institució pública o privada és essencial trobar una forma senzilla i eficaç de gestionar el control d'accés a diferents recintes. Per exemple una gran empresa rep milers de visitants que han de ser organitzats de manera diferent en cada un dels seus espais. En un altre exemple a menor escala una persona pot veure alterada la seva seguretat a la llar amb situacions conflictives com robatoris per culpa d'ingressos no autoritzats.

Els objectius del control d'accés son en definitiva la identificació, l'autentificació i la autorització en aquest ordre. La identificació és el primer pas que realitza un sistema de control. Es basa en identificar via detectors (d'empenta dactilar, reconeixement veu / facial, targeta, etc.), la persona que intenta accedir. L'autenticació és el pas posterior a la identificació que consisteix en detectar si l'accés està autoritzat i té els permisos necessaris. L'autorització és el pas final consistent en procedir a donar resposta a la petició d'accés.

Fem un breu repàs als sistemes de control d'accés actuals [2]:

- Targetes identificatives: Utilitzen targetes o altres elements que és llegeixen al inserir-se a un terminal i permeten així l'autorització. La majoria d'espais utilitzen sistemes de control d'accés via targetes identificatives, no obstant, el propi procés pot ser insegur pel simple fet de ser realitzat per una altra persona en cas de pèrdua o robatori.

- Sistemes de proximitat RFID: La identificació per radiofreqüència, o RFID és una tecnologia segura, precisa, fiable i amb gran capacitat d'emmagatzematge de dades. RFID utilitza les ones de ràdio per comunicar-se amb un microxip. A partir de les dades transmeses és garanteix l'autorització. Com amb les targetes identificatives, el procés és vulnerable a suplantacions.

- Autònoms: Els sistemes de control d'accés autònoms són aquells que, sense necessitat d'estar connectats a un equip central o xarxa, permeten l'autorització, per exemple, els sistemes de control via codi numèric introduït a través d'un teclat. Com a contrapartida, no poden emmagatzemar dades com els esdeveniments de control d'accés o limitar a grups o usuaris.

- En xarxa: Aquests sistemes de control utilitzen equips centrals, locals o remots, per gestionar totes les vies d'accés a instal·lacions o serveis informàtics registrant cada esdeveniment amb molta informació i precisió.

- Biomètrics: Es tracta d'un sistema de control d'accés que permet actuar amb dades físiques extretes a temps real afegint un nou nivell de seguretat en qualsevol sistema. La biometria parteix de la unicitat de cada persona a nivell físic per identificar-la inequívocament. Amb aquest tipus d'accés no cal que l'usuari disposi de cap dispositiu físic que és pugui perdre o robar.
  Les característiques biofísiques, com les dades facials o d'empremta dactilar, permeten construir sistemes de control d'accés a partir de dades identificaries úniques en cada usuari.

El principal eix del projecte és l'elaboració d'un sistema de control d'accés biomètric via reconeixement facial a temps real. A través d'una sola fotografia el sistema entrenat via aprenentatge profund serà capaç de reconèixer la persona que té al davant gràcies a l'extracció i classificació de característiques facials úniques.

Una característica fonamental d'aquest projecte respecte la resta és que alhora d'utilitzar aprenentatge profund, utilitza una molt petita quantitat de dades de cada usuari per a resoldre el reconeixement facial. Concretament s'utilitzarà un conjunt reduït d'imatges per a cada persona, de tan sols 25 imatges, i una xarxa neuronal convolucional per extreure informació característica de les imatges de cada usuari.

El resultat final que ens aporta el Deep Learning pot tenir multitud d'aplicacions al món real que seran analitzades al llarg del projecte, sovint moltes d'elles d'un gran cost econòmic. Es per això que l'objectiu del projecte és també mostrar la potència del Deep Learning treballant amb conjunts de dades reduïts, construint un sistema capaç de reconèixer persones de forma eficaç, sense un gran cost tecnològic i alhora sense un elevat cost econòmic.

El projecte no pretén aportar un mètode únic de protecció a espais, serveis o dades sinó que aporta una nova solució útil i de baix cost apte per a incorporar en conjunt a altres sistemes de protecció.

Actualment el mercat ofereix diferents dispositius que identifiquen les persones via dades facials. No obstant el pressupost d'aquest tipus de dispositius és alt i fàcilment supera els 1000 euros en qualsevol cas.

Figura 1. Dispositiu de control d'accés Face-Temp-T [3]

Sorgeix per tant la necessitat de crear un mètode de control d'accés de baix cost, gran eficiència i d'actualitat tecnològica. Els mètodes d'intel·ligència artificial en conjunt amb mètodes de visió per computador ofereixen aquesta solució que estudiem i apliquem al llarg del projecte.

CAPÍTOL 2

Motivació

La motivació per realitzar aquest projecte neix per l'interès personal alhora d'estudiar coneixements de Machine Learning a nivell teòric i experimental.
El projecte em suposa un punt final en la meva etapa com estudiant i una obertura al món laboral destacant per haver obtingut coneixement en un camp cada com més present i amb més recerca i innovació.

Al projecte exposo de forma introductòria conceptes d'intel·ligència artificial i Machine Learning per acabar parlant a nivell expert de l'arquitectura CNN que em permet desenvolupar una aplicació capaç d'extreure característiques facials amb molt poques imatges i capaç de dur a terme el propi reconeixement de persones tal com un sistema de vigilància podria fer usant aquesta tecnologia.

Aquest treball em suposa doncs un primer pas dins l'immens mon del Machine Learning i en concret, la seva aplicació real en un cas de seguretat.

CAPÍTOL 3

Objectius

3.1 Objectiu general

Elaboració d'un software de reconeixement facial via aprenentatge profund utilitzant un conjunt d'imatges reduït de tan sols 25 imatges per persona amb el propòsit de detectar i identificar correctament usuaris sense un gran cost computacional.

3.2 Objectius específics

La totalitat del projecte es divideix en definitiva en els següents objectius específics:

- Estudi de les tècniques de reconeixement facial identificant la de millor propòsit per a la realització del projecte.
- Estudi de les tècniques d'aprenentatge profund.
- Identificació i parametrització de l'arquitectura d'aprenentatge profund adequada.
- Entrenament del model d'aprenentatge profund per l'extracció útil de característiques facials.
- Selecció del classificador més adequat segons la informació extreta pel model entrenat.
- Utilització del millor model i classificador obtingut per a un sistema de reconeixement facial a temps real.

3.3 Producte final

El producte final obtingut és un sistema de reconeixement facial en temps real resultat d'entrenar un model d'aprenentatge profund amb tan sols 25 imatges per persona. El software final obtingut permet reconèixer i alhora afegir nous usuaris per a al seu reconeixement sense necessitat de reentrenar el model.

CAPÍTOL 4

Marc de treball i conceptes previs

Abans d'aventurar-nos a la part pràctica del projecte i a fi d'entendre tot el procediment del mateix, ens cal conèixer tota la part teòrica prèvia. En aquest apartat, s'expliquen de forma clara tots els coneixements adquirits i necessaris per a desenvolupar el projecte.
Es comencen introduint conceptes teòrics de sistemes de control d'accés existents i actuals més usats a fi contextualitzar la localització del projecte, finalment s'introdueixen els conceptes teòrics més avançats d'intel·ligència artificial i concretament la seva branca més destacada i alhora usada al projecte, la branca de l'aprenentatge automàtic o Machine Learning.

4.1 Reconeixement facial

El reconeixement facial és la tecnologia de control d'accés biomètric que utilitza dades facials extretes d'imatges o fotogrames de vídeo com a elements autoritzadors. Avui dia és present en innumerables camps com serien la vigilància, la seguretat, l'entreteniment i la pròpia identificació de persones.

El procés d'identificació via reconeixement facial es divideix en les següents etapes:

- Adquisició: En aquesta etapa s'adquireix al imatge ja sigui d'una fotografia o d'un fotograma de vídeo.
- Detecció: Partint de la imatge posterior és detecta la cara. Cal remarcar que detecció i reconeixement facial són conceptes totalment diferents i que és en aquest últim on és centra el marc del projecte.
- Pre processat: Un cop detectat el rostre facial, es procedeix al seu pre processat per acomodar la imatge a l'entrada esperada pel posterior algoritme d'extracció de característiques.
- Extracció: Els algoritmes d'extracció de característiques reben les imatges facials i n'extreuen les característiques facials úniques per cada usuari.
- Reconeixement: És en aquesta fase on es duu a terme la classificació de l'usuari a partir de les característiques facials extretes anteriorment i que identifiquen la identitat de la persona en qüestió.

Per a realitzar el procés d'identificació, actualment s'utilitzen diferents branques de la intel·ligència artificial en funció dels recursos disponibles però no ha estat sempre així. A continuació revisarem els algoritmes més destacats al camp del reconeixement facial.

4.2 Estat de l'art del reconeixement facial

A continuació és fa un breu repàs a la història del reconeixement facial remarcant els successos i algoritmes desenvolupats més importants [4].

- Marcadors facials: El 1960 Woodrow Wilson Bledsoe desenvolupa un sistema basat en marcadors facials computats de forma manual. A través d'aquests marcadors i una tableta RAND (dispositiu per entrar coordenades horitzontals i verticals a través d'un llapis òptic en una quadrícula), s'associaven les dades extretes a un nom d'individu. Es considera aquesta tecnologia com la pionera del reconeixement facial.

- Major precisió: Una dècada més tard, al 1970, arriben mètodes més precisos que utilitzen marcadors facials específics com el gruix dels llavis, color cabell, etc. per identificar cares automàticament amb poca precisió.

- Arribada de l'àlgebra lineal: L'arribada de l'àlgebra lineal al camp del reconeixement facial al 1980 suposa un avenç per al desenvolupament del primer algoritme de reconeixement facial conegut, Eigenfaces [5]. L'algoritme és basava en capturar informació facial rellevant a través de la variació estadística entre imatges.

- FERET: A fi de fomentar l'ús comercial del reconeixement facial, l'institut nacional d'estàndards tecnològics impulsa el programa FERET al 1990. S'aconsegueix una base de dades d'imatges facials de 856 persones amb l'esperança d'inspirar noves investigacions en el camp.

- Xarxes socials: L'impuls de la xarxa social Facebook fa que la mateixa empresa generi una base de dades etiquetada a partir dels propis usuaris de la xarxa social. Alhora és genera un gran debat polític i social sobre la privacitat de dades. Al 2014 la companyia desenvolupa DeepFace, un sistema de reconeixement facial que integra intel·ligència artificial al reconeixement facial mitjançant una xarxa neuronal artificial de 9 capes amb més de 120 milions de connexions i entrenada amb 4 milions d'imatges. El resultat, una tassa d'encert del 97%.

- Iphone: L'avenç tecnològic competitiu present a la indústria del desenvolupament de telèfons mòbils fa generar nous algoritmes de reconeixement. És al 2017 on Apple anuncia Face ID, el seu propi software de reconeixement facial que permet als usuaris desbloquejar els telèfons mòbils automàticament.

- Actualitat: L'actualitat del reconeixement facial és basa en tècniques de Machine Learning per obtenir representacions compactes de característiques facials per a una posterior classificació. El model FaceNet [6] n'és un exemple. Es tracta d'una xarxa neuronal desenvolupada per Google que va aconseguir un nou record en precisió, un 99'63% de precisió al aplicar-se sobre el conjunt Youtube Faces DB.

  A diferència de la xarxa neuronal DeepFace anterior, FaceNet [6] destaca per implementar un nou model d'entrenament anomenat "One Shot Learning". Aquest model en té prou amb un recull molt petit o únic d'imatges d'una sola persona per a classificar-la correctament atès que genera uns vectors característics molt bons per a cada persona a més de distanciar-los convenientment durant l'entrenament del model. Alhora és capaç de reduir la dimensionalitat de dades en vectors característics de 128 elements per cada rostre fent-la ideal per aplicacions de reconeixement facial que treballin fins i tot en temps real.

L'objectiu d'aquest projecte és utilitzar aquest tipus de xarxa neuronal de referència, estudiar-la, utilitzar-la, veure'n les mancances i obtenir-ne els resultats alhora que s'aprenen múltiples i útils conceptes sobre Machine Learning exposats al llarg del projecte.

4.3 Intel·ligència artificial

Per parlar de Machine Learning primer ens cal parlar de la branca de ciències informàtiques en la que s'inclou, la intel·ligència artificial. Les ciències informàtiques defineixen la maquina intel·ligent com un agent flexible que percep l'entorn i executa accions maximitzant les possibilitats d'èxit d'una tasca.

La intel·ligència artificial, generalment coneguda com a (IA), és l'intent de simular via computadors la intel·ligència humana. El seu origen es considera a l'antiguitat quan els humans pretenien per incrementar les seves habilitats, creant automatismes. Actualment, es tracta d'una nova forma de resoldre problemes via sistemes experts capaços d'emmagatzemar i utilitzar coneixement que es traduït en la seva capacitat d'actuació sobre una àrea determinada.
Informàticament podem considerar que les màquines que incorporen IA utilitzen algoritmes capaços de aprendre de les dades i utilitzar l'aprenentatge per a la presa de decisions imitant les capacitats d'un ser humà. Podem trobar exemples d'aplicació de IA en diversos camps com el control de sistemes automàtics, la medicina, el reconeixement de patrons (parlats, escrits, visuals), l'economia, les comunicacions i en gran varietat d'aplicacions i jocs, sobretot d'estratègia.

El ràpid avenç tecnològic que estem vivint durant el segle XXI fa que tecnologies imitadores de funcions cognitives humanes interpretades anteriorment com a IA siguin ara simples funcions avançades tecnològicament. Un clar exemple d'aquesta evolució es el reconeixement visual de caràcters. Per contrapartida, també existeixen tecnologies de IA que no devaluen en classificació com per exemple els sistemes de conducció autònoms (relativament nous) o el sistema de IA capaç de jugar als escacs.

Podem diferenciar els tipus de IA segons la seva actuació en quatre categories [7]:

- Màquines reactives: IA incorporada en màquines simples que actuen en funció d'una o varies entrades. No executen cap procés d'aprenentatge ni emmagatzemen dades per actuar en conseqüència al passat.

- Memòria limitada: IA pensada per a sistemes amb memòria limitada amb capacitat per emmagatzemar dades i utilitzar-les per a millors prediccions.

- Teoria de la ment: IA més avançada que a més d'utilitzar dades passades per a executar millors accions, incorpora el que es coneix com a "teoria de la ment" que implica la comprensió de que els agents del mon tenen pensaments i emocions pròpies que afecten al seu comportament. Per tant no es tracta d'una "IA plana" sinó que incorpora variables per interactuar a raó de les emocions i pensaments humans.

- Autoconsciència: Es tracta de l'últim pas de la IA que està en discussió actualment. Amb aquest tipus de futura IA la pròpia màquina te una representació d'ella mateixa, té consciencia i això ajuda a la presa de decisions via experiències passades, estats interns de la màquina i estats interns dels agents de l'entorn.

Un cop entesos els principals tipus de IA cal introduir les diferents branques existents abans d'introduir-nos al Machine Learning que n'és una de elles.

Les principals branques de la IA son [8]:

- Machine Learning: Es basa en l'entrenament dels algorismes d'aprenentatge a traves d'una gran quantitat de dades perquè passat el temps d'entrenament els algoritmes millorin el seu objectiu.

- Reconeixement visual: S'encarrega de processar imatges o vídeos amb l'objectiu de reconèixer patrons i identificar-los.

- Robòtica: Es centra en la recerca i desenvolupament de disseny robòtic. És desenvolupen robots capaços d'establir interaccions socials.

- Sistemes experts: Referits a aquells sistemes experts en un determinat àmbit que prenen decisions imitant la presa de decisions humanes. Com més informació prèvia, millor és l'eficiència. Un exemple clàssic és el joc dels escacs.

- Processament natural del llenguatge: És una disciplina lligada al camp de la lingüística en la que el propòsit es comprendre la intenció de l'usuari al llençar una comanda, pregunta o afirmació. En resum és un camp que ajuda a la comunicació entre home i màquina.

4.4 Machine Learning

Un cop introduïts al món de la intel·ligència artificial podem començar a parlar en profunditat de la seva branca més destacada, el Machine Learning.

La gran quantitat de dades generades diàriament pels usuaris i la capacitat de còmput actual han portat desenvolupar sistemes per resoldre la necessitat real de processar les dades i extreure'n informació. En aquest marc neix el Machine Learning, per resoldre la problemàtica del tractament manual de dades massives (Big Data) de forma eficient per un ésser humà.

El Machine Learning o aprenentatge automàtic és una branca de la intel·ligència artificial que permet que les màquines aprenguin i millorin el seu objectiu a partir de dades d'entrenament. L'estadística és sens dubte la base fonamental que ajuda al Machine Learning a deduir els resultats més òptims per al programa.
El gran "hit" del Machine Learning ha estat passar de la programació mitjançant regles a l'aprenentatge autònom a partir de dades prèvies. Podem veure reflectida la seva capacitat en motors de cerca, en robòtica, en diagnòstic mèdic o en aplicacions d'estratègia entre d'altres.

El Machine Learning es centra en l'obtenció de grans quantitats de dades per obtenir de forma automàtica o semi-automàtica, un model entrenat capaç d'extreure informació útil per facilitar posteriors tasques d'extracció, representació o classificació de la informació. En termes més precisos, tots els sistemes de Machine Learning busquen generar un model encarregat de rebre i processar grans quantitats de dades i prendre decisions per si mateixos. Fonamentalment el model rep dades, aprèn a processar-les i a donar la sortida que l'usuari espera rebre sobre aquestes.

Tota tècnica existent a la intel·ligència artificial té en comú que intenta emular la intel·ligència que aportaria un esser humà per a la realització d'una tasca. El Machine Learning es diferència de la resta de tècniques de IA per realitzar un aprenentatge actualitzant automàticament els paràmetres del model per donar la resposta apropiada. Avui dia és una de les tècniques més esteses del món gràcies a la immensa quantitat de camps al que és pot aplicar per resoldre problemes de forma ràpida i precisa. Generalment està present a qualsevol camp que treballa amb grans quantitats d'informació.

4.4.1 Perspectiva històrica

Com ja havíem dit anteriorment, la gran quantitat de dades que pot generar un usuari no ha deixat de créixer en els últims anys. Tot això comporta un creixement en el nombre de dades generades, informació que, tractada correctament, pot ser molt útil per als diferents sectors existents.

Pot semblar estrany atribuir la paraula història a un camp que s'ha acabat de consolidar els darrers anys. El motiu pel que sembla estrany és perquè la tecnologia està actualment patint un gran avenç però no ha estat sempre així. Al llarg de la historia de l'aprenentatge automàtic és parla d'èpoques amb grans avenços o d'hiverns estancats.

Fem un repàs als fets històrics que marquen aquesta tecnologia [9]:

- El teorema de Bayes (1763): Es una proposició plantejada pel filòsof Thomas Bayes que expressa la probabilitat condicional que vincula la probabilitat de que un esdeveniment aleatori A condicionat per B amb la probabilitat de B donat A. Es a dir, que coneguda la condició A sabent que tenim B, som capaços de saber la probabilitat de B si es té A

- Programació informàtica (dècada 1940): Quan els científics van establir les bases de la programació informàtica traduint una sèrie d'instruccions en accions executables per una màquina alhora van establir la base perquè anys després, al 1950, el matemàtic Alan Turing, plantegés per primera vegada la possibilitat de que les màquines poguessin pensar, sembrant així la llavor de la creació d'ordinadors amb intel·ligència artificial. Es crea el test de Turing per determinar si una màquina era o no era intel·ligent.

- Intel·ligència artificial (1956): Martin Minsky y John McCarthy, amb l'ajuda de Claude Shannon y Nathan Rochester organitzen la conferencia de Darthmouth de 1956 on es reconeix la Intel·ligència Artificial com a nou camp.

- Xarxes neuronals (1950 - 1960): Entre 1950 i 1960 diferents científics van començar a investigar com aplicar la biologia de les xarxes neuronals del cervell humà per crear màquines intel·ligents. La idea va derivar creant xarxes neuronals artificials, un model que imita la manera en que les neurones es transmeten la informació a través d'una xarxa de nodes interconnectats. Frank Rosenblatt dissenya el Perceptró, la primera xarxa neuronal artificial d'una sola neurona.

- Primer hivern de la IA (1974 - 1980): Nombroses agències tallen fonts d'investigació pels pocs avenços obtinguts. En aquests anys s'escriu l'algoritme "Nearest Neighbors" que fa néixer el camp de reconeixement de patrons en computadors.

- Explosió dels 80: Als anys 80 neixen els sistemes experts basats en regles generant un nou interès pel Machine Learning. Al 1981 s'introdueix el "Explanation Based Learning" on el computador analitza dades i crea regles generals per tractar dades posteriors. Al 1985 Terry Sejnowski inventa NetTalk, un algoritme que aprèn a pronunciar de la mateixa manera que ho faria un nen.

- Segon hivern de la IA (1987 - 1993): La reputació del camp no es va recuperar fins als anys 2000. Durant aquest hivern el Machine Learning passa de girar envers al coneixement per enfocar-se l'anàlisi de les grans quantitats de dades. Al 1997 l'ordinador Deep Blue de IBM venç al campió mundial de escacs Gary Kaspárov.

- Explosió comercial i actualitat (2006 - actualitat): L'augment de la potencia de càlcul en conjunt amb la gran quantitat de dades impulsen el Deep Learning. Nombroses empreses transformen els seus negocis via Machine Learning aplicat a les dades, processos, productes i serveis per sobrepassar la competència. Fets recents destacables son per exemple:
    - 1989 Yann LeCun proposa l'arquitectura CNN LeNet [24].
    - 2006 Geoffrey Hinton adopta el terme Deep Learning per refeimprir-se a noves arquitectures de de xarxes neuronals profundes.
    - 2012 L'arquitectura CNN AlexNet guanya la competició ImageNet aconseguint sobrepassar en un marge d'un 10% l'error respecte la segona millor arquitectura. [10][11]
    - 2012 Google desenvolupa una xarxa neuronal profunda per detectar patrons en imatges
    - 2014 Facebook desenvolupa DeepFace [22] per reconèixer persones amb precisió humana.
    - 2015 Elon Musk i Sam Altman formen l'organització OpenAI per assegurar el desenvolupament de la Intel·ligència artificial amb impacte positiu en la humanitat.

A dia d'avui estem vivent la tercera explosió de la intel·ligència artificial. Els sectors troben cada cop més aplicabilitat en empreses produint canvis significatius. La gran disponibilitat de dades juntament amb la gran capacitat de còmput actual semblen ser els combustibles d'aquesta tecnologia que promet ser encara més frenètica en els pròxims anys.

4.4.2 Tipologia

Vista l'evolució històrica del Machine Learning vegem-ne els seus tipus.
Els models de Machine Learning es poden agrupar, segons la forma en que aprenen, en tres grans tipus (Sanchez, 2020):

- Aprenentatge no supervisat: Les dades amb les que el sistema aprèn no estan prèviament etiquetades, és a dir, no és distingeixen. El propi sistema s'encarrega de processar i posteriorment categoritzar les dades un cop coneguts els patrons.
  Dins l'aprenentatge no supervisat, destaquen els algoritmes de:
  - D'agrupament o clustering: Tècnica d'anàlisi de dades per organitzar la informació en grups i sense coneixement previ de la seva estructura. Aquest algorisme crea grups de característiques similars, per exemple, tipus de consumidors per realitzar màrqueting efectiu i personalitzat.
  - Reducció dimensional: La reducció dimensional, com el seu nom indica, treballa amb observacions de grans característiques o d'alta dimensionalitat. Aquest tipus d'aprenentatge no supervisat s'encarrega d'establir correlacions entre característiques eliminant informació redundant. S'elimina així el soroll de dades per oferir models més reduïts i de informació totalment rellevant.

- Aprenentatge supervisat: Es refereix al model de Machine Learning en que s'entrena amb un conjunt de exemples amb resultats de sortida coneguts, és a dir, dades prèviament etiquetades. Un cop apresos el patrons i ajustats els paràmetres del model durant l'entrenament, aquest pot predir adequadament davant noves dades no processades anteriorment.
  Dins l'aprenentatge supervisat, destaquen els algoritmes de:
  - Classificació: L'objectiu es predir les classes categòriques (com la pertinença a grups, o la detecció de spam).
  - Regressió: En aquest tipus d'aprenentatge supervisat tenim variables explicatives i una variable de resposta o resultat. La regressió estableix relacions entre aquestes variables oferint un resultat continuo, amb valors reals. La tasca de regressió serà la que efectuarà el nostre model un cop entrenat, proporcionant-nos vectors de valors numèrics característics per cada classe que el model en qüestió ha predit.

- Aprenentatge per reforç: Es una de les branques més importants en que l'objectiu és construir un model amb un agent que millori el seu rendiment basat en la millora obtinguda a cada iteració. Prioritza la millor opció a cada iteració. Un clar exemple es la màquina estratega del joc dels escacs.

4.5 Xarxes neuronals

Un cop hem entès fins on pot arribar i com treballa el Machine Learning per aportar solucions, veiem de forma més precisa el seu component principal, la xarxa neuronal.

Les xarxes neuronals no són més que el tipus de model utilitzat per la resolució dels algoritmes de Machine Learning. Tenen com a idea imitar el funcionament de les xarxes neuronals dels éssers vius. Tot i això, la idea biològica present darrera les xarxes neuronals no ha tingut molta utilitat. Les xarxes neuronals han acabat variant fins a convertir-se en esclaves de les matemàtiques i l'estadística que en determinen la seva arquitectura (capes i interconnexions de neurones) segons el problema a resoldre.

La idea és ben senzilla, donada una quantitat de dades immensa i una xarxa neuronal, les neurones canvien els seus paràmetres o pesos per donar una sortida d'acord a l'esperada per una entrada concreta. Per exemple, coneguts els píxels d'una imatge en escala de grisos conèixer quin nombre està escrit.
El problema a resoldre és clarament l'ajustament d'aquests paràmetres de la xarxa fins obtenir una solució ajustable a la gran majoria o totes les nostres dades d'entrada. Aquest procés és precisament l'anomenat entrenament de la xarxa neuronal. La xarxa un cop entrenada és capaç de predir els resultats esperats donades unes dades d'entrada mai vistes.

Per entendre bé com funciona una xarxa neuronal comencem per la seva unitat mínima, la neurona artificial o perceptró. El perceptró és l'element mínim de la xarxa neuronal composat per varies entrades amb un pes associat cada una. La sortida del perceptró no serà res més que la suma de les entrades multiplicades pels seus respectius pesos.

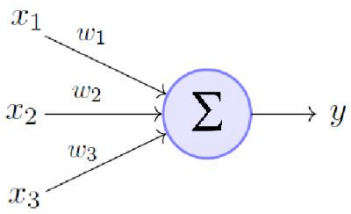

Figura 2. Xarxa neuronal d'una sola neurona (perceptró)

Suposant un problema típic, imaginem que un estudiant vol calcular la nota final d'un examen, per a fer-ho, calcula cada exercici realitzat i el multiplica pel seu pes corresponent per obtenir la nota final. Si escalem el problema, veurem com el conjunt exàmens realitzats i les seves corresponents notes i pesos conformen la nota de l'assignatura. De la mateixa manera el conjunt de notes de l'assignatura conforma la nota final del curs i el conjunt de notes dels cursos la nota final de la carrera. Es fàcil veure la interrelació entre neurones i les diferents capes per obtenir una sortida concreta en aquest exemple.

Suposant ara un possible entrenament de la xarxa, si un professor volgués modificar l'avaluació del curs de forma que la major quantitat d'estudiants aprovessin, podria entrenar la xarxa perquè, conegudes les dades d'entrada (notes dels estudiants) i la sortida esperada (nota superior a 5 per a tots ells) s'ajustin els pesos dels treballs, exàmens i exercicis (pesos de les connexions entre neurones) per tal que tots els estudiants o la gran majoria d'ells aprovin.

4.5.1 Representació matemàtica

Hem dit que la neurona bàsicament realitza la suma ponderada de les entrades i pesos i torna una sortida. Alhora aquesta sortida pot variar en funció de les anomenades funcions d'activació que veurem posteriorment.

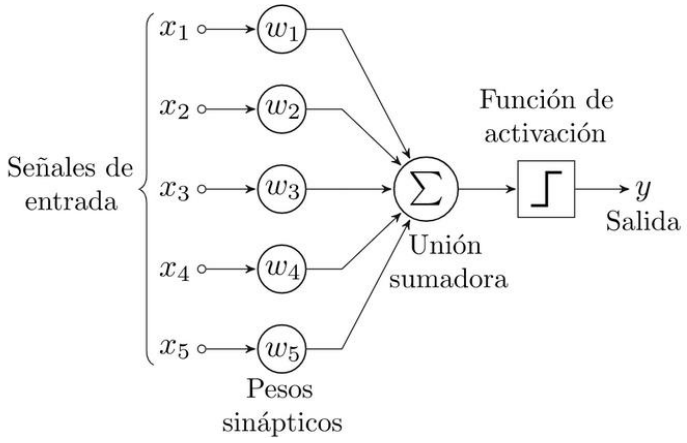

Figura 3. Funció d'activació aplicada a un perceptró

El símil amb les matemàtiques és clar. Els n-valors d'entrada son multiplicats pels seus respectius pesos, és a dir, multipliquem els vectors d'entrada i el vectors de pes donant com a resultat un vector combinació lineal d'entrades i pesos. Aquesta operació entre entrades i pesos és l'anomenada funció de ponderació.

$$X * W^t = (x_1, x_2, \dots x_n) * \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{pmatrix} = \sum_{i=1} x_i * w_i$$

Figura 4. Funció de ponderació

La funció d'activació s'aplica sobre la funció de ponderació aplicant una no-linealitat que permetrà aproximar funcions no lineals. Per últim el resultat es propaga com a sortida cap a altres neurones de la següent capa. Aquest valor a propagar és l'entrada de la següent neurona formant així una xarxa neuronal. La variable de resposta final pren valors continus, com el preu d'un actiu, binaris, si o no, o bé categòrics, una classificació etiquetada per exemple.

Com dèiem, la funció d'activació és l'encarregada de transmetre una sortida fruit de la combinació lineal de pesos i entrades.
Existeixen múltiples funcions d'activació no lineals i no existeix un protocol per decidir si una és millor que l'altre. La tria per utilitzar una funció depèn essencialment del rang de les dades d'entrada i la sortida desitjada.

Algunes de les funcions d'activació més conegudes són [12]:

- Funció llindar o esglaó: Aquesta funció indica que si l'entrada és menor que zero, la neurona donarà un 0 som a sortida però si és major a 0 donarà 1. Aquesta funció és especialment útil quan es tenen sortides categòriques binàries.

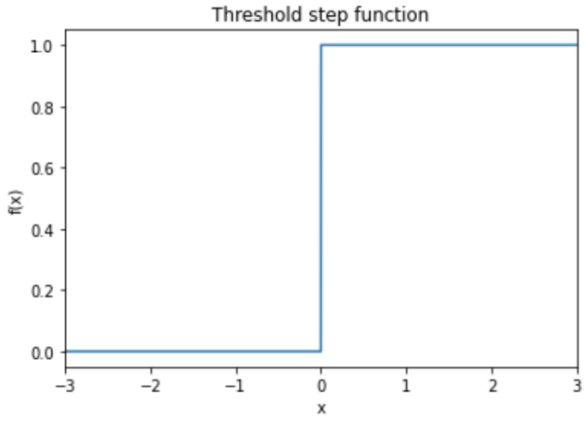

Figura 5. Funció d'activació esglaó

- Funció sigmoidal o logística: Aquesta funció interpreta la sortida com una probabilitat en un rang de valors d'entre 0 i 1. La funció doncs, s'utilitza com a classificadora de dues categories a l'última capa de la xarxa neuronal. Actualment però, no és molt utilitzada ates que no està centrada i això afecta a l'aprenentatge i entrenament de la neurona en el que es coneix com a problema de desaparició de gradient, que tractarem posteriorment.

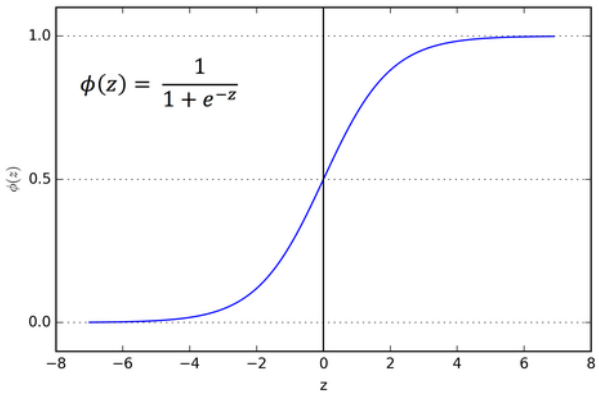

Figura 6. Funció d'activació logística

- Funció tangent-hiperbòlica: Aquesta funció te un rang de valors de sortida d'entre -1 i 1. És tracta d'una funció escalable a la logarítmica amb la qual cosa pateix del problema anomenat anteriorment, la desaparició del gradient.

$$f(x) = \frac{2}{1+e^{-2x}} - 1$$

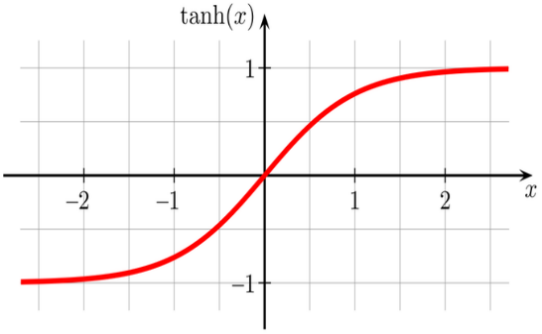

Figura 7. Funció d'activació hiperbòlica

- Funció rectificadora (ReLu): Es tracta de la funció més utilitzada atès que permet que la xarxa aprengui molt ràpidament. Si la funció pren valors negatius dona sempre 0 com a resposta mentre que si pren valors positius la funció és comporta linealment retornant el mateix resultat.

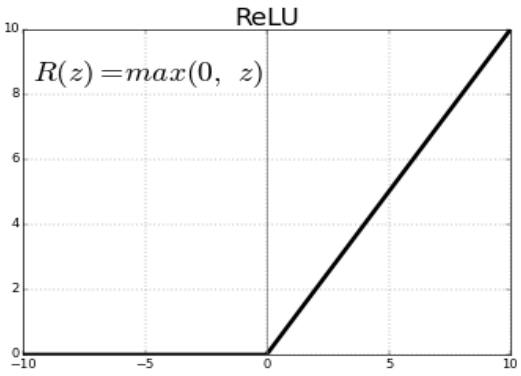

Figura 8. Funció d'activació ReLu

- Funció Softmax: La funció d'activació softmax o funció exponencial normalitzada, és una generalització de la funció logarítmica anterior. S'utilitza per classificar dades múltiples en probabilitats per cada una d'elles, és a dir, obtenim una distribució de probabilitat sobre les diferents possibles sortides.

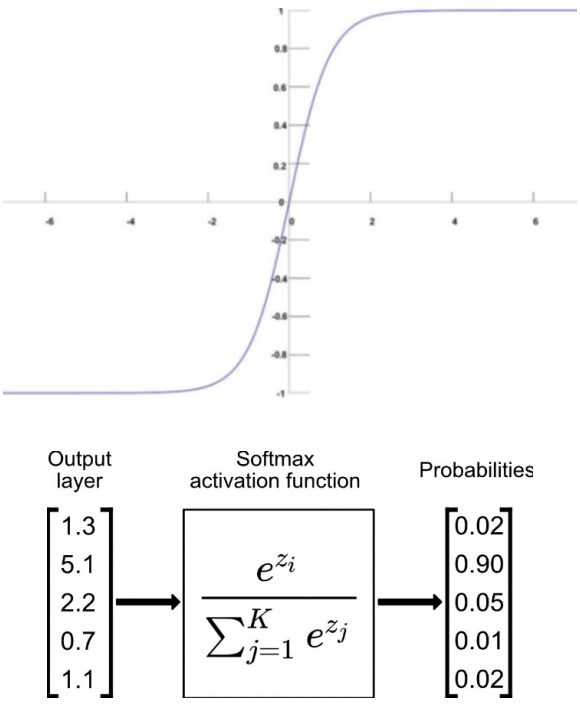

Figura 9. Funció d'activació softmax

4.5.2 Tipologia de xarxes neuronals

Existeixen múltiples xarxes. Totes elles poden ser classificades segons estructura, connexions entre capes, profunditat, activacions, etc.
El tipus de xarxa neuronal que es tria està marcada en gran mesura pel tipus de problema a resoldre. Cal doncs conèixer les xarxes neuronals existents, el seu funcionament i en quin context és millor utilitzar-les [13][14].

- Perceptró: El perceptró o neurona artificial és l'element mínim de qualsevol xarxa neuronal i és alhora una xarxa neuronal. És tracta del model més simple que rep un conjunt d'entrades, aplica una funció d'activació i dona el resultat com a sortida.

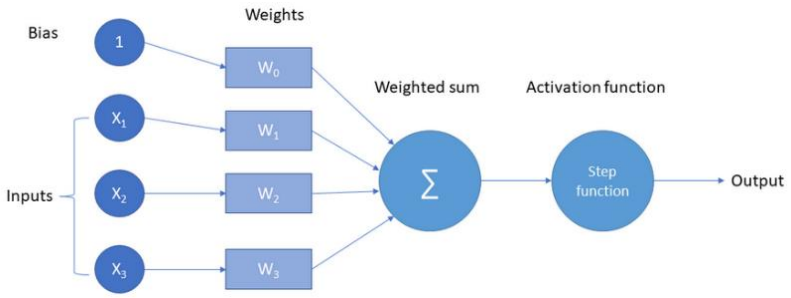

Figura 10. Perceptró

- Xarxes neuronals estàndard: Les xarxes neuronals estàndard són el model més simple després del perceptró. No pretenen reduir la dimensionalitat de les dades d'entrada o realitzar un treball específic en el temps, tan sols obtenir una predicció partir de dades d'entrada.

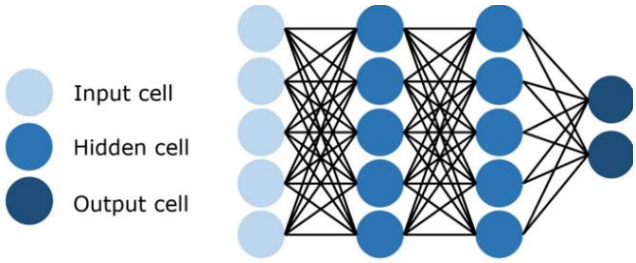

Figura 11. Xarxa neuronal genèrica

- Xarxa neuronal residual: Un dels problemes de les xarxes neuronals és l'esvaïment de gradient o "vanish gradient" en anglès. L'esvaïment de gradient es dona quan les activacions de les neurones son tant petites en valor que és deixen de tenir en compte per la xarxa neuronal. La arquitectura de xarxa neuronal residual permet solucionar aquest problema creant connexions entre capes no consecutives.

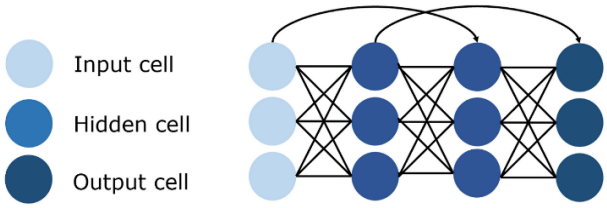

Figura 12. Arquitectura d'una xarxa neuronal residual

- Xarxa neuronal recurrent (RNN)**:** Tipus de xarxa neuronal que conté cicles, d'aquí el seu nom. Aquest tipus de xarxa és útil perquè té en compte més informació prèvia alhora de prendre cada decisió. S'utilitza en tasques seqüencials on les dades d'entrada tenen una component temporal, per exemple, al reconeixement d'àudio o paraules. El principal desavantatge de la xarxa serà la lluita contra l'esvaïment de gradient, que farà molt difícil l'ús de dades de capes posteriors alhora de prendre decisions.

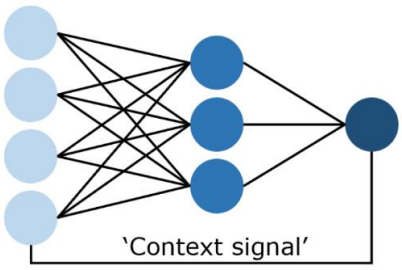

Figura 13. Arquitectura d'una xarxa neuronal recurrent

- Xarxa neuronal de llarg curt termini de memòria (LSTM): Aquest tipus de xarxa neuronal neix per resoldre els problemes d'esvaïment de gradient de les xarxes RNN anteriors. Les long short term memory networks (LSTM) implementen cicles en les seves neurones a mode de memòria. S'utilitzen freqüentment en anàlisi gesticular, de veu o predicció de text.

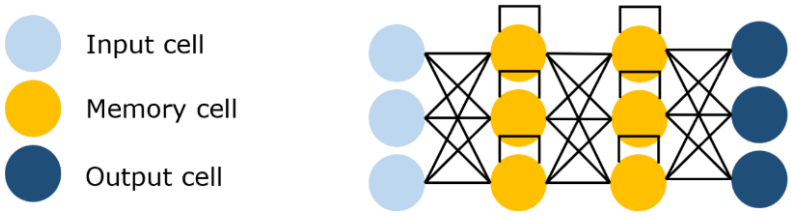

Figura 14. Arquitectura d'una xarxa neuronal LSTM

- Xarxa neuronal convolucional (CNN): Les CNN s'utilitzen principalment per reduir dimensionalitat de dades. És la xarxa neuronal per excel·lència en tractament d'imatge on es necessiten milers de neurones per a formar la xarxa. Per comprimir la informació, la xarxa és basa en operacions de convolució en que s'aconsegueix la compressió de la informació i la reducció de la informació redundant.

  Aquesta és precisament l'arquitectura de xarxa neuronal que utilitza el nostre projecte per rebre una imatge de 224 x 224 píxels i codificar-ne la informació fins obtenir un vector de 128 característiques rellevants úniques i que utilitzarem a posteriori per identificar cada persona present en la imatge d'entrada.

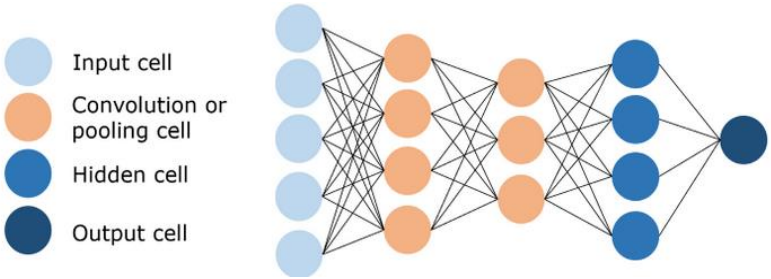

Figura 15. Arquitectura d'una xarxa neuronal convolucional

4.5.3 Descens per gradient

Anteriorment hem vist com les xarxes neuronals es poden expressar en definitiva com una gran funció matemàtica que es va actualitzant ajustant els diferents pesos fins a obtenir la sortida desitjada. En aquest sentit, les xarxes neuronals busquen la configuració de pesos que generalitzi millor la sortida i, per tant, executi les prediccions esperades donades dades mai vistes per la xarxa.
L'error produït alhora de realitzar prediccions és mesura segons l'anomenada funció de cost o loss function en anglès. Tal com passa a les funcions d'activació, existeixen diferents tipus de funcions de cost.

Per a totes les funcions de cost següents, n és el nombre d'exemples del conjunt de dades, $\hat{y}_i$ és el valor estimat i y_i el valor real o predicció que realitza el model per un exemple determinat.

Les funcions de cost més rellevants son [15]:

- Arrel quadrada mitja (RMSE): És calculada com l'arrel quadrada mitjana de la diferència entre el valor previst i el valor real obtingut en la predicció.

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

Figura 16. Arrel quadrada mitja

- Error absolut mitjà (MAE): Es calcula com la suma mitja dels valors absoluts dels errors.

$$MAE = \sum_{i=1}^{n} \frac{|\hat{y}_i - y_i|}{n}$$

Figura 17. Error mitjà absolut

- Error absolut mitjà escalat (MASE): És calcula com l'error absolut mitjà però amb un escalat, com el seu nom indica.

$$MASE = \frac{\sum_{i=1}^{n} |\hat{y}_i - y_i|}{\frac{n}{n-1} \sum_{i=2}^{n} |\hat{y}_n - y_{n-1}|}$$

Figura 18. Error absolut mitjà escalat

- Entropia creuada categòrica (Categorical cross entropy): És una mesura de precisió per a variables categòriques.

$$\mathcal{L}(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m} y_{ij}\log(p_{ij})$$

Figura 19. Entropia creuada categòrica

- Entropia creuada binaria (Binary cross entropy): És una mesura de precisió per a variables binaries.

$$\mathcal{L}(\theta) = -\frac{1}{n}\sum_{i=1}^{n} y_i\log(p_i + (1-y)\log(1-pi))$$

Figura 20. Entropia creuada binaria

L'error és bastant intuïtiu, una mesura sumatòria entre els exemples del conjunt de dades que el model prediu correctament o incorrectament. La forma per la qual el model s'entrena actualitzant els seus pesos mentre redueix la funció de pèrdua s'efectua tal i com fan la majoria d'algoritmes que busquen una solució òptima a un problema, via el descens per gradient.

El descens per gradient és un dels millor mètodes d'optimització numèrica més utilitzats per estimar els coeficients que ens minimitzen la funció de cost. És tracta d'una generalització de la derivada que calcula com el conjunt de totes les derivades parcials de la funció de cost. L'ús de les derivades parcials ens marca la direcció de màxima pendent i per tant en quina direcció hem de anar per trobar el mínim de la funció. L'aprenentatge del model consisteix doncs a trobar iterativament aquells paràmetres que minimitzen la funció de cost. És a dir, que a cada iteració l'algoritme aprèn, via el mètode del descens per gradient, com apropar-se més al mínim de la funció.

La velocitat de convergència fins al punt mínim de la funció de pèrdua es dona a través del paràmetre rati d'aprenentatge, més conegut com learning rate.

El valor que donem al learning rate marca de forma crucial el procés d'aprenentatge del model.

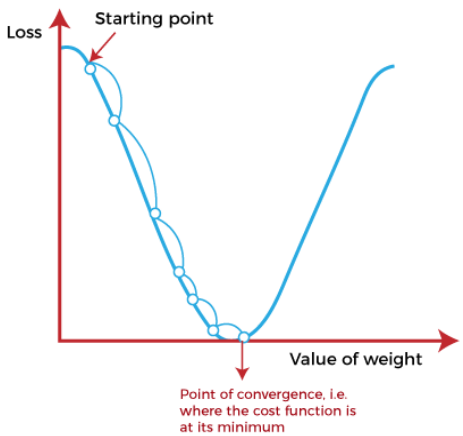

Figura 21. Descens per gradient utilitzant un learning rate de major a menor valor

Un learning rate molt gran fa que convergim la funció de pèrdua de forma ràpida però amb el problema de la possible sobrepassada del punt mínim de la funció. Per altra banda, un learning rate molt petit farà que convergim en petits passos i que per tant l'entrenament és torni molt lent.

És fàcil deduir que durant la fase d'aprenentatge del model, seria ideal que cada cert nombre d'iteracions reduïm el valor del learning rate. Aquesta és la funcionalitat que ens aporta l'anomenat learning rate decay fent que el valor del learning rate es vegi reduït cada cert nombre d'iteracions donat proporcionant-nos així un descens ràpid per la funció de cost a les primeres iteracions de la fase d'aprenentatge i un descens més petit quan ens acostem al mínim de la funció i per tant no el sobrepassem [16].

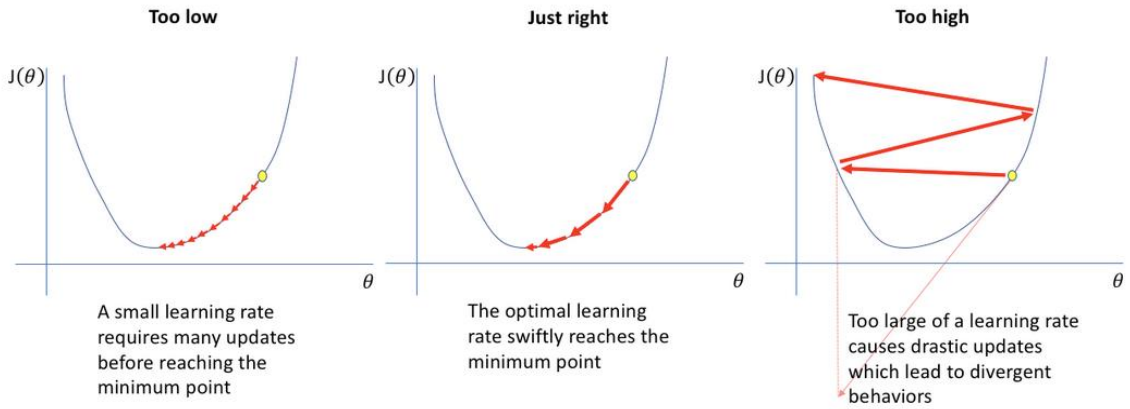

Figura 22. Convergència a la funció de pèrdua segons valor del learning rate

Per conèixer com de bé hem establert els híper-paràmetres d'entrenament del model anteriors, learning rate i learning rate decay, hem de fixar-nos en com evoluciona la funció de pèrdua durant la fase d'entrenament.

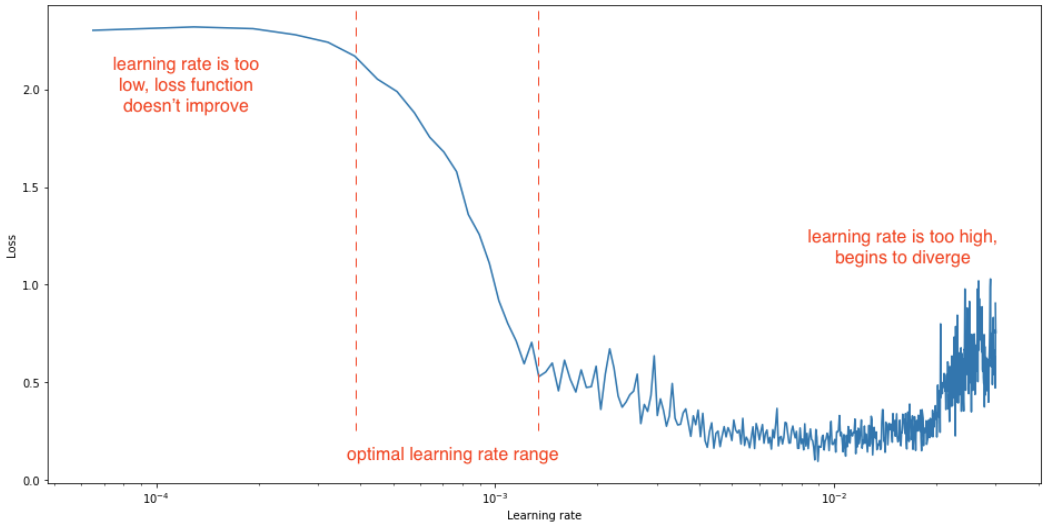

Figura 23. Convergència o divergència segons l'evolució de la funció de cost

A la figura anterior és mostra si la funció de pèrdua no convergeix, convergeix o divergeix entre valors fent referència a l'ús d'un learning rate massa petit, òptim o massa gran respectivament.

Fins ara hem vist una representació de la funció de pèrdua en dues dimensions, tanmateix aquesta no és la realitat, la nostra funció de pèrdua tindrà centenars i fins i tot milers de paràmetres i per tant la seva representació no és senzilla ni tampoc ho son els problemes que això comporta durant la fase d'entrenament.

Imaginant la possible representació de la nostre funció de pèrdua en un espai tridimensional és fàcil veure com durant la recerca del mínim global a la fase d'entrenament podem caure en mínims locals dins la nostre funció de pèrdua, en punts d'inflexió (saddle points) o bé, en el millor dels casos al mínim global de la funció de pèrdua.
Els tres possibles resultats anteriors és poden observar fàcilment a la següent figura:

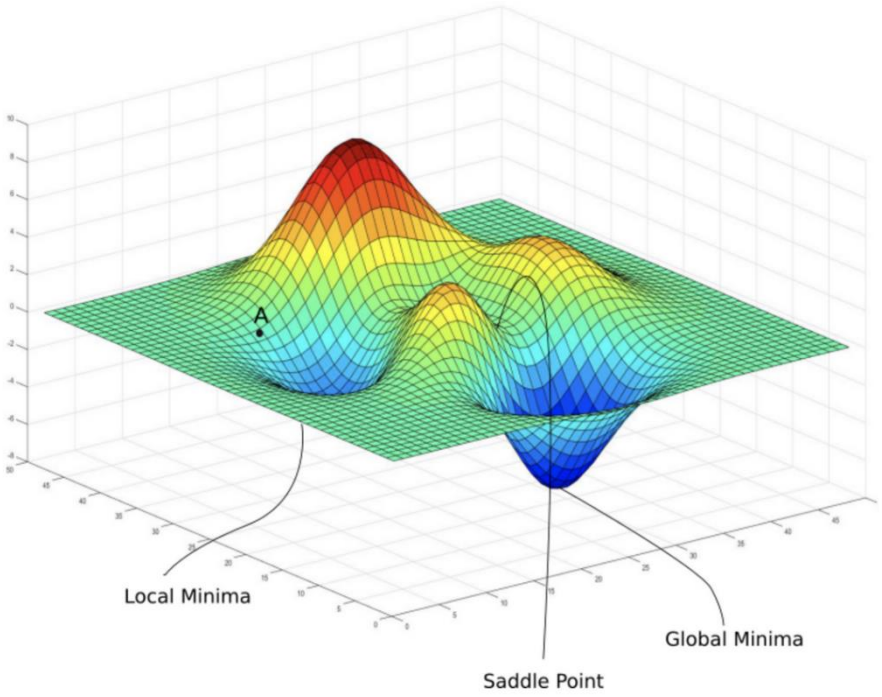

Figura 24. Mínim local, global i punt de cadira en una funció de cost tridimensional

La realitat és que no podem evitar totalment quedar-nos atrapats als mínims locals o en punts d'inflexió de la funció de pèrdua durant l'entrenament. Tanmateix existeixen tres variants de l'algoritme de gradient descent que ens poden ajudar a mitigar-ne les possibilitats.

Les variants de l'algoritme de descens de gradient son [17]:

- Descens de gradient per lots: Conegut com batch gradient descent, aquesta és la variant que hem vist fins ara i que utilitza totes les dades disponibles d'un sol cop comportant als problemes d'estancament vistos. Això passa perquè el gradient és calcula sempre utilitzant totes les mostres i per tant arriba un punt on les variacions són mínimes. Alhora el cost computacional és veu afectat. Utilitzar tot el set de dades per al càlcul del descens de gradient en cada iteració és extremadament costos. Com a norma general, sempre busquem l'aleatorietat a la xarxa neuronal per tal que aquesta generalitzi millor entrades noves dades.

- Descens de gradient estocàstic: Conegut com a stochastic gradient descent, introdueix una sola mostra aleatòria a cada iteració. El gradient és calcula per aquella mostra concreta introduint aleatorietat al model i dificultant l'estancament de l'algoritme. El problema és que necessita més iteracions per arribar al mínim global desitjat precisament per utilitzar aquesta aleatorietat. L'aventatge és que obté un nou càlcul a cada iteració i això fa que avanci més ràpidament.

- Descens de gradient estocàstic en mini-lots: Conegut com a mini-batch stochastic gradient descent, utilitza n mostres aleatòries a cada iteració conservant els aventatges de la primera i la segona versió explicades i aconseguint així un entrenament ràpid i efectiu.
  Aquest mètode és precisament l'utilitzat al projecte pel fet d'obtenir un bon balanç entre aleatorietat i temps d'entrenament. El nostre model, utilitzant aquesta variant, dividirà el set de dades d'entrenament en conjunts de mostres aleatòries i utilitzarà cada un d'ells per intentar minimitzar la funció de cost a cada iteració.

A la següent figura és pot veure el funcionament dels tipus de variants de l'algoritme gradient descent explicats anteriorment:

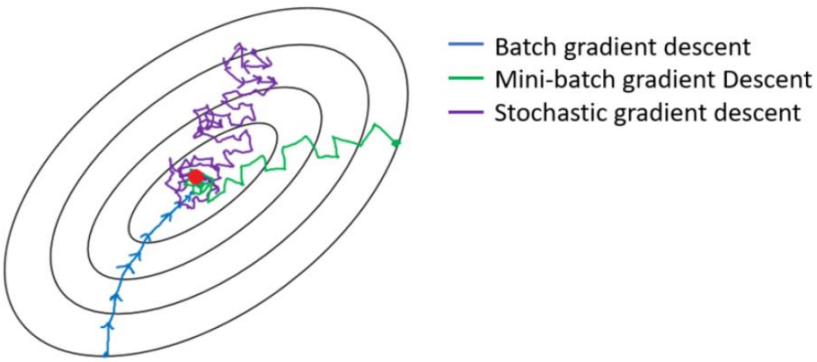

Figura 25. Trajectòries de l'algoritme gradient descent segons la variant

4.5.4 Sobre ajustament i sub ajustament

Un cop seleccionada la variant del descens de gradient i els optimitzadors adequats a utilitzar comencem a entrenar el model de xarxa neuronal. El procés d'entrenament passa per calcular totes les activacions de neurones de cada capa donada una entrada (forward propagation). Seguidament i amb les activacions anteriors guardades, és computa l'error per passar a actualitzar els pesos de les activacions guardades de forma que aquest error és minimitzi i per tant ens acostem al mínim global de la funció de pèrdua.

Arribats a aquest punt ja coneixem com s'entrena el model de xarxa neuronal profunda i quins problemes pot presentar durant l'entrenament, tanmateix això no vol dir que funcioni de manera òptima. El fet d'obtenir un model entrenat no vol dir que aquest ens realitzi sempre les prediccions correctes. Podem presentar problemes de sobre ajustament i sub ajustament, vegem-los [18]:

- Sobre ajustament (overfitting): El sobre ajustament és produeix quan el model s'ha sobre ajustat a les dades d'entrenament fent que no és comporti com s'espera davant noves dades mai vistes.
- Sub ajustament (underfitting): El sub ajustament per altra banda és produeix en cas contrari, quan el model no s'ha entrenat correctament amb les dades d'entrenament i no és capaç de predir correctament les noves dades mai vistes.

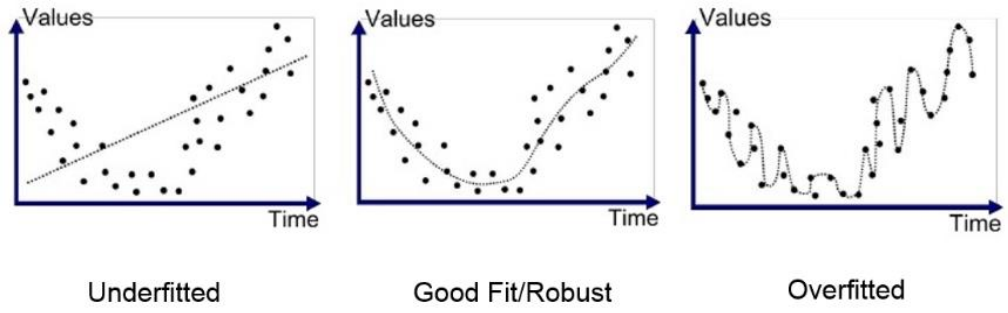

Figura 26. Ajustament, sobre ajustament i sub ajustament

Aquest problema és pot identificar de forma fàcil gràcies als diferents conjunts de dades fonamentalment utilitzats. Per entrenar un model es divideix tot el set de dades en tres sub conjunts totalment diferents.

- Conjunt d'entrenament: És el conjunt utilitzat per entrenar el model.
- Conjunt de validació: És el conjunt utilitzat per testejar el model durant l'entrenament.
- Conjunt de testatge: És el conjunt utilitzat per testejar el model ja entrenat.

Via els tres subconjunts anteriors som capaços d'identificar els problemes de sobre ajustament o sub ajustament durant el mateix entrenament.
Durant la fase d'entrenament, el nostre model rep dos subconjunts, el d'entrenament i el de validació. Quan ambdós subconjunts disminueixen la funció de pèrdua estem en el cas de sub ajustament, és a dir, el nostre model continua aprenent, encara no ha arribat al mínim local. Quan ambdós subconjunts comencen a estancar-se en la seva funció de pèrdua corresponent, vol dir que estem arribant al model òptim. En aquest punt el model ja no estem aprenent gaire de les dades d'entrenament i l'error de validació també deixa veure que el model ha après i esta realitzant bones prediccions.
El moment crític i on hem d'aturar l'entrenament és quan la funció de pèrdua pel conjunt d'entrenament s'estanca o segueix disminuint poc a poc i en canvi la funció de pèrdua per al subconjunt de validació comença a augmentar. En aquest últim cas, on l'error de validació augmenta, estarem davant la situació de overfitting i és on el nostre model s'haurà sobre ajustat a les dades d'entrenament, fent que pel conjunt de validació les prediccions no siguin correctes. Estarem davant un model que deixa de generalitzar correctament.

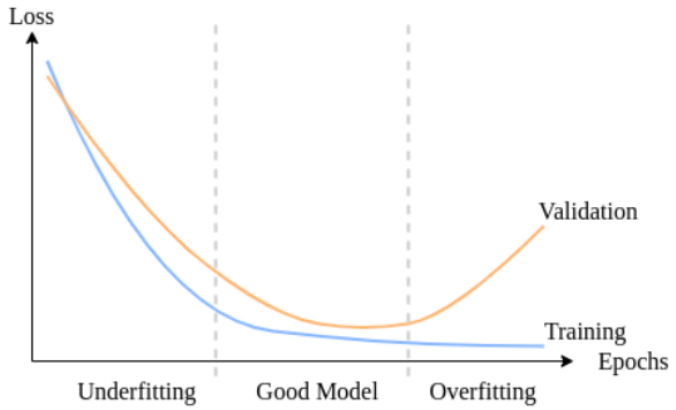

Figura 27. Ajustament, sobre ajustament i sub ajustament analitzant les funcions de pèrdua del conjunt d'entrenament i validació

És clau doncs entendre que hem de quedar-nos amb el millor model si volem generar bones prediccions. Serà només amb aquest model on obtindrem un bon ajustament al nostre set de dades sempre a l'espera de provar el model final entrenat a l'últim subconjunt de testatge.

Una altra manera de resoldre els problemes sobre ajustament i sub ajustament és mirar les mètriques de la precisió del model.

La precisió o acuracy no és res mes que la predicció que realitza el model sobre un conjunt de dades donat.
La precisió és calcula com:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} = \frac{TP + TN}{TP + TN + FP + FN}$$

Figura 28. Càlcul de la precisió classificatòria d'un model

on:

- Positiu / negatiu: Dos possibles valors a classificar
- TP / TN – True positive / negative: Valor actual i predicció donada son iguals
- FP / FN – False positive / negative: Valor actual i predicció donada son diferents

Calculant la precisió per els subconjunts d'entrenament i validació podrem avaluar l'entrenament tal com fèiem amb la funció de pèrdua.
Si el nostre model millora successivament la precisió per ambdós conjunts és que el model encara esta millorant, aprenent durant l'entrenament, i per tant estem en un cas de underfitting.
Si el nostre model comença a estancar-se en precisió és que esta deixant d'aprendre i poc a poc s'acosta a la solució òptima on per ambdós conjunts realitza prediccions correctes.
Si el nostre model segueix millorant la precisió per al subconjunt d'entrenament però no per el subconjunt de validació, estem en el cas d'overfitting en que realitzem correctes prediccions pel conjunt de d'entrenament però deixem de predir correctament pel conjunt de dades de validació.

Com abans, trobarem el millor model just abans de començar a entrar en overfitting, el model que millor generalitza ambdós subconjunts.
És precisament via precisió com és comporta el nostre model, guardant-se sempre els pesos del model que generalitza millor. Cal ser conscients però que el nostre projecte treballa amb un set de dades molt petit i per tant aquestes mètriques gairebé no son identificatives de l'evolució del nostre model sinó tan sols una referència.

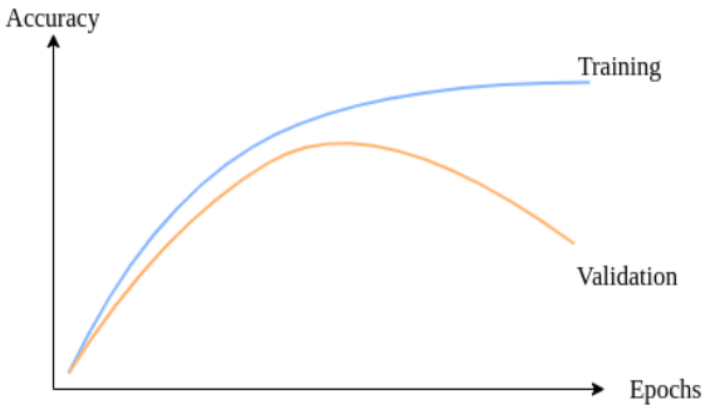

Figura 29. Ajustament, sobre ajustament i sub ajustament analitzant l'accuracy dels conjunts d'entrenament i validació

4.5.5 Optimitzadors

Altres formes de mitigar les probabilitats d'efectuar un mal entrenament ja sigui per caure en un mínim local, saddle point, per no haver arribat al mínim global o per tenir de problemes d'overfitting o underfittng, son els optimitzadors.
Existeixen múltiples optimitzadors que podem triar alhora d'entrenar el nostre model [19].

- Regularització: La regularització ajuda a reduir els problemes d'overfitting pel fet de penalitzar els valors elevats. Uns valors elevats és tradueixen directament en overfitting, atès que durant el procés d'aprenentatge aquests prenen molta més importància pel model. Si el que volem és que el model generalitzi millor, cal penalitzar els pesos elevats per tenir tots en compte totes les dades disponibles a l'entrenament. Tanmateix, determinar un coeficient de regularització molt elevat ens pot dur a problemes d'underfitting. Cal optimitzar el valor de regularització triat per obtenir un model que s'ajusti correctament a les dades, un model que generalitzi bé.
  La regularització L1 (Lasso) i L2 (Ridge) són els tipus de regularització més comuns. Aquests actuen afegint un terme a la funció de pèrdua conegut com a coeficient de regularització.
  - Regularització L1 (Lasso): Penalitza els valors absoluts dels pesos a valors que poden arribar a ser zero.

$$Loss = Error(y, \hat{y}) + \lambda \sum_{i=1}^{N} |w_i|$$

Figura 30. Fórmula regularització L1

  - Regularització L2 (Ridge): Força a que els pesos tendeixin a zero però mai sense arribar a ser zero. És coneix aquest tipus de regularització com a weight decay i és precisament la que la nostra xarxa neuronal utilitza, evitant el problema del valor dels pesos igual a 0 present a la regularització L1 anterior.

$$Loss = Error(y, \hat{y}) + \lambda \sum_{i=1}^{N} w_i^2$$

Figura 31. Fórmula regularització L2

- Drop-out: És tracta d'una tècnica que consisteix en desactivar aleatòriament neurones de la xarxa. D'aquesta manera, per cada iteració d'entrenament, la xarxa neuronal proveirà un set diferent de prediccions amb la conseqüència d'obtenir com a resultat final una xarxa neuronal que generalitza millor.

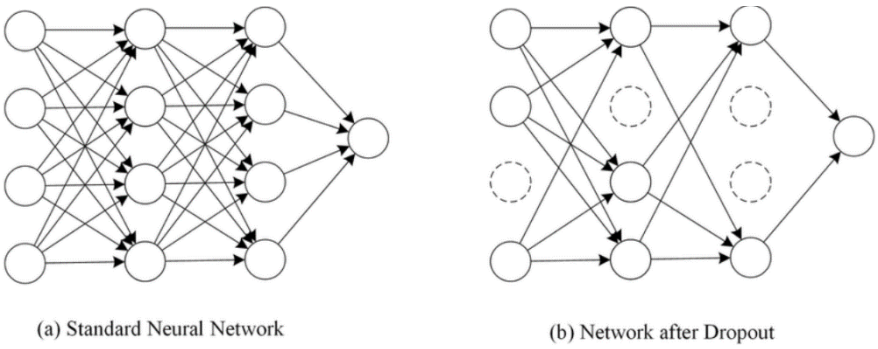

Figura 32. Desactivació de neurones aleatòries (Drop-out)

- Data augmentation: El mètode d'augment de dades és basa en incrementar el set de dades d'entrenament. D'aquesta forma tant simple el model és capaç de rebre noves dades aconseguint generalitzar millor evitant l'overfitting de les dades originals.
  Considerant un set petit de dades, data augmentation és una tècnica que permet aconseguir noves dades aplicant transformacions a les dades originals. Per aquest motiu, aquest projecte treballa amb data augmentation, pel fet de partir d'un set de dades d'entrenament de molt poques imatges per persona.
  Concretament el nostre entrenament utilitzarà data augmentation en forma de canvis d'il·luminació, saturació i contrast de les imatges de d'entrenament.

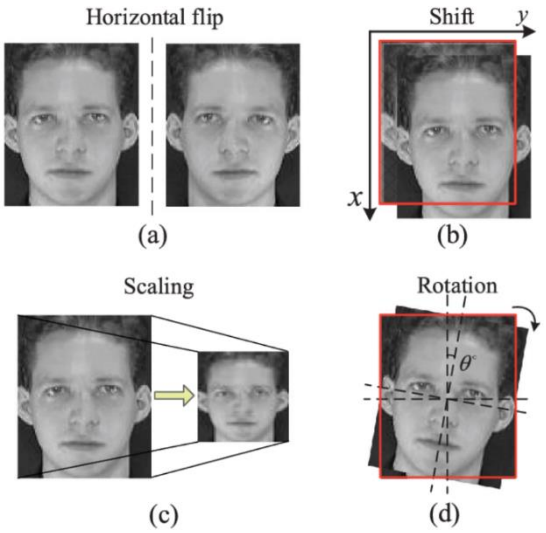

Figura 33. Tècniques d'augment de dades

- Early stopping: Es una estratègia d'entrenament que hem vist anteriorment. El mètode consisteix en parar l'entrenament just abans que la funció de pèrdua al set de validació comenci a augmentar de forma unilateral a la funció de pèrdua del set d'entrenament. El mètode és centra en obtenir el millor model i evitar l'overfitting. Aquest però no és exactament un optimitzador que s'apliqui al nostre projecte, ja que el nostre projecte és guarda aquest millor model just abans que la funció de pèrdua del set de validació augmenti i continua l'entrenament fins al final, no para. D'aquesta forma és possible obtenir un millor model a posteriori. Tanmateix cal recordar que el nostre set de dades és massa petit com per utilitzar aquesta tècnica, amb la qual cosa el que fem és guardar el millor model obtingut, el de millor accuracy i prosseguir amb l'entrenament per quedar-nos finalment amb el millor model, l'últim guardat de l'entrenament o bé el de més accuracy registrat.

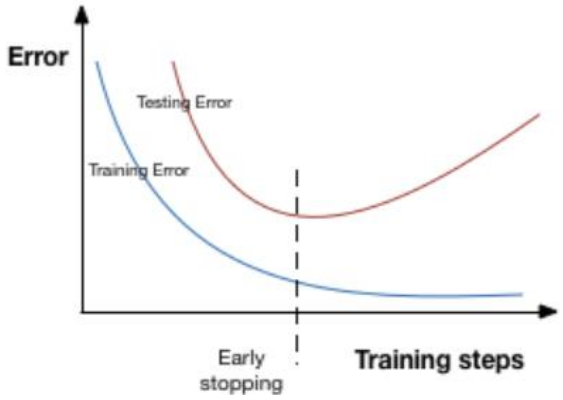

Figura 34. Obtenció del model òptim segons la tècnica Early stopping

Fins ara hem vist mètodes d'optimització que busquen la reducció de l'overfitting, tanmateix, existeixen altres mètodes optimitzadors que, en paral·lel a aquests, busquen la convergència de la funció de pèrdua cap al mínim global de la manera més ràpida possible [20].

- Adagard: Adaptative gradient descent permet canviar el learning rate de cada paràmetre cada cert temps. És el mètode d'entrenament més lent atès que el learning rate només decau en valor. AdaDeleta n'és la seva variant que elimina el problema del valor del learning rate. Ho fa limitant el valor que pot tenir el propi paràmetre. Com a contrapartida, el learning rate no decau prou i l'entrenament pot no tenir final.

- Momentum: Com el seu nom indica, momentum tracta de accelerar la convergència cap a la direcció més rellevant. Redueix l'oscil·lació en mètodes amb gran variància com al descens de gradient estocàstic vist anteriorment. Com a contrapartida, introdueix un nou híper-paràmetre que caldrà definir en valor.

- Rms-Prop: Root mean square propagation és una versió combinada de Adagard i Momentum. Permet la variància del learning rate per cada paràmetre i la ràpida convergència de la funció de pèrdua alhora sumant els problemes d'ambdós optimitzadors, la decadència de valor del learning rate i el possible salt dels mínims globals de la funció.

- Adam: Adaptative moment estimation utilitza els principi dels optimitzadors RMS-Prop, AdaDelta i Momentum per convergir ràpidament. Permet adaptar el learning rate de cada paràmetre cautelosament i sense accelerar la convergència massa per no passar per sobre un possible mínim global. En definitiva convergeix ràpida i cautelosament rectificant alhora la possible variància present al descens de gradient estocàstic. El seu baix computacional i l'adaptació dels aventatges dels optimitzadors anteriors fa que sigui molt atractiu. Aquest és el motiu pel qual l'optimitzador Adam és l'utilitzat en el nostre projecte.

L'ús dels optimitzadors pot estalviar hores o fins i tot dies d'entrenament. No existeix una norma directa per conèixer quin és el millor optimizador a utilitzar sinó que hem d'anar entrenant, provant i coneixent el model per treballar en la tria dels optimitzadors correctes un cop coneguts els conceptes teòrics, avantatges i desavantatges de cada un d'ells.
Tal i com podem veure a la següent figura, tot i que Adam és per molts el millor optimitzador per tenir avantatges d'altres optimitzadors, és també lent i altres optimitzadors podrien ser millors per aquesta funció de pèrdua en concret.

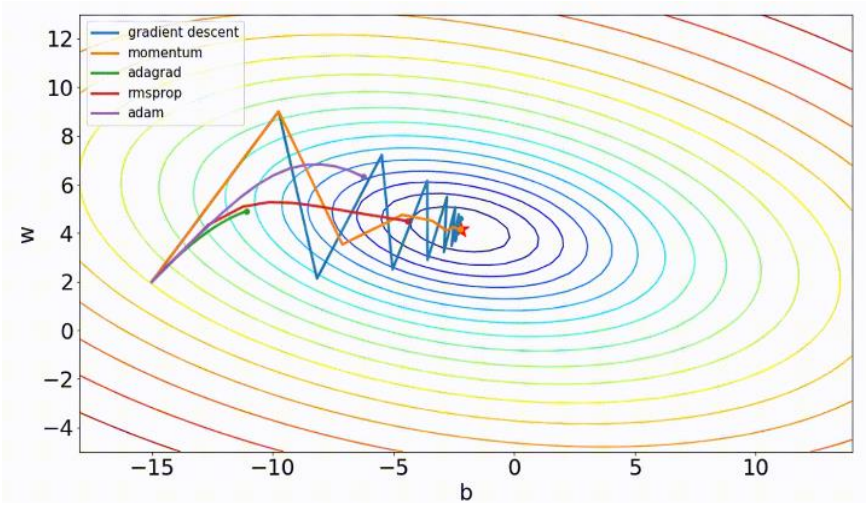

Figura 35. Trajectòria al mínim global de la funció de cost segons l'ús de diferents optimitzadors

CAPÍTOL 5

Resultats

A continuació és detalla el procés seguit per el desenvolupament del projecte. En primer lloc és descriuen les especificacions tècniques i eines usades. En segon lloc s'exposa tota la informació necessària per entendre el funcionament i implementació del model FaceNet utilitzat [6]. Finalment és mostren els resultats obtinguts durant el projecte així com un anàlisi dels mateixos.

5.1 Especificacions tècniques

A continuació és detallen les especificacions tècniques a nivell de software i de hardware usades per a realitzar el projecte.
Software utilitzat:

Software	Descripció
PyCharm Community Edition	Entorn de desenvolupament integrat (IDE) per a la programció utilitzant Python
SublimeText	Editor de text i de codi font
Tensorflow	Biblioteca de codi obert desenvolupada per Google per a Machine Learning
PyTorch	Biblioteca de codi obert per a Machine Learning basada en Torch
Conda	Eina de gestió de paquets i d'entorns virtuals

Hardware utilitzat:

Processador	12th Gen Intel Core i9-12900K - 16 cores- 24 threads
Memòria RAM	Corsair Vengeance 5200MHz DDR5 64Gb(2x32Gb) x 4
Targeta Gràfica	Nvidia GeForce RTX 3070
Placa base	Asus Prime Z690-A
Memòria	Corsair MP600 1TB x4
Sistema operatiu	Ubuntu Desktop 20.04.4 LTS
Webcam	Logitech Brio Webcam 4k Ultra HD

5.2 Entorn de treball

Per a l'elaboració del projecte s'han desenvolupat diversos programes utilitzant el llenguatge Python en conjunt amb el gestor d'entorns virtuals Conda. Per a guardar, entrenar, carregar i utilitzar el model d'identificació facial entrenat en Machine Learning s'utilitza PyTorch, una biblioteca de codi obert basada en Python i centrada en facilitar eines per l'ús de l'aprenentatge automàtic.

Durant el projecte s'utilitza un model prèviament entrenat via el conjunt de dades VGGv2 de 3.31 milions d'imatges i 9131 subjectes [21] i testejat amb el conjunt de dades LFW de 13.233 imatges i 5749 subjectes aconseguint un 90% de precisió. Es parteix doncs d'un model extractor de característiques facials prou bones però llunyanes a l'extracció de característiques esperada per poder dur a terme el reconeixement facial amb les pròpies dades.

Totes les imatges utilitzades per a entrenar i classificar a través del model han estat extretes de companys de feina, amics i familiars amb el degut consentiment previ. Per tant, les imatges utilitzades són extretes d'individus reals.

Un cop finalitzat el procés d'entrenament del projecte, s'utilitzen alhora noves imatges de nous usuaris no incorporats al sistema que col·laboren a realitzar proves més extenses i generalitzades de reconeixement de usuaris no identificats.

5.3 Model FaceNet

Un cop vista tota la teoria referent a les xarxes neuronals, a continuació és detalla la seva utilització al nostre projecte.

L'objectiu principal d'aquesta fase pràctica del projecte consisteix en processar totes les dades d'entrada (imatges de cares de diferents persones) mitjançant una xarxa neuronal convolucional (CNN) anomenada FaceNet [23]. Via aquest tipus d'arquitectura de xarxa neuronal vista a l'apartat teòric anterior, es rep una imatge d'entrada i en generen les característiques més rellevants en un vector anomenat d'ara en endavant embedding.

Per a poder processar les dades i extreure característiques cal definir un model i entrenar-lo amb una gran quantitat de dades. Com ja sabem el projecte és centra en aconseguir el procés d'identificació de persones precisament amb un conjunt d'imatges molt reduït i per aquest motiu s'utilitza un model molt popular utilitzat en la majoria de problemes de reconeixement facial amb intel·ligència artificial, prèviament entrenat i de fàcil integració.

El model FaceNet [6] proposa una solució al problema de la verificació i el reconeixement de cares descomposant imatges d'entrada en un embedding via una arquitectura de xarxa neuronal convolucional (CNN) que es detalla posteriorment.

FaceNet [6] és publica l'any 2015 per un equip de Google com una solució al problema de la verificació, la identificació i el clustering de cares.
És presenta amb dues arquitectures alternatives:

- Model NN1: Basat en l'arquitectura Zelier & Fergus
- Models NN2-NN3-NN4/NNS1-NNS2: Basats en l'arquitectura Inception

La principal diferència entre arquitectures és en primer lloc la mida de la imatge d'entrada i en segon lloc el nombre de paràmetres configurables. Cal destacar en aquest segon punt que l'arquitectura Inception NN2 va aconseguir millorar la precisió del model NN1 amb un nombre d'operacions molt menor fruit de reduir el nombre de paràmetres configurables a una vintena dels configurables a l'arquitectura NN1.

architecture	VAL
NN1 (Zeiler&Fergus 220×220)	$87.9\% \pm 1.9$
NN2 (Inception 224×224)	$89.4\% \pm 1.6$
NN3 (Inception 160×160)	$88.3\% \pm 1.7$
NN4 (Inception 96×96)	$82.0\% \pm 2.3$
NNS1 (mini Inception 165×165)	$82.4\% \pm 2.4$
NNS2 (tiny Inception 140×116)	$51.9\% \pm 2.9$

Figura 36. Arquitectura i precisió del model FaceNet sobre les dades de validació del conjunt LFW

Basant-nos en els resultats anteriors, aquest projecte se centra en utilitzar l'arquitectura Inception NN2 que rep imatges d'entrada de 224 x 224 x 3 píxels.

5.3.1 Arquitectura Inception NN2

El model Inception NN2 rep una imatge d'entrada de 224 x 224 x 3 píxels i realitza operacions de convolució, max pooling i average pooling per acabar obtenir un vector de característiques representatiu de dimensionalitat 1 x 1 x 128. Alhora es realitzen operacions de regularització que ja hem vist anteriorment.

Veiem amb detall aquestes operacions:

- Convolució: L'operació de convolució utilitza una matriu anomenada kernel que passa per cada píxel de la imatge i canvia el seu valor segons si és compleix o no la condició del kernel. Per reduir la dimensionalitat de la imatge al kernel se li pot passar o no un paràmetre stride o salt, de manera que el kernel no s'apliqui a cada píxel.
  En aquesta arquitectura, s'apliquen strides de valor 1 i 2 que mantenen o redueixen a la meitat la mida de la imatge d'entrada. Utilitzant un stride de valor 1 el que farem és passar el kernel per cada píxel de la imatge i en conseqüència no reduir la mida de la imatge. Si per altra banda un stride igual a 2 és aplicat, estarem passant el kernel cada dos píxels amb la conseqüència de reduir a la meitat la dimensionalitat de la imatge.
  Cal notar que als kernels usats s'aplica padding, és a dir, s'utilitzen valors igual a 0 per acabar d'omplir els píxels sense valor que tindrà el kernel quan comenci per les primeres i ultimes files i columnes de la imatge. Si no fos així, tot i utilitzar un stride de 1, estaríem perdent píxels al aplicar el kernel i reduiríem la mida de la imatge conforme s'incrementa la profunditat de la xarxa.

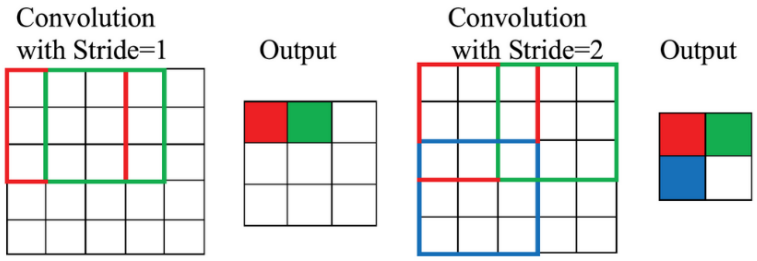

Figura 37. Il·lustració d'una operació de convolució amb stride o salt

- Max pool i average pool: Són operacions que seleccionen l'element màxim o la mitja dels elements, segons el tipus d'operació. Aquestes operacions serveixen per reduir la dimensionalitat de les dades guardant només aquella informació important, de més valor o de valor mitjà. Com passava anteriorment, és freqüent usar un valor strides, en aquest cas per no passar el filtre per valors vistos pels filtres anteriors.

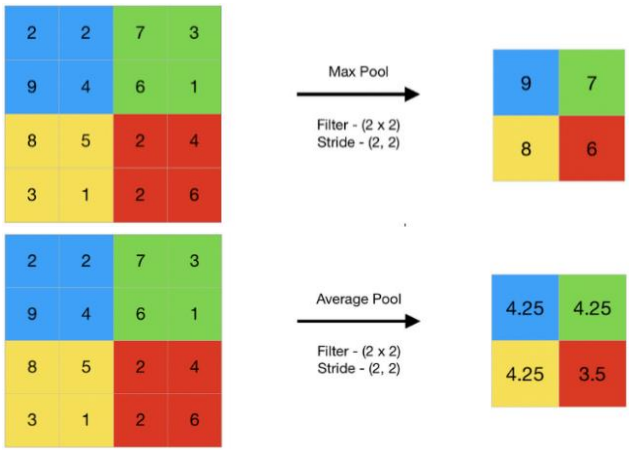

Figura 38. Operacions max pooling i average pooling

- Capes inception: Les capes inception tenen com idea clau executar múltiples filtres de convolució i operacions de pooling simultàniament. En aquesta arquitectura NN2 i respecte l'arquitectura NN1, són usades pel fet de reduir el nombre de paràmetres totals de la xarxa alhora que redueixen la potència necessària sense perdre la precisió final del model obtingut. La intenció de les capes inception és doncs deixar que la xarxa neuronal ajusti els paràmetres adequadament per afavorir a la reducció de la dimensionalitat de la imatge d'entrada i afavorir alhora l'ajustament de paràmetres per extreure les millors característiques descriptores de la imatge. Per resoldre l'augment en el nombre de paràmetres a configurar, múltiples versions de capes inception han estat desenvolupades fins a dia d'avui.

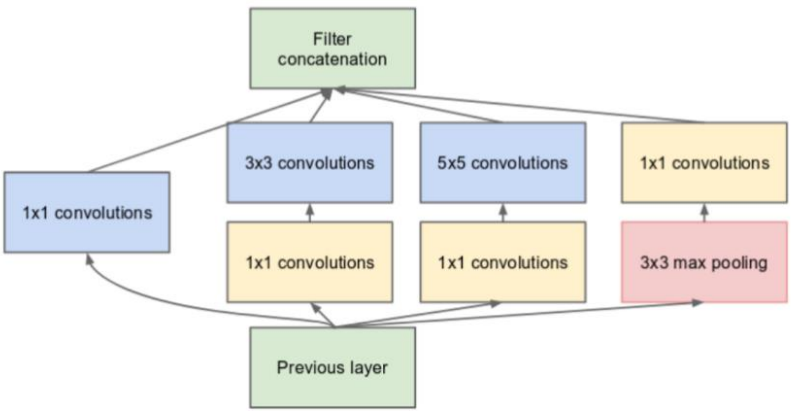

Figura 39. Estructura capa inception V1

Vistes les operacions anteriors ara ja podem entendre com funciona l'arquitectura Inception NN2 usada. A la següent taula es mostren les operacions que executa la xarxa. Podem veure les convolucions aplicades amb la mida del kernel i el valor de stride o salt entre píxels on aplicar el kernel. També podem veure les operacions max-pooling i average-pooling amb la seva mida de kernel i valor de stride. En tercer lloc podem veure les normalitzacions aplicades (L2 norm), ja explicades prèviament al marc teòric del projecte. Per últim, podem veure també les operacions de les capes inception amb el nombre convolucions aplicades a cada kernel segons mida 1x1, 3x3 o 5x5.

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 40. Arquitectura FaceNet Inception NN2

5.3.2 Funció de pèrdua Triplet Loss

FaceNet [6] va suposar un canvi de paradigma també pel que fa a l'entrenament dels models de xarxa neuronal CNN anteriors pel fet de desenvolupament d'una nova funció de pèrdua, la triplet loss.

L'objectiu de la funció de pèrdua triplet loss és entrenar via la construcció de triplets. Els triplets son conjunts de tres imatges de dues persones diferents formades per una imatge anomenada anchor, una imatge de la mateixa classe que la imatge anchor anomenada positive i una imatge de diferent classe anomenada negative.
La funció de pèrdua triplet loss, via la construcció de triplets, entrena la xarxa de tal forma que entre el parell d'imatges anchor - positive tinguem molta similitud i entre el parell anchor - negative tinguem poca o cap similitud.
El que aconseguim és que les representacions vectorials de cada cara s'apropin cada cop més en l'espai de característiques per a les imatges de la mateixa classe i és distanciïn per a imatges de diferent classe. Per tant aquesta funció de pèrdua tan sols és basa en buscar la distanciació dels parells anchor - positive respecte el parell anchor - negative.

Formalitzant la funció de pèrdua triplet loss tenim que:

$$\mathcal{L} = max(d(a,p) - d(a,n) + margin, 0)$$

Figura 41. Funció de pèrdua triplet loss

On:

a: Imatge anchor
p: Imatge positive
n: Imatge negative
d(a,p): Distància entre els embeddings de anchor i positive
d(a,n): Distància entre els embeddings de anchor i negative
margin: Marge mínim entre els parells positiu i negatiu

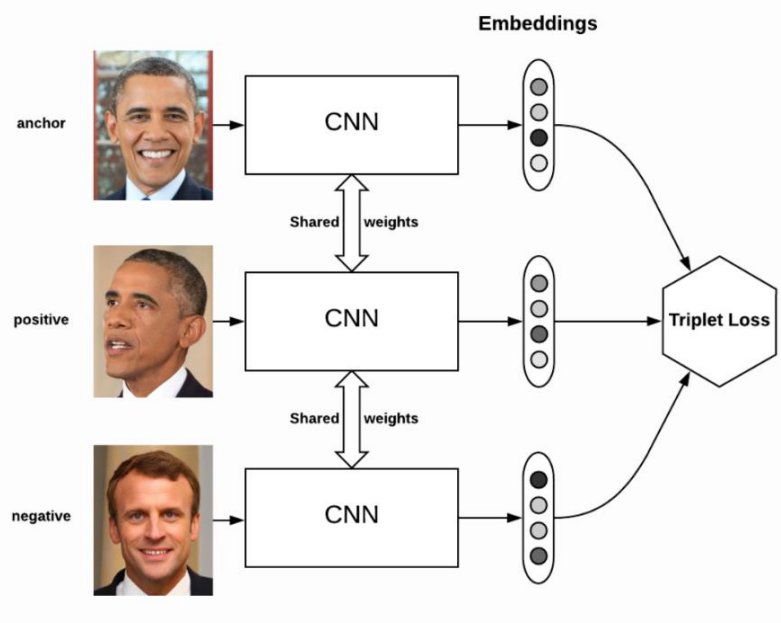

Figura 42. Representació d'una xarxa siamesa per entrenar amb la funció de triple pèrdua

Minimitzar la funció de pèrdua durant l'entrenament equivaldrà a aconseguir que d(a,p) s'apropi a 0 i d(a,n) sigui més gran que d(a,p) + margin.

La tria dels triplets és doncs un aspecte clau durant l'entrenament si és vol convergir ràpidament cap al mínim global. Per a fer-ho cal triar aquells triplets més difícils, d'aquesta forma dins un mateix batch (conjunt d'imatges de persones a avaluar), seleccionarem els parells positius de major distància (per minimitzar-la) i els parells negatius de menor distància (per maximitzar-la).

Basant-nos en la definició de la funció de pèrdua, existeixen tres tipus de triplets que podem utilitzar per entrenar:

- Easy triplets: Triplets on la funció de pèrdua val 0 perquè la distancia entre els parells positius més el marge no supera la distancia entre els parells negatius. d(a,p) + margin < d(a,n).

- Hard triplets: Triplets on la distància entre parells positius és menor a la distància entre parells negatius. d(a,n) < d(a,p).

- Semi-hard triplets: Triplets on la distància entre parells positius més el marge sí que supera la distància entre parells negatius. d(a,p) + margin > d(a,n).

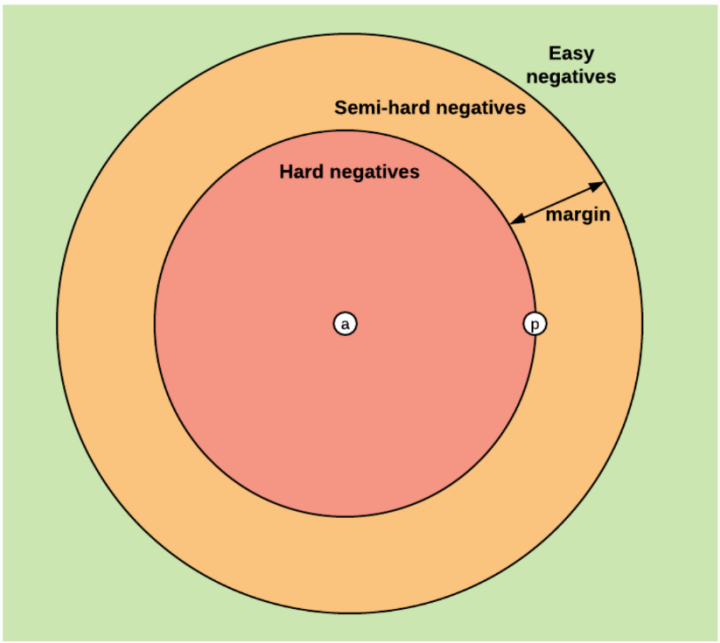

Figura 43. Tria dels triplets segons la variant utilitzada

5.4 Proposta

En aquest apartat és detalla la proposta realitzada per crear un sistema de reconeixement facial a partir d'un conjunt d'imatges reduït. Aquest sistema consisteix a utilitzar una xarxa neuronal convolucional (CNN) prèviament entrenada per extraure característiques i testejar-la com a sistema de control d'accés via reconeixement facial.

Atès que la xarxa pre entrenada no aconsegueix els resultats obtinguts però ja genera uns bons embeddings és millora la mateixa via l'entrenament triplet loss vist utilitzant semi-hard triplets. Els resultats després d'entrenar el model són molt millors. Obtenim menor distanciació entre embeddings de parells positius i major distanciació entre embeddings parells negatius. Gracies a aquest entrenament és pot finalment obtenir el sistema de control d'accés via reconeixement facial esperat.
A continuació és detalla tot el procés seguit així com l'anàlisi de cada resultat fins arribar a aquesta solució.

5.5 Etapes

5.5.1 Creació de la base de dades

Per a crear la base de dades s'utilitzen imatges capturades via web cam en dies aleatoris a companys de feina i amics. Les imatges obtingudes són guardades amb una resolució de 3840 x 2160 píxels, una mida que si bé és molt alta ens serveix per obtenir amb detall les cares un cop extretes de les mateixes imatges.

La lògica amb que és guarden les dades és de vital importància ja que al realitzar els diferents testos al llarg del projecte aquestes han d'estar guardades seguint una estructura concreta. La jerarquia amb que és guarden segueix dos directoris. El primer directori conté les imatges originals anteriors en que tenim cada usuari individualment capturat. Per cada usuari és crea un nou directori on trobem totes les seves imatges.

L'extracció de les cares de les fotografies es realitza via una xarxa neuronal convolucional anomenada MTCNN. Aquesta xarxa neuronal combina el potencial de les tasques de detecció de cares juntament amb la detecció de punts clau, el que ens permet realitzar la detecció de cares ràpidament. Concretament, les cares extretes és guarden en format .png i amb una resolució de 224 x 244 píxels. La mida és precisament la d'entrada que especifica l'arquitectura de la nostra xarxa neuronal. Tanmateix aquestes imatges generades afegeixen un marge de 44 píxels entre la cara i la mida de la imatge perquè, tal com indica al paper, la xarxa neuronal és comporta millor amb informació contextual de la cara.

Obtenim finalment el conjunt de cares extretes via MTCNN amb una mida de 224 x 224 píxels i un marge de 44 píxels a cada imatge. El conjunt de cares extretes, tal i com l'original del que s'originen, manté la jerarquia de les carpetes per cada usuari.

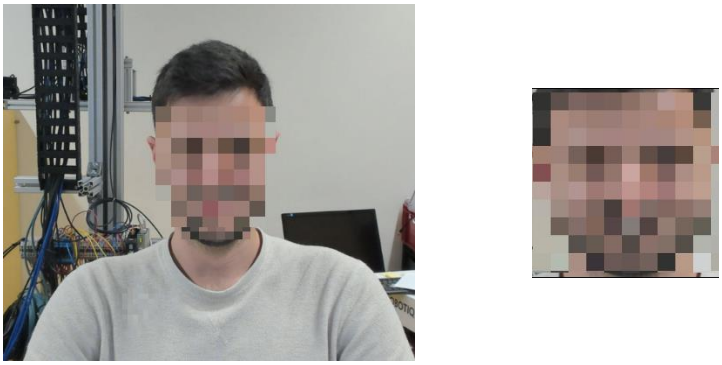

Figura 44. Imatge original i retallada per la xarxa neuronal MTCNN

Cal definir ara les tres carpetes principals dins tot conjunt de Machine Learning. Aquestes són les carpetes d'entrenament, validació i testatge que ja hem vist anteriorment al marc teòric previ i que ens serveixen per entrenar i validar el model respectivament. Per què aquestes tres carpetes siguin representatives el que fem és construir-les agafant imatges aleatòries, això vol dir que tindrem informació aleatòria per cada usuari dins un mateix conjunt de manera que les probabilitats de que el nostre model final entrenat generalitzi, són més altes.

- Conjunt de dades d'entrenament: 25 imatges aleatòries per persona.
- Conjunt de dades de validació: 10 imatges aleatòries per persona.
- Conjunt de dades de testatge: 15 imatges aleatòries per persona.

La forma més senzilla d'utilitzar aquesta divisió de forma automàtica és utilitzar el paquet split-folders dins una línia de comandes Python. A través d'aquest paquet podem dividir les 50 imatges que tenim per cada usuari en les tres carpetes anteriors.

```
>>> import splitfolders
>>> splitfolders.ratio(inputfolder, output="dataset",seed=42,ratio=(.5,.20,.3),group_prefix=None)
```

Figura 45. Utilització del paquet split-folders per dividir el set de dades utilitzat

A través de la comanda anterior hem dividit el conjunt original en un 50% de les imatges al conjunt d'entrenament, un 20% de les imatges al conjunt de validació i un 30% restant al conjunt d'entrenament de forma que ja tenim les 25 imatges per persona per entrenar el model, les 10 imatges per validar-lo i les 15 restants per testejar-lo respectivament. Tanmateix, s'ha ampliat generosament el conjunt de dades de testeig a fi de donar mesures acurades de la bona generalització dels diferents models entrenats. La divisió dels conjunts de dades en funció de percentatges té sentit en conjunts de dades massius però no per aquest cas on treballem amb conjunts de dades molt reduïts. El conjunt de dades de testeig queda finalment amb 60 imatges per persona.

Figura 46. Mostra de cares extretes a un usuari dins el conjunt d'entrenament

5.5.2 Primer contacte amb el model

El model pre entrenat (original) del que és parteix ja realitza una bona extracció de característiques de cada usuari. Podríem pensar que, amb aquests embeddings podem ja establir llindars per classificar correctament. Tot i així, és freqüent que aquest classificació falli en una execució a temps real. Això succeeix perquè el model, per molt ben entrenat que estigui per extreure característiques, està entrenat amb altres imatges i mai les del nostre conjunt amb la possible i més que probable conseqüència de no ajustar-se a les nostres imatges.

Per analitzar el comportament del d'aquest model, podem en primera instància mostrar la distància euclidiana o el cosine similarity entre els embeddings generats per cada usuari. Recordem que uns embeddings similars per a imatges d'un mateix usuari son símptoma d'una bona generació de característiques facials mentre que una similitud dels mateixos entre embeddings de diferents usuaris ens pot indicar un mal model.

El que precisament veiem utilitzant el model original és que les distancies entre embeddings d'imatges de la mateixa classe no són tan similars com pensàvem. Alhora, els embeddings entre imatges d'igual classe no són tant diferents.
El que esperem del gràfic és veure menor distància (blau) entre imatges de la mateixa classe i major distància (groc) entre imatges de classe diferents.

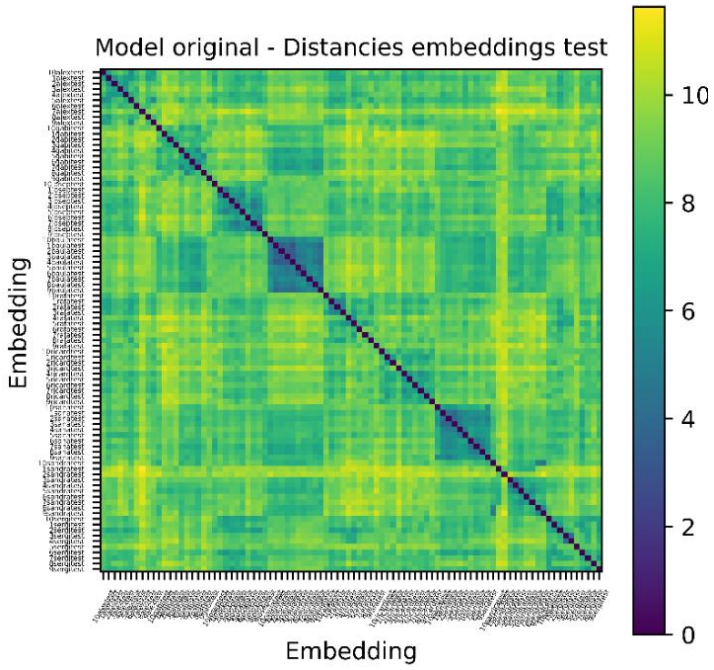

Figura 47. Distàncies entre embeddings generats pel model original

Tanmateix analitzar el comportament del model via els embeddings que genera la xarxa neuronal ens serveix de primera referència del seu comportament però no és un indicatiu vàlid. Podríem tenir dues classes diferents amb els embeddings agrupats, molt properes entre sí però també propers entre embeddings de les altres classes de manera que entre classes diferents obtinguem una distància petita. Es doncs incorrecte analitzar el comportament del model d'aquesta manera.

El que hem de fer realment és establir un llindar de distància euclidiana o cosine similarity entre embeddings a partir del qual podem establir quan un embedding pertany o no a una certa classe. Podem establir manualment aquest llindar però la forma correcte passa per córrer un test automàtic per extreure el millor llindar de distància euclidiana o cosine similarity.

Realitzem un test de cerca de millor llindar de distància euclidiana entre embeddings a partir del qual és realitzin correctament el major nombre de classificacions. El test utilitzarà 2000 parells d'imatges aleatòries dins el conjunt de testatge, tot i que en aquest cas el model no ha vist cap de les imatges del nostre conjunt i podríem utilitzar per tant, qualsevol conjunt. El test cercarà el millor llindar provant valors d'entre 6 fins a 12'5 amb salts de 0'5.

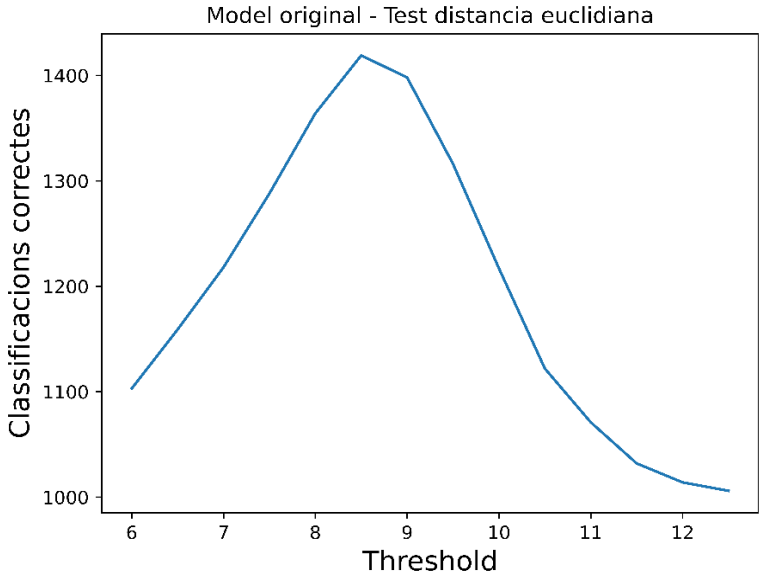

Figura 48. Test de millor llindar euclidià pel model original

Realitzat el test obtenim que el millor llindar de distància euclidiana per decidir la correspondència d'un embedding a una classe és 8'5, amb un nombre de classificacions correctes de 1419 i aconseguint una precisió molt baixa de 70'95 %.

Realitzem ara el mateix test però cercant el millor llindar via cosine similarity entre embeddings. El test utilitzarà com abans 2000 parells d'imatges del conjunt de testatge i llindars d'entre 0'2 i 0'8 amb salts de 0'05.

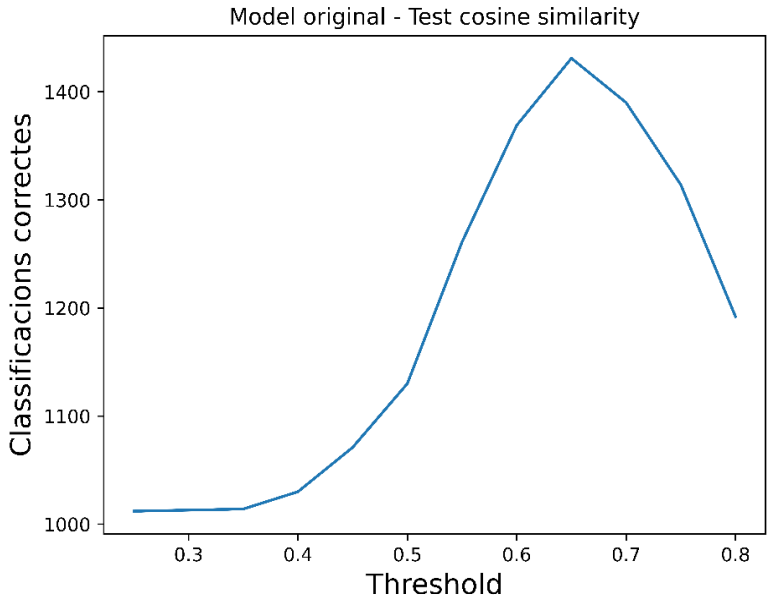

Figura 49. Test de cerca del millor llindar de distància cosinusoïal pel model original

El que obtenim en aquest cas és que el millor llindar separador via cosine similarity entre embeddings és 0'65 amb un nombre de classificacions correctes igual a 1431 obtenint una precisió més baixa a l'anterior igual a 71'55 %.

Un cop cercat el millor llindar el que podem fer per analitzar correctament qualsevol model de xarxa neuronal és la matriu de confusió.
La matriu de confusió ens ajuda a comprendre com és comporta el model segons les prediccions extretes donades unes entrades. Recordem que ens calia establir el llindar classificador per decidir quan un embedding pertany o no a una certa classe. Abans analitzem la informació que ens dona la matriu de confusió:

- Matrius de confusió individuals: Ens ajuden a entendre com és comporta el model en una classificació binària per una sola classe (positive) respecte les altres (negative).
- True positives: Casos on classifiquem correctament una entrada positiva.
- True negatives: Casos on classifiquem correctament una entrada negativa.
- False positives: Casos on classifiquem incorrectament una entrada negativa.
- False negatives: Casos on classifiquem incorrectament una entrada positiva.
- Accuracy: La mètrica més simple, intuïtiva i representativa. El rati de casos correctament classificats respecte tots els casos.
- Misclassification: Equivalent a 1 - accuracy. Ens informa del rati dels incorrectament classificats respecte tots els casos.
- Sensitivity o recall: Ens informa del rati de classificacions correctes quan l'entrada és positiva. L'objectiu és minimitzar falsos negatius i que la mètrica s'apropi a 1.
- False positive rate: Ens informa del rati de classificacions incorrectes quan l'entrada és negativa. L'objectiu és minimitzar falsos positius i que la mètrica s'apropi a 0.
- Specify: Ens informa del rati de classificacions correctes quan l'entrada és negativa. L'objectiu és minimitzar falsos positius i que la mètrica s'apropi a 1.
- False negative rate: Ens informa del rati de classificacions incorrectes quan l'entrada és positiva. L'objectiu és minimitzar falsos negatius i que la mètrica s'apropi a 0.
- Precision: Rati de prediccions correctes d'entrades positives respecte el total de prediccions positives. L'objectiu és minimitzar falsos positius i que la mètrica s'apropi a 1.
- F-measure: Combina recall i precision. L'objectiu és minimitzar falsos positius i falsos negatius perquè que la mètrica s'apropi a 1.

Per realitzar la matriu de confusió utilitzarem tota la base de dades d'imatges d'usuaris de testatge i els llindars de distància euclidiana i cosine similarity anteriors.
Atès que en un espai tridimensional podem tenir embeddings molt distanciats respecte els embeddings de la mateixa classe el que fem és també analitzar el model amb dos tipus embeddings representatius.
El primer embedding representatiu de cada classe serà l'embedding calculat com a mitja de tots els embeddings de la classe. El segon embedding representatiu de cada classe serà el que minimitza la distància a tota la resta d'embbedings de la mateixa classe. S'intenta evitar d'aquesta forma que una mala generació d'embbeding afecti al còmput de l'embbeding representatiu de la classe.

Mètriques de la matriu de confusió via distància euclidiana 8'5 i embeddings mitjans

Model original	Predicted values		
Actual values		Positive	Negative
	Positive	307	233
	Negative	233	4087
Accuracy: 0'57			

Mètriques de la matriu de confusió via distància euclidiana 8'5 i embeddings que minimitzen la distància

Model original	Predicted values		
Actual values		Positive	Negative
	Positive	276	28
	Negative	35	4292
Accuracy: 0'51			

Mètriques de la matriu de confusió via cosine similarity 0'65 i embeddings mitjans

Model original	Predicted values		
Actual values		Positive	Negative
	Positive	312	228
	Negative	228	4092
Accuracy: 0'58			

Mètriques de la matriu de confusió via cosine similarity 0'65 i embeddings que minimitzen la distància

Model original	Predicted values		
Actual values		Positive	Negative
	Positive	292	248
	Negative	248	4072
Accuracy: 0'54			

El resultat de les diferents matrius de confusió ens informen clarament d'un mal comportament del model que tan sols pot arribar a un accuracy de 0'58 utilitzant embeddings representatius mitjans de cada classe i un llindar via cosine similarity igual a 0'65. L'accuracy del model, tal com ja anàvem veient, no és l'esperat. Ens cal doncs partir d'aquest model i millorar-lo via l'entrenament triplet loss.

5.5.3 Entrenament del model

L'entrenament via la funció de pèrdua triplet loss és la peça més important de l'entrenament. Aquesta funció de pèrdua ens garanteix un ràpid entrenament ja que magnifica la distància entre dos embeddings de classes diferents i apropa els embeddings de classes similars en una mateixa iteració. L'entrenament és doncs molt ràpid i alhora permet entrenar un model de xarxa neuronal CNN amb molt poques dades. Estem entrenant el model amb tan sols 25 imatges per persona i en un temps rècord. Per analitzar el comportament d'aquest entrenament és realitzen tres entrenaments i s'obtenen tres models.
Les característiques de l'entrenament de cada model són les següents:

Model 1		
Triplet loss margin 0'5	Epochs 400	Validation triplets 300
Batch size 9	Training triplets 500	Learning rate 0'001
Learning rate decay: 0'01 cada 50 epochs		

Model 2		
Triplet loss margin 0'9	Epochs 800	Validation triplets 1000
Batch size 9	Training triplets 2000	Learning rate 0'001
Learning rate decay: 0'01 cada 50 epochs		

Model 3		
Triplet loss margin 0'9	Epochs 1000	Validation triplets 3000
Batch size 9	Training triplets 5000	Learning rate 0'001
Learning rate decay: 0'01 cada 75 epochs		

S'espera que els models que utilitzen més marge a la funció de pèrdua ens permeti classificar millor pel fet de distanciar més els embeddings de classes diferents i apropar els embeddings de classes similars. Per altra banda, l'últim model s'entrena amb més epochs i més triplets i per tant s'espera que funcioni millor a diferència del segon model, entrenat amb un molt menor nombre de triplets i de epochs.
Un cop entrenats els models, analitzem en primer lloc les distàncies entre els embeddings del conjunt de testatge generats per cada model.

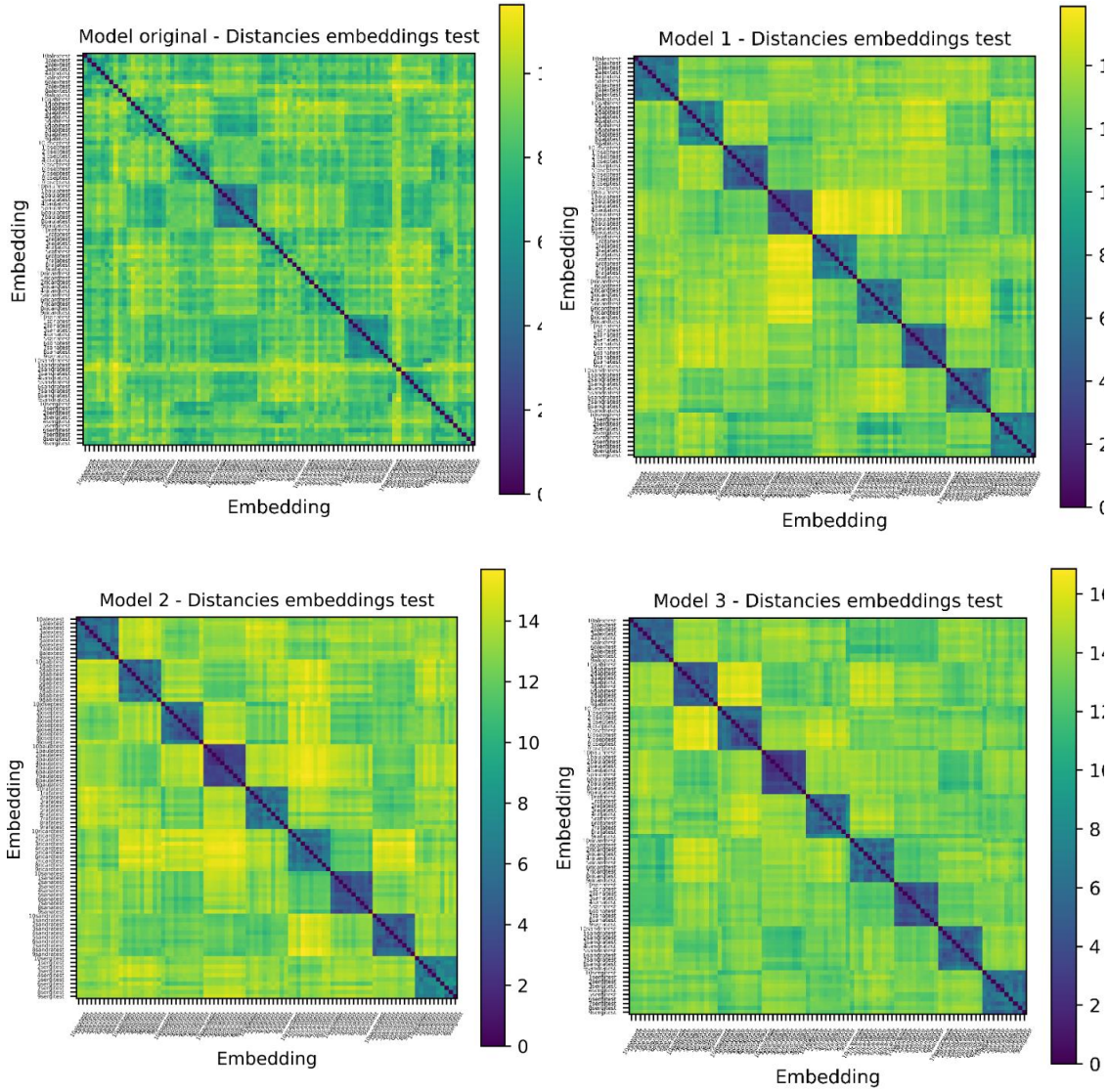

Figura 50. Distàncies entre embeddings generades per cada model

Podem veure clarament la diferencia entre els model original i la resta de models amb una clara distanciació del embeddings entre classes diferents i una aproximació dels embeddings de classes iguals. Pel que fa a diferències entre models entrenats, veiem com la diferència del valor del marge utilitzat per entrenar entre els models 1 i 2 / 3 fa que varin les distàncies entre embeddings generats per cada model. Entre el model 2 i 3, tenim alhora una millor distanciació dels embeddings. Això és deu a que s'han utilitzat més triplets per entrenar i més epochs. Alhora el l'entrenament del model 3 utilitza un learning rate que es minimitza cada 75 epochs i no cada 50 epochs com al model 2. Això fa aproximar-nos més al possible mínim global de la funció.

Cal considerar també que el model 1 i 2 s'han entrenat en 30 minuts i 2 hores respectivament, mentre que el model 3 ha estat entrenat en aproximadament 8 hores.

5.5.4 Validació del model

Sembla que els models entrenats rendiran millor considerant els resultats anteriors. La prova definitiva però és crear les matrius de confusió a fi d'analitzar com realment és comporta el model. Per a fer-ho, cerquem en primer lloc els millors llindars separadors via testos de distància euclidiana i testos de cosine similarity.

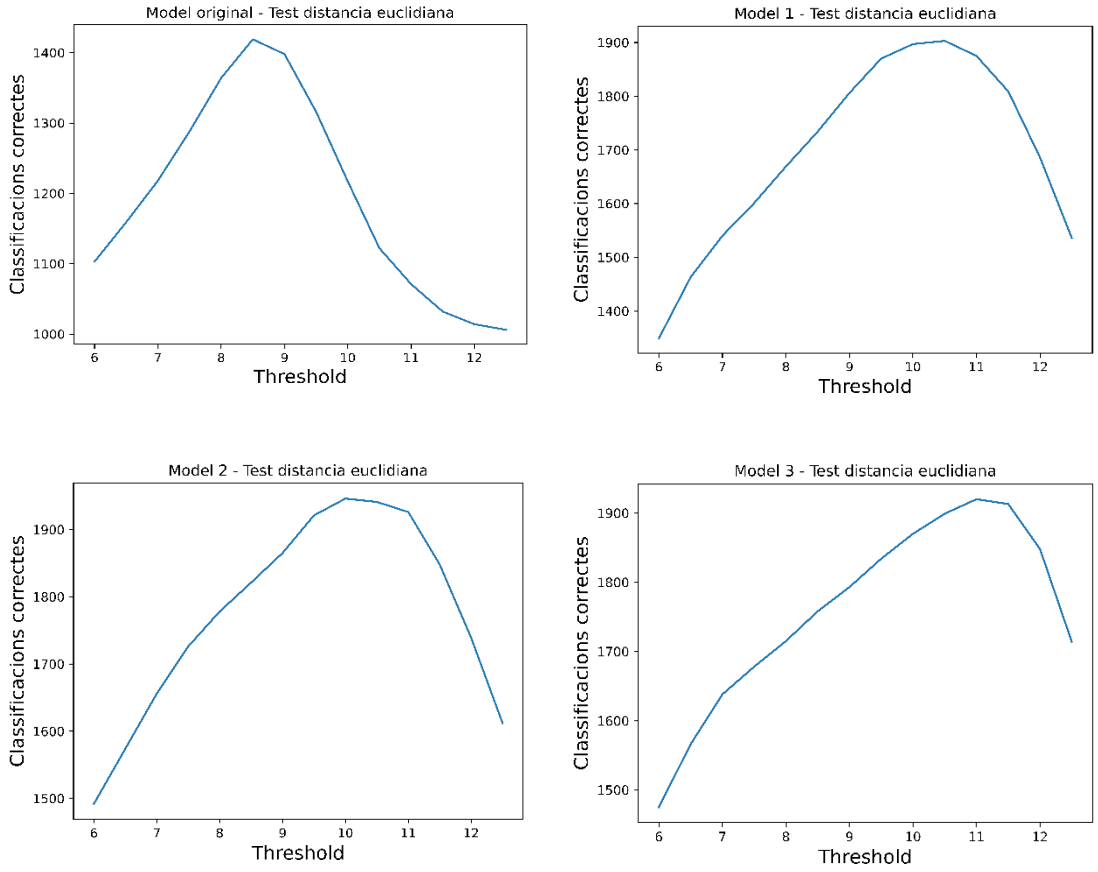

Figura 51. Cerca de la millor distància euclidiana classificadora per cada model

Clarament podem veure com als models entrenats el resultat en nombre de classificacions correctes és molt major. El canvi és més notori entre el model original i el model 1, mentre que entre tots els models entrenats el nombre de classificacions correctes supera en qualsevol cas les 1900. Finalment al model 2 veiem el major nombre de classificacions correctes obtingudes, 1946 de 2000 possibles aconseguint un 97'3 % de precisió amb un llindar de 10'0.

Busquem ara el millor llindar separador via cosine similarity.

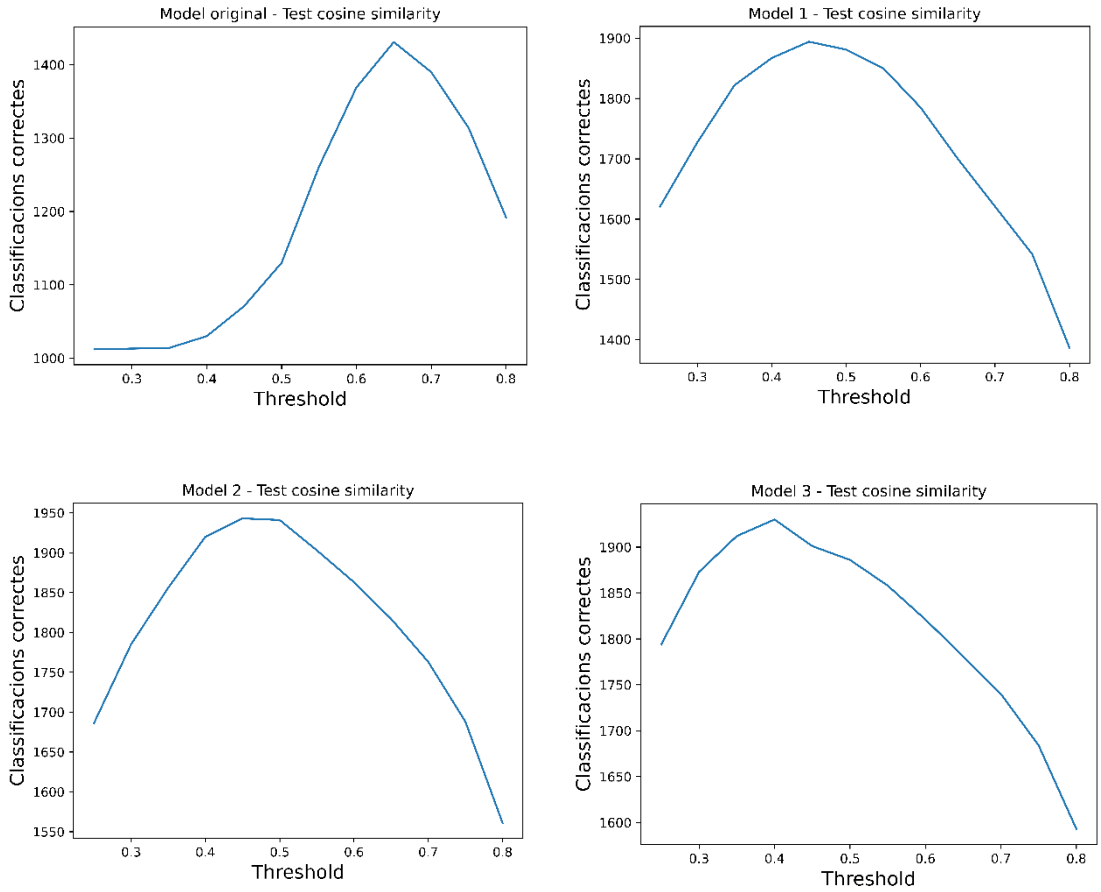

Figura 52. Cerca del millor llindar cosinusoïal per cada model

Com ja havíem vist amb el test de llindar via distància euclidiana aconseguim que els models entrenats és comportin molt millor que el model original. Veiem com anteriorment que el model 2 obté el major nombre de classificacions correctes, 1943 de 2000 possibles aconseguint un 97'15% de precisió amb un llindar de 0'45. El model 2 és el que millor ens servirà per classificar posteriorment precisament per aquesta gran precisió aconseguida.

El model 3 en canvi semblaria que hauria de tenir el major nombre de classificacions correctes. Tanmateix podem pensar que ha estat sobre entrenat i s'ha ajustat massa a les dades del conjunt d'entrenament fruit de realitzar masses iteracions sobre aquestes durant l'entrenament que ha durat 6 hores més respecte el model 2, que n'ha durat dues.

Aquest és un resum dels resultats de les classificacions anteriors:

Model	Classificacions	Accuracy	Best Euclidian Thresh
Model original	1419/2000	70’95 %	8’5
Model 1	1903/2000	95’15 %	10’5
Model 2	1946/2000	97’3 %	10’0
Model 3	1920/2000	96 %	11’0

Model	Classificacions	Accuracy	Best Cosine Thresh
Model original	1431/2000	71’55 %	0’65
Model 1	1894/2000	94’7 %	0’45
Model 2	1943/2000	97’15 %	0’45
Model 3	1930/2000	96’5 %	0’4

Ara ja podem construir les matrius de confusió de cada model amb els valors de distància euclidiana o cosine similarity que millor s’han ajustat per classificar correctament els embeddings.

Les matrius de confusió via embeddings mitjans representatius de cada classe usant un llindar de distància euclidiana són:

Model original	Predicted values		
Actual values		Positive	Negative
	Positive	307	233
	Negative	233	4087
Accuracy: 0’57			

Model 1	Predicted values		
Actual values		Positive	Negative
	Positive	511	29
	Negative	29	4291
Accuracy: 0’95			

Model 2	Predicted values		
Actual values		Positive	Negative
	Positive	532	5
	Negative	8	4315
Accuracy: 0’99			

Model 3	Predicted values		
Actual values		Positive	Negative
	Positive	513	27
	Negative	27	4293
Accuracy: 0’95			

Les matrius de confusió via embeddings representatius de cada classe que minimitzen la distància usant un llindar de distància euclidiana són:

Model original	Predicted values		
Actual values		Positive	Negative
	Positive	276	264
	Negative	264	4056
Accuracy: 0’51			

Model 1	Predicted values		
Actual values		Positive	Negative
	Positive	505	28
	Negative	35	4292
Accuracy: 0’94			

Model 2	Predicted values		
Actual values		Positive	Negative
	Positive	524	3
	Negative	16	4317
Accuracy: 0’97			

Model 3	Predicted values		
Actual values		Positive	Negative
	Positive	514	20
	Negative	26	4300
Accuracy: 0’95			

De les anteriors matrius de confusió podem extreure que tenim uns molt bons models entrenats destacant el model 2. Alhora veiem com utilitzar un embedding que minimitza la distància a la resta de embeddings de la mateixa classe, funciona pitjor que utilitzar un embedding representatiu mitjà de cada classe. La diferencia és poca però a cada model tenim menys error si utilitzem els embeddings mitjans.

Veiem ara com és comportaria el model usant llindars de cosine similarity.

Les matrius de confusió via embeddings mitjans representatius de cada classe usant un llindar de cosine similarity són:

Model original	Predicted values		
Actual values		Positive	Negative
	Positive	312	228
	Negative	228	4092
Accuracy: 0’58			

Model 1	Predicted values		
Actual values		Positive	Negative
	Positive	512	28
	Negative	28	4292
Accuracy: 0’95			

Model 2	Predicted values		
Actual values		Positive	Negative
	Positive	532	7
	Negative	8	4313
Accuracy: 0’99			

Model 3	Predicted values		
Actual values		Positive	Negative
	Positive	512	24
	Negative	28	4296
Accuracy: 0’95			

Les matrius de confusió via embeddings representatius de cada classe que minimitzen la distància usant un llindar de cosine similarity són:

Model original	Predicted values		
Actual values		Positive	Negative
	Positive	292	248
	Negative	248	4072
Accuracy: 0’54			

Model 1	Predicted values		
Actual values		Positive	Negative
	Positive	505	28
	Negative	35	4292
Accuracy: 0’94			

Model 2	Predicted values		
Actual values		Positive	Negative
	Positive	532	5
	Negative	8	4315
Accuracy: 0’99			

Model 3	Predicted values		
Actual values		Positive	Negative
	Positive	514	20
	Negative	26	4300
Accuracy: 0’95			

Com abans veiem com els models entrenats és comporten millor. Alhora veiem com l’ús d’embbedings representatius mitjans de cada classe fa que es comporti millor el model per classificar.

Tal i com esperàvem podem concloure que el model 2, ja sigui utilitzant distancia euclidiana com a llindar classificador o cosine similarity, és el millor model. El seguent millor model, el model 3, ha estat entrenat amb més epochs, més triplets i més temps, aproximadament 6 hores més resultant en el que podem concloure com un model sobre ajustat.
Finalment ens quedarem amb el model 2. Utilitzarem embeddings representatius mitjans per cada classe i un llindar classificador de distància euclidiana de 10’0 atès que és el model que major nombre de classificacions correctes ha obtingut.

Finalment hem pogut utilitzar un model pre entrenat i l'hem ajustat a les nostres dades utilitzant un entrenament via la funció de pèrdua triplet loss.
Fem retrospectiva per analitzar com de bé funciona el model entrenat amb les nostres dades respecte al model pre entrenat original.

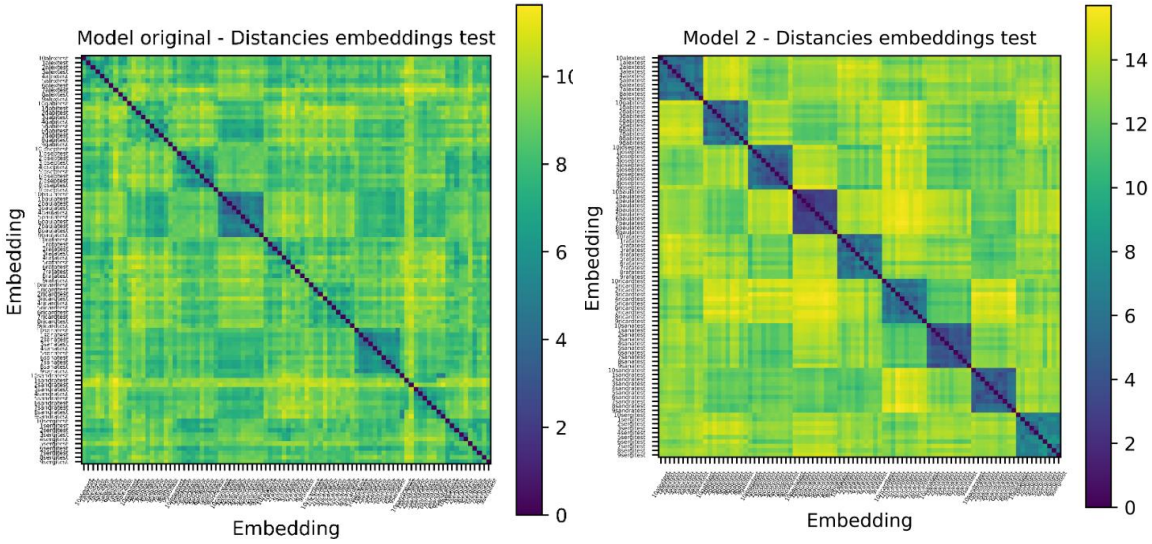

Figura 53. Embeddings generats pel model original i model 2

Pel que fa als embeddings generats és molt fàcil veure la gran millora entre model original i entrenat. L'entrenament via triplet loss ens ha assegurat que es realitza una minimització en la distància entre embeddings de la mateixa classe mentre que alhora es distancien els embeddings de classes diferents. El resultat és veu fàcilment als gràfics on pel model pre entrenat tenim un patró de distàncies euclidianes entre embeddings quasi aleatori i al gràfic del model entrenat veiem clarament el color blau representatiu de distància euclidiana mínima entre embeddings de la mateixa classe i el color proper a groc representatiu de distància maximitzada entre embeddings de classes diferents.

Considerant que els embeddings que genera la nostra xarxa neuronal no son res mes que vectors de 128 dimensions, podem representar els seus valors gràficament. Ho veiem a continuació.

El model pre entrenat original generava unes bones característiques representatives però recordem que no ha estat entrenat amb el nostre conjunt i per tant no s'ajusta al 100% a les nostres dades. En canvi, després de realitzar l'entrenament, la nostra xarxa ha ajustat correctament els pesos per tal de generar uns embeddings representatius que son molt semblants per una mateixa classe i alhora son diferents per una classe diferent.
Si representem tots els embeddings generats per la classe 'Sergi' a través del model original o a través del millor model entrenat, el model 2, veiem clarament una millor representació amb aquest últim model.

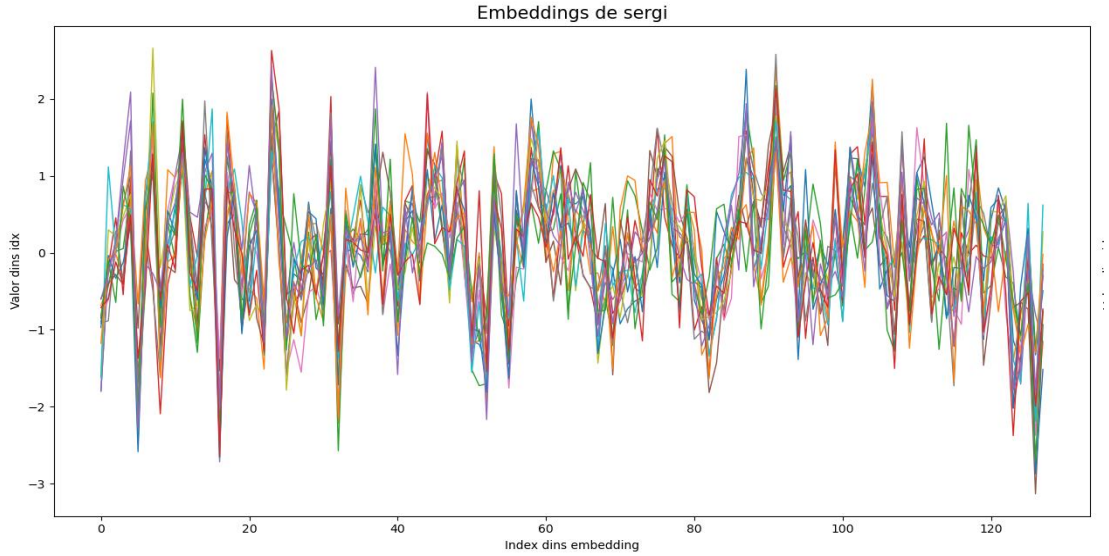

Figura 54. Embeddings generats per la classe 'Sergi' a través del model original

Veiem ara la representació dels embeddings de la classe 'Sergi' que realitza el model 2.

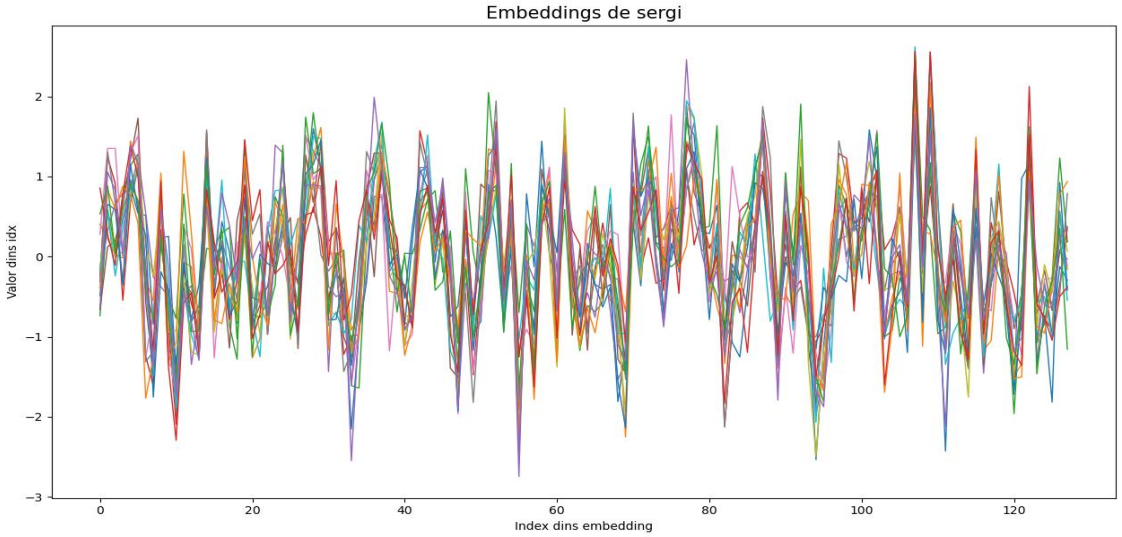

Figura 55. Embeddings generats per la classe 'Sergi' a través del model 2

Com abans, veiem com la representació dels embeddings és molt millor un cop entrenat el model. Cada un dels valors del vector de característiques és únic per cada classe i es diferencia de la resta. Al model original veiem un cert desordre d'aquestes característiques, el que realment esta passant és que cap classe esta ben representada.

Imaginant els vectors en un espai tridimensional, podem imaginar els vectors com punts en l'espai. La representació del núvol de punts per una classe seria molt distant i aleatòria en canvi utilitzant el model entrenat aquests punts o embeddings s'haurien apropat per una mateixa classe. En conseqüència, el vector representatiu de característiques mitjà de cada classe és molt més aleatori al model original que al model entrenat.

El test de cerca de millor llindar classificador via distància euclidiana ens ha reforçat el concepte anterior. Veiem clarament una millora als gràfic atès que els embeddings s'han separat o ajuntat en un espai euclidià i això ha fet que tinguem més marge en distància euclidiana classificadora.

Model	Classificacions	Accuracy	Best Euclidian Thresh
Model original	1419/2000	70'95 %	8'5
Model 2	1946/2000	97'3 %	10'0

Pel model original el millor resultat aconseguit és de 1946 classificacions de parells d'imatges correctes de 2000 donades amb un 97'3 % de precisió utilitzant un llindar de 10'0. En contra tenim el model entrenat que aconsegueix 1419 classificacions correctes de 2000 possibles amb una precisió del 70'95 % i un llindar de 8'5.

És important veure que aquest llindar és el primer que ha aconseguit aquest nombre de classificacions correctes tot i que gràcies a l'entrenament via triplet loss. Podem concloure que també obtindrem un molt bon resultat utilitzant un llindar de distància euclidiana d'entre 9'5 a 11 mentre que al model original només se'ns permetria utilitzar un llindar de 8'5.

Establir el menor llindar, assegura que la generació de embeddings per una imatge captada per la càmera a temps real hagi de ser molt propera a una classe a fi de considerar-se i classificar-se com d'aquesta. Establir un major llindar ens aportarà major error al sistema de control d'accés ja que qualsevol classe podria ser similar a les altres a una certa distància.

Finalment, utilitzant els llindars anteriors, hem validat el model entrenat creant una matriu de confusió que ens reforça el bon funcionament del model entrenat.

Model original	Predicted values		
Actual values		Positive	Negative
	Positive	307	233
	Negative	233	4087
Accuracy: 0’57			

Model 3	Predicted values		
Actual values		Positive	Negative
	Positive	532	5
	Negative	8	4315
Accuracy: 0’99			

Ara que ja tenim el model entrenat i n’hem validat el seu comportament tan sols ens queda una cosa, utilitzar-lo en un cas real. Creem un test per veure com és comporta el model donats parells d’imatges aleatoris. Establint el millor llindar trobat, fem que el model classifiqui entre si les imatges donades son d’igual o diferent classe.

Figura 56. Classificacions i distàncies entre parells d’imatges pel model 2

A la figura següent anterior comprovem clarament com el model 2 es comporta correctament establint un llindar de 10’0 a partir del qual dues imatges son considerades de classe diferent. Podem concloure que el nostre model està perfectament entrenat.
Hem trobat la millor representació de cada classe (embedding mitjà) i el millor classificador (llindar euclidià de 10’0).

Ja podem crear el nostre sistema de control d’accés via reconeixement facial en temps real. Per a fer-ho, tan sols caldrà crear un sistema capaç de detectar les cares als fotogrames de vídeo, extreure-les, passar-les a la xarxa neuronal i donar-ne la predicció.

Tenim totes les eines. MTCNN ens permetrà detectar les cares i un cop extretes, la xarxa neuronal que hem entrenat ens donarà la resposta adequada a cada cara extreta, el millor embedding representatiu generat que podrem utilitzar per classificar segons si s’apropa o no a un dels possibles embeddings representatius mitjans guardats que tenim de cada usuari.

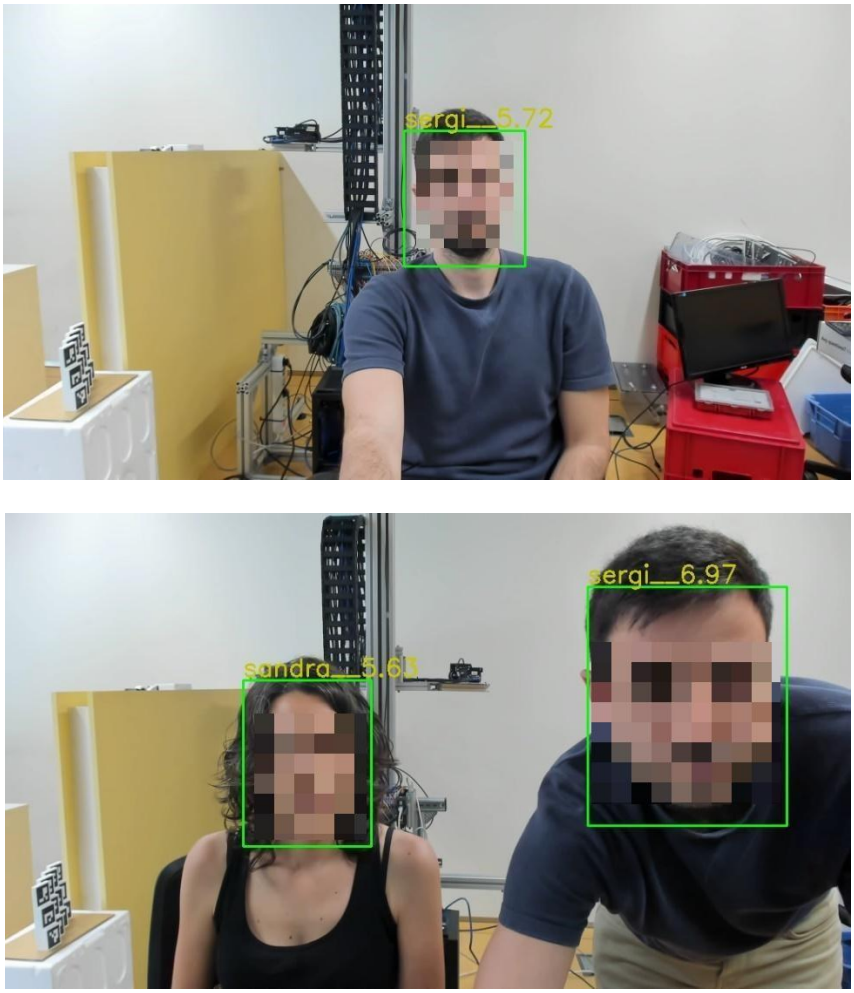

Figura 57. Imatges del sistema de reconeixement utilitzant el model 2

Hem vist com efectivament el sistema de reconeixement facial funciona. El model creat ens aporta una solució molt pràctica per resoldre el problema del control d'accés per exemple en l'àmbit empresarial.

5.5.5 Cerca d'un millor classificador

A la pràctica, utilitzant el sistema a temps real (30 imatges per segon), es freqüent veure com en algun fotograma el reconeixement facial no obté la resposta esperada. Això ens pot passar varies raons:

- La imatge captada per la càmera té prou qualitat o no s'ajusta en rati d'aspecte a les utilitzades per entrenar el model. D'aquesta forma la imatge que arriba a l'entrada del model no és realment representativa de l'usuari a identificar.
- La distanciació entre embeddings dels usuaris a l'espai no és constant, fet que impossibilita l'ús d'un classificador amb un llindar d'igual valor ja sigui amb distancia euclidiana o cosinusoïal igual.

En el primer cas, tan sols hem d'assegurar que la imatge d'entrada sigui conseqüent amb la que espera el model. Pel segon cas, podem repetir l'entrenament ja que aquest no és gens costós ni en capacitat de còmput ni temporalment. Pel tercer cas el més útil serà reduir el llindar utilitzat per aportat més robustesa a les identificacions, tal com ja havíem vist.

Cal ser conscients que el nostre problema esta adaptat per a resoldre la identificació en un conjunt reduït d'usuaris, 9 persones concretament. Per aquest cas i després de reduir el llindar per identificar una persona podem obtenir un molt bon sistema de reconeixement. Tanmateix utilitzar aquest llindar de distància fixa per classificar no és la millor de les pràctiques i es que imaginant els embeddings com a núvols de punts a l'espai és fàcil veure com aquests no estan distanciats en un mateix valor segons cada possible parell de classes.

A mesura que el nostre sistema avanci amb més usuaris serà freqüent caure en l'error i en aquest cas on ja no podrem classificar amb un llindar de valor fixe en distància entre tots els usuaris. Ens caldrà variar la distància separadora entre els diferents embeddings de cada classe per resoldre la identificació. En definitiva el que ens caldrà utilitzar és un classificador de suport vectorial.

Un model de suport vectorial (SVM) s'utilitza per resoldre problemes de classificació i regressió. El model separa les dades corresponents a cada classe mitjançant un hiperplà definit entre els dos punts més propers entre dues classes anomenat vector de suport. SVM és per tant un classificador binari. Quan una nova mostra arriba al model classificador, aquesta queda classificada segons a quina part de l'hiperplà correspon.

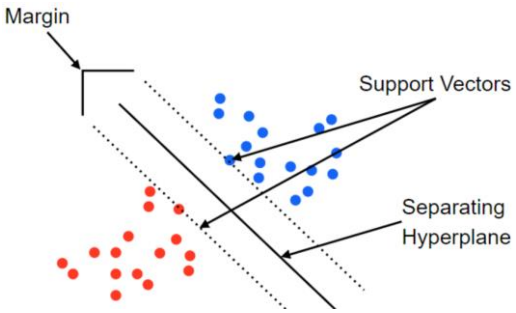

Figura 58. Mostra d'un híper pla separador per una classificació binaria

Seguint l'esquema anterior, el que podem fer per classificar en una representació multi classe (SVC) és la construcció d'un classificador binari per cada parell de classes. D'aquesta forma obtindrem un classificador multi classe on per cada classe assegurem el millor llindar separador de valor variant segons cada parell de classes i no utilitzem un llindar de valor fixe tal com estàvem fent als classificadors via distancia euclidiana o cosinusoïal anteriors.

Aquesta és la matriu de confusió resultat d'aplicar el SVC al model 2:

Model 2	Predicted values		
Actual values		Positive	Negative
	Positive	535	5
	Negative	5	4315
Accuracy: 0'99			

Com és d'esperar, l'ús d'aquest tipus de classificador ens aporta la màxima precisió que ja havíem obtingut via distancia euclidiana fixa de 10'0.
Cal ser conscients que l'obtenció de la màxima precisió obtinguda usant un llindar de valor fixe és fruit de la gran distància entre els embeddings que obté el model. Alhora si el mateix model entrenat s'utilitzés per resoldre la identificació amb un conjunt més gran d'usuaris veuríem l'error i en contra el bon funcionament utilitzant el model de suport vectorial.

Hem obtingut el model generador d'embeddings esperat després de entrenar via triplet loss un model pre entrenat. Gràcies a la bona generació embeddings, hem trobat el millor llindar separador útil per resoldre la identificació d'un usuari. Finalment hem validat el comportament del model i n'hem millorat la classificació utilitzant un classificador de suport vectorial.

Els resultats en temps real passen a ser més sòlids mitjançant el classificador SVC. Ens adaptem en distància euclidiana separadora segons cada parell de classes on es crea el classificador binari. Cal recordar però que el vector de suport es crea entre els punts més propers de dues classes i per tant es molt important novament que l'entrenament hagi arribat al mínim global de la funció de pèrdua assegurant la millor extracció de característiques per cada usuari i per tant el millor llindar separador variant entre classes. El model obtingut finalment resol el problema d'identificació d'usuaris. Alhora, si volguéssim afegir un nou empleat, tan sols hauríem d'afegir les seves imatges al sistema, de tal forma que al iniciar el reconeixement facial el sistema generi embeddings per aquest nou usuari i es generin nous hyper plans separadors utilitzant la nova classe entrada pel nou usuari.

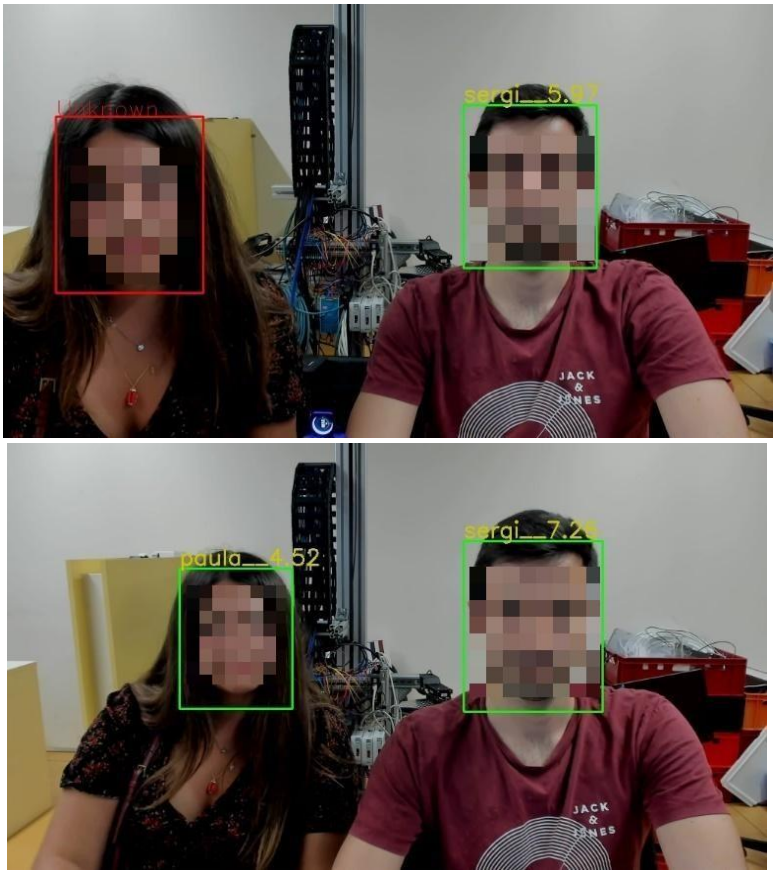

Figura 59. Reconeixement a temps real abans i després d'afegir l'usuari

El sistema de reconeixement facial obtingut és dons un sistema molt robust entrenat amb un conjunt reduït d'imatges per persona i que no requereix cap tipus de re-entrenament per afegir nous usuaris.

En qualsevol cas, si el sistema no fos prou robust, tan sols caldria tornar a entrenar el model de forma que sigui capaç de generar una representació més acurada de les dades facials dels nous usuaris.

CAPÍTOL 6

Conclusions i treball futur

6.1 Valoració personal

La realització d'aquest projecte ha resultat una experiència molt satisfactòria.

La gran quantitat de conceptes nous apresos sobre la intel·ligència artificial i l'aprenentatge automàtic per realitzar el projecte han resultat d'interès personal i de base futura per a l'estudi del màster en intel·ligència artificial, que em genera ara més interès.

Aquest tipus de conceptes, excepte els de visió per computador, no han estat estudiats en profunditat durant el transcurs de la carrera. En conclusió, la realització del projecte m'ha aportat una gran base per cursar el màster d'intel·ligència artificial on son introduïts tots aquests conceptes.

La realització del projecte m'ha fet experimentar en primera persona l'elevat cost computacional que poden tenir les xarxes neuronals així com el seu potencial alhora de resoldre problemes ja siguin amb poques o moltes dades.

6.2 Futures ampliacions i millores

El desenvolupament del projecte s'ha centrat en la recerca i aplicació de l'aprenentatge automàtic en un sistema de reconeixement facial. Tanmateix hi ha múltiples factors que poden ser explotats a fi d'obtenir un millor sistema o variar del mateix.

Per exemple es poden desenvolupar múltiples aplicacions a fi d'analitzar l'entrenament del model i validar que s'arribi al mínim global de la funció de pèrdua.

L'anàlisi d'ús dels diferents tipus triplets per entrenar un millor model pot ser també objecte d'estudi.

Un altre factor a estudiar és l'anàlisi d´us dels múltiples classificadors existents. També es podrien explorar més mètodes d'augment de dades (data augmentation) per a entrenar models amb conjunts de dades molt reduïts.

CAPÍTOL 7

Bibliografia

[1] Jaimovich, D. (2022). *Cuánto tiempo demora un hacker en descubrir tu contraseña.* Recollit de Infobae: https://www.infobae.com/america/tecno/2022/05/06/infografia-cuanto-tiempo-demora-un-hacker-en-descubrir-tu-contrasena/

[2] Prodein. *Tipos de control de acceso.* Recollit de Prodein: https://blog.prodeincendio.com/tipos-sistemas-control-acceso/

[3] Visiotech. *Face-Temp.* Recollit de Visiotechsecurity*:* https://www.visiotechsecurity.com/es/productos/ip-1/temperatura-corporal-fiebre/control-de-accesos-418/face-temp-detail#tab=prod_0

[4] Facefirst. *Breve historia del reconocimento facial.* Recollit de Facefirst: https://www.facefirst.com/blog/breve-historia-del-reconocimiento-facial/

[5] L. Sirovich; M. Kirby. (1987). *Low-dimensional procedure for the characterization of human faces.* Recollit de researchgate: https://www.researchgate.net/publication/19588504_Low-Dimensional_Procedure_for_the_Characterization_of_Human_Faces

[6] Florian Schroff, D. K. (2015). *FaceNet: A Unified Embedding for Face Recognition and Clustering.* Recollit de Arxiv: https://arxiv.org/abs/1503.03832

[7] APD. (2021). *4 tipos de inteligencia artificial que debes conocer.* Recollit de Apd: https://www.apd.es/tipos-de-inteligencia-artificial/#:~:text=Los%20cuatro%20tipos%20de%20inteligencia%20artificial%20que%20debes,sistemas%20que%20puedan%20formar%20representaciones%20sobre%20s%C3%AD%20mismos.

[8] Tyagi, N. (2020). *6 Major Branches of Artificial Intelligence (AI)*. Recollit de Analyticssteps: https://www.analyticssteps.com/blogs/6-major-branches-artificial-intelligence-ai

[9] Pacheco, V. G. (2019). *Una Breve Historia del Machine Learning*. Recollit de Blogthinkbig: https://empresas.blogthinkbig.com/una-breve-historia-del-machine-learning/

[10] Wikipedia (2022). *AlexNet*. Recollit de Wikipedia: https://en.wikipedia.org/wiki/AlexNet

[11] Standford Vision Lab (2010). *Large Scale Visual Recognition Challenge 2021*. Recollit de Image-net: https://image-net.org/challenges/LSVRC/2012/results.html

[12] Murzone, F. (2021). *Funciones de activación para redes neuronales*. Recollit de Medium: https://medium.com/escueladeinteligenciaartificial/funciones-de-activaci%C3%B3n-para-redes-neuronales-de00fefb7150

[13] IBM Cloud Education. (2020). *¿Qué son las redes neuronales?* Recollit de Ibm: https://www.ibm.com/es-es/cloud/learn/neural-networks

[14] Great learning team. (2021). *Types of Neural Networks and Definition of Neural Network*. Recollit de Mygreatlearning: https://www.mygreatlearning.com/blog/types-of-neural-networks/

[15] Calvo, D. (2018). *Función de coste – Redes neuronales*. Recollit de Diegocalvo: https://www.diegocalvo.es/funcion-de-coste-redes-neuronales/

[16] Jordan, J. (2018). *Setting the learning rate of your neural network*. Recollit de Jeremy Jordan: https://www.jeremyjordan.me/nn-learning-rate/

[17] Suniljangir. (2018). *Variants of Gradient Descent*. Recollit de Wordpress: https://suniljangirblog.wordpress.com/2018/12/13/variants-of-gradient-descent/

[18] Bhande, A. (2018). *What is underfitting and overfitting in machine learning and how to deal with it*. Recollit de Medium: https://medium.com/greyatom/what-is-underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6803a989c76

[19] Data science team. (2020). *Regularization*. Recollit de Datascience: https://datascience.eu/machine-learning/regularization-in-machine-learning/

[20] Balawejder, M. (2020). *Optimizers in Machine Learning*. Recollit de Medium: https://medium.com/nerd-for-tech/optimizers-in-machine-learning-f1a9c549f8b4

[21] Qiong Cao (2017) *VGGFace2: A dataset for recognising faces across pose and age*. Recollit de Arxiv: https://arxiv.org/abs/1710.08092

[22] Facebook (2007) *DeepFace*. Recollit de Wikipedia: https://es.wikipedia.org/wiki/DeepFace

[23] Sanchez, J. A. (2020). *datos.Gob.es*. Recollit de Machine Learning y sus diferentes tipos: https://datos.gob.es/es/blog/como-aprenden-las-maquinas-machine-learning-y-sus-diferentes-tipos

[24] Yann LeCun (1989). *Gradient-Based Learning Applied to Document Recognition.* Recollit de standford: http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf

CAPÍTOL 8

Manual d'usuari i/o instal·lació

El codi amb tots els programes necessaris per entrenar, validar i utilitzar el model es poden trobar al següent repositori de GitHub:

github.com/sergif-github/Facenet

Després de recarregar el repositori cal crear un entorn virtual via el programa *Conda*. Dins aquest entorn virtual instal·larem els paquets utilitzats durant el projecte i que alhora es donen.

```
(base) coronis@coronis:~/Desktop/PFG/facenetpytorch$ conda create -n virtualenv python=3.6
Collecting package metadata (current_repodata.json): done
Solving environment: failed with repodata from current_repodata.json, will retry with next repodata source.
Collecting package metadata (repodata.json): done
Solving environment: done
```

```
(base) coronis@coronis:~/Desktop/PFG/facenetpytorch$ conda activate virtualenv
```

```
(virtualenv) coronis@coronis:~/Desktop/PFG/facenetpytorch$ pip install -r requirements.txt
```

Un cop instal·lats els requeriments amb les llibreries necessàries, podem començar a utilitzar tots els programes donats per entrenar, validar i utilitzar un model de reconeixement facial.

El primer que cal realitzar és dividir el dataset en els subconjunts d'entrenament, validació i testatge. El paquet *splitfolders* realitza aquesta divisió automàticament.

```
>>> import splitfolders
>>> splitfolders.ratio(inputfolder, output="dataset",seed=42,ratio=(.5,.20,.3),group_prefix=None)
```

Un cop tenim els datasets, cal extreure'n les dades facials i redimensionar-les a la mida esperada per l'entrada de la xarxa neuronal. El programa *align_dataset_mtcnn.py* ens permetrà utilitzar la GPU per extreure ràpidament els rostres de les imatges amb una mida de 224 x 224 píxels.

```
(facenetpytorch) coronis@coronis:~/Desktop/PFG/facenetpytorch$ for N in {1..4}; do python align_dataset_mtcnn.py /h
ome/coronis/Desktop/PFG/facenetpytorch/datasets/oficina  /home/coronis/Desktop/PFG/facenetpytorch/datasets/oficinaa
ligned --image_size 224 --margin 44 --gpu_memory_fraction 0.25 & done
```

Per tal d'utilitzar els posteriors programes, cal que les imatges d'usuaris estiguin diposades en les seves respectives carpetes dins els conjunts d'entrenament, validació i testatge. Un cop els hi tenim, generem els arxius .csv a través del programa *write_csv_for_making_dataset.py*. Els arxius .csv generats contindran la informació necessària per a la posterior carrega de dades al model de xarxa neuronal.

```
(facenetpytorch) coronis@coronis:~/Desktop/PFG/facenetpytorch$ python write_csv_for_making_dataset.py --root-dir=/h
ome/coronis/Desktop/PFG/facenetpytorch/datasets/oficinaaligned/train --final-file=/home/coronis/Desktop/PFG/facenet
pytorch/datasets/oficinaaligned/train/oficinaaligneddatasettrain.csv
```

Amb les dades facials extretes ja podem començar a utilitzar els diferents programes per entrenar, validar i utilitzar el model de xarxa neuronal.

Per a entrenar un model de xarxa neuronal utilitzarem el programa *train.py*. Aquest programa rep els paràmetres d'entrenament (Batch size, nombre d'epochs, learning rate, learning rate decay, nombre de triplets d'entrenament / validació) i executa l'entrenament del model pre entrenat. Per cada epoch, es guarda automàticament el model entrenat en un format preparat per ser recuperat posteriorment.

Per mostrar els resultats de l'entrenament s'utilitza el programa *plottrainresults.py*. Aquest programa s'encarrega de llegir les dades generades durant l'entrenament per mostrar gràficament la funció de pèrdua obtinguda a cada epoch de l'entrenament pels conjunts d'entrenament i validació. Amb aquesta gràfica, es poden identificar problemes d'overfitting o underfitting. Tanmateix si el conjunt és de dimensionalitat reduïda, les gràfiques no ens permetran identificar aquest problema.

```
(facenetpytorch) coronis@coronis:~/Desktop/PFG/facenetpytorch$ python plottrainresults.py --root-dir /home/coronis/
Desktop/PFG/facenetpytorch/log/modeloriginal
```

Per mostrejar la bona generació dels embeddings del model s'utilitza el programa *embeddingsPlot.py*. Aquest programa rep el model entrenat i el conjunt de testatge a fi de mostrejar els embeddings generats de cada classe existent. Amb això es busca veure com d'aleatòria es la generació dels embeddings per un mateix usuari en cas de no tenir un model perfectament entrenat. La segona gràfica que genera mostra els embeddings mitjans calculats per cada classe a fi de veure com de diferents son entre ells.

```
(facenetpytorch) coronis@coronis:~/Desktop/PFG/facenetpytorch$ python embeddingsPlot.py --pretrain --load-last './l
og/model3/last_checkpoint.pth' --path-dataset /home/coronis/Desktop/PFG/facenetpytorch/datasets/oficinaaligned/test
```

Els testos de cerca del millor llindar de valor fixe per tal d'identificar correctament el major nombre d'usuaris és realitza a través del programa *thresholdtest.py* i *cosinetest.py*. El primer programa cerca el millor llindar de distància euclidiana entre embeddings provant valors d'entre 0 a 15 en salts de 0.5. El segon programa cerca el millor llindar cosinusoïal provant valors d'entre 0 i 1 en salts de 0.05. Ambdós programes reben el model entrenat i el conjunt de testatge i cerquen el millor llindar que permeti identificar correctament el major nombre d'embeddings generats pel model.

Tanmateix cal recordar que s'està utilitzant un llindar fixe útil en un conjunt de dades reduït on els embeddings de cada usuari queden perfectament distanciats en l'espai. La solució però, no serà aplicable a grans conjunts de dades on s'hauria de cercar un altre millor classificador multi classe com SVM.

```
(facenetpytorch) coronis@coronis:~/Desktop/PFG/facenetpytorch$ python thresholdtest.py --test-root-dir /home/coroni
s/Desktop/PFG/facenetpytorch/datasets/oficinaaligned/test --test-csv-name /home/coronis/Desktop/PFG/facenetpytorch/
datasets/oficinaaligned/test/oficinaaligneddatasettest.csv --pretrain --load-last './log/model3/last_checkpoint.pth
' --num-classif 1000
(facenetpytorch) coronis@coronis:~/Desktop/PFG/facenetpytorch$ python cosinetest.py --test-root-dir /home/coronis/D
esktop/PFG/facenetpytorch/datasets/oficinaaligned/test --test-csv-name /home/coronis/Desktop/PFG/facenetpytorch/dat
asets/oficinaaligned/test/oficinaaligneddatasettest.csv --pretrain --load-last './log/model3/last_checkpoint.pth'
--num-classif 1000
```

Per mostrar classificacions de diferents parells d'imatges s'utilitza el programa *classify.py*. Aquest programa classifica 20 parells d'imatges aleatòries del conjunt de testatge donat. Utilitza el model donat, el tipus de llindar (euclidià o cosinusoïal) i el valor del llindar per classificar les imatges. Per cada parell és mostren les imatges, la distància euclidiana entre elles, la predicció i el valor esperat en la predicció. El programa és útil per identificar aquelles imatges en que el model genera prediccions errònies o bé per conèixer la distanciació entre els embeddings generats entre classes diferents.

```
(facenetpytorch) coronis@coronis:~/Desktop/PFG/facenetpytorch$ python classify.py --test-root-dir /home/coronis/Des
ktop/PFG/facenetpytorch/datasets/oficinaaligned/test --test-csv-name /home/coronis/Desktop/PFG/facenetpytorch/datas
ets/oficinaaligned/test/oficinaaligneddatasettest.csv --pretrain  --load-last './log/model3/last_checkpoint.pth' --
threshold 0.55
```

Per validar el model s'utilitzen les matrius de confusió. Aquestes matrius son generades a través del programa *modelAccuracy.py*. El programa rep el conjunt d'entrenament per generar-ne els embeddings representatius, en cas d'utilitzar un llindar euclidià o cosinusoïal, i el conjunt de testatge, que s'utilitza per realitzar i compatibilitzar les bones i males prediccions del model donat. Permet l'ús de diferents classificadors (llindar euclidià o cosinusoïal i SVC) i l'ús d'embeddings representatius mitjans o que minimitzen la distancia a la resta per una mateixa classe (en cas d'utilitzar llindars euclidians o cosinusoïal).
Com a sortida, s'obtenen les diferents matriu individuals de cada classe, la matriu de confusió general i la precisió final del model.

```
(facenetpytorch) coronis@coronis:~/Desktop/PFG/facenetpytorch$ python modelAccuracy.py --pretrain --load-last './lo
g/model3/last_checkpoint.pth' --path-dataset /home/coronis/Desktop/PFG/facenetpytorch/datasets/oficinaaligned/train
 --path-dataset-testing /home/coronis/Desktop/PFG/facenetpytorch/datasets/oficinaaligned/test --use-min --cosineThr
esh 0.55
```

Un cop tenim entrenat i validat el model, aquest pot ser utilitzat en un cas d'us a temps real. Per a fer-ho s'utilitza el programa *realTimeRecognition.py* que rep el model entrenat, el conjunt d'entrenament i el tipus de classificador a utilitzar. A través d'aquest programa s'obté a temps real una classificació.

```
(facenetpytorch) coronis@coronis:~/Desktop/PFG/facenetpytorch$ python realTimeRecognition.py --pretrain --load-last
 './log/model3/last_checkpoint.pth' --path-dataset /home/coronis/Desktop/PFG/facenetpytorch/datasets/oficinaaligned
/train --use-svm
```