Universitat Oberta de Catalunya

uoc.edu

Disseny i implementació d'un sistema d'interfície natural d'usuari basat en càmera de profunditat i aprenentatge profund

Vicent Ortiz Castelló
Grau d'Enginyeria de Tecnologies i Serveis de Telecomunicació
Especialitat en Sistemes de Telecomunicació
Àrea d'Aplicacions Multimèdia Basades en Processament de Senyal

Verónica Vilaplana Besler

Juny de 2022

Aquesta obra està subjecta a una llicència de Reconeixement-NoComercial-SenseObraDerivada 3.0 Espanya de Creative Commons

FITXA DEL TREBALL FINAL

Títol del treball:	*Disseny i implementació d'un sistema d'interfície natural d'usuari basat en càmera de profunditat i aprenentatge profund*
Nom de l'autor:	*Vicent Ortiz Castelló*
Nom de la consultora:	*Verónica Vilaplana Besler*
Nom del PRA:	*José Antonio Morán Moreno*
Data de lliurament:	*06/2022*
Titulació o programa:	*Grau d'Enginyeria de Tecnologies i Serveis de Telecomunicació, especialitat en Sistemes de Telecomunicació*
Àrea del Treball Final:	*Aplicacions Multimèdia Basades en Processament de Senyal*
Idioma del treball:	*Català*
Paraules clau	*Interfície natural d'usuari, natural user interface, càmera de profunditat, depth camera, aprenentatge profund, deep learning*

Resum del Treball:

En la nostra vida quotidiana s'han generalitzat diferents interfícies d'usuari intel·ligents, entre les quals destaquen les que es basen en sistemes tàctils i de veu perquè són versàtils i poc invasives. Per exemple, actualment ja són ubics els telèfons intel·ligents i els assistents virtuals. No obstant això, poques solucions actuals interpreten gestos corporals perquè els sistemes d'altes prestacions sovint són cars, i els més barats, que treballen amb sensors d'espectre visible, depenen fortament de la variabilitat de les condicions de captura a què es veuen sotmesos, especialment d'il·luminació i de distància de treball, i presenten problemes de privacitat. En aquest projecte es duu a terme el disseny i implementació d'una interfície natural d'usuari de baix cost basada en càmera de profunditat Intel RealSense i en models d'aprenentatge profund que s'executen en una plataforma mòbil de computació NVIDIA Jetson Nano, de consum reduït. L'entrenament del model és modular i es pot dur a terme en la mateixa

plataforma mòbil o en un maquinari més potent per a guanyar temps. La solució permet la interacció robusta humà-màquina a distància, en qualsevol condició lumínica –fins i tot en la foscor– i el conjunt de posats detectats es poden personalitzar prenent una seqüència de vídeo de cada nou gest i reentrenant-ne el model. Les possibles aplicacions inclouen la interacció amb diferents tipus d'aparells intel·ligents –ordinadors, televisors, elements de calefacció i domòtica, entre d'altres–, amb un nivell de privacitat elevat, atés que no es registra informació d'espectre visible.

Abstract:

Different intelligent user interfaces have become widespread in our daily lives, including those based on touch and voice systems because they are versatile and non-invasive. For example, smartphones and virtual assistants are now everywhere. However, few current solutions interpret body gestures because high-performance systems are often expensive, and the cheaper ones, which work with visible spectrum sensors, depend heavily on the variability of the capture conditions to which they are subjected, especially to lighting and working distance, and raise privacy concerns. This project designs and implements a low-cost natural user interface based on Intel RealSense depth camera and deep learning models running on an NVIDIA Jetson Nano, a low-consumption mobile computing platform. Model training is modular and may be done on the same mobile platform or on more powerful hardware to save time. The solution allows robust human-machine interaction at a distance, in any light condition –even in the dark– and the set of detected poses can be customised by taking a video sequence of each new gesture and retraining the model. Possible applications include interacting with different types of smart devices –computers, televisions, heating elements, and home automation, among others– with a high level of privacy since no visible spectrum information is recorded.

Índex

Llista d'il·lustracions

1. Introducció

1.1 Context i justificació del Treball

A hores d'ara, en la nostra vida corrent s'ha normalitzat l'ús d'un gran nombre d'aparells tecnològics (ordinadors, tauletes, telèfons intel·ligents, televisors...). Sense pensar-ho, per a interactuar-hi fem servir interfícies d'usuari (en anglés, *human-machine interfaces*). La majoria són ben conegudes i tenen ja molts anys d'ús (per exemple, el comandament a distància via raigs infrarojos). No obstant això, altres són molt més recents, com els comandaments per Bluetooth o les pantalles tàctils.

Darrerament, amb l'adveniment de la realitat virtual i augmentada, les anomenades interfícies naturals d'usuari han esdevingut elements cabdals per la naturalitat que proporcionen en la interacció entre humà i màquina. Per posar-ne un exemple, aparells com l'assistent de veu han revolucionat en pocs anys el món de la domòtica gràcies a la computació en el núvol. A més a més, les ulleres de realitat virtual han permés nivells de realisme i interacció en el món dels videojocs d'alt nivell mai vistos fins aleshores.

Fa aproximadament una dècada, precisament en aquest darrer camp dels videojocs, va aparéixer al mercat un producte innovador i de baix cost que va suposar una revolució en la interacció de l'usuari amb la consola. Es tracta de Kinect, un controlador per a la consola XBOX 360 de Microsoft que permetia el control del transcurs del joc mitjançant la detecció de gestos i comandes de veu. Amb el posterior alliberament del controlador de programari obtingut per enginyeria inversa el 2010, van fer eclosió una miríada de projectes d'ús de la càmera de profunditat que incorpora aquest maquinari (Mejía-Trujillo *et al*., 2019).

No obstant això, aquests projectes tenien alguns problemes remarcables. Primerament, el sensor tenia un camp de visió i un rang de profunditats de treball bastant limitats, i la resolució i el marge d'error de profunditat eren en general pobres perquè es basava en la tecnologia d'escàner de llum estructurada (Intel Corporation, 2019). Tot això limitava molt els àmbits d'aplicació d'aquest dispositiu.

Actualment, hi ha al mercat càmeres de profunditat de baix cost que permeten assolir nivells superiors de qualitat (resolució de profunditat major, camp de visió ample, rang de profunditat estés i error de profunditat reduït). Entre aquestes, destaquen especialment les Intel RealSense de la sèrie 400. Aquestes càmeres es basen en sensors estèreo de llum, i treballen en qualsevol condició lumínica

(fins i tot en la foscor) perquè aprofiten un major rang espectral, incloent-hi la llum visible i infraroja (Intel Corporation, 2019). A més a més, tenen una mida molt reduïda i pesen poc, per la qual cosa són molt fàcils d'emportar.

Al seu torn, i pel que fa a la predicció a partir d'una entrada de dades qualsevol (imatges, dades numèriques, vídeos, llenguatge natural...), l'ús de sistemes experts basats en una extracció de característiques *ad hoc* en mans d'experts ha donat pas a un paradigma d'aprenentatge automàtic (*machine learning*), dins del qual l'aprenentatge profund (*deep learning*) ha rebut molta atenció en nombrosos àmbits del coneixement per la capacitat d'adaptació que té a problemes molt diversos. Actualment, els sistemes automàtics més avançats de la majoria de disciplines fan servir el paradigma d'aprenentatge profund amb xarxes neuronals (Buntine, 2020). Es tracta d'aplicacions tan importants com la detecció de malalties per marcadors, la traducció automàtica, els agents de jocs com ara els escacs o el go, el reconeixement de la parla o la detecció d'objectes en vídeos, per posar-ne alguns exemples.

En aquest treball, per tant, es tracta de combinar l'ús d'un maquinari com aquestes càmeres de profunditat amb un programari basat en un model d'aprenentatge profund per a generar un sistema d'interfície natural d'usuari que permetera, en el seu cas, de controlar qualsevol tipus d'aparell intel·ligent (ordinadors, televisors, domòtica...) mitjançant gestos programables. Addicionalment, atesa la naturalesa del senyal de profunditat, l'ús d'un sistema com aquest generaria poc de rebuig entre els usuaris a causa que no es registra informació de l'espectre visible, per la qual cosa, es tracta d'una solució que respecta la privacitat de l'usuari en alt grau.

1.2 Objectius

Els **objectius generals** del present Treball Final de Grau són els següents:

- Obtindre els fotogrames de profunditat del conjunt de gestos corporals
- Entrenar diversos models d'aprenentatge profund per a detecció de gestos i validar-los
- Implementar el model per a inferència en el sistema d'interfície natural d'usuari

Més concretament, aquests es poden desgranar en els **objectius específics** següents:

- Dur a terme la gravació d'un conjunt de seqüències per cada gest, incloent-hi diferents usuaris per a atorgar robustesa al sistema

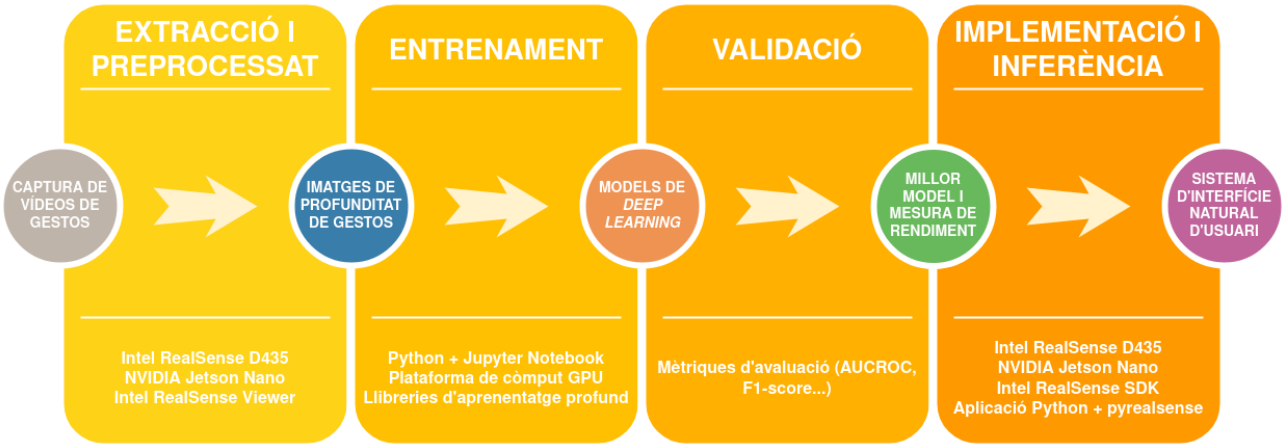
- Extraure els fotogrames de profunditat de cada vídeo i anotar-los segons el gest
- Programar una canonada (*pipeline*) d'entrenament automàtic basat en xarxes neuronals convolucionals d'aprenentatge profund
- Obtindre diversos models de classificació de fotogrames de gestos
- Verificar el rendiment real dels models comparant-los en fotogrames no emprats per a entrenar-los
- Implementar el model en inferència per a l'eventual control de dispositius
- Efectuar una demostració de funcionament del sistema d'interfície natural en temps real

1.3 Enfocament i metodologia

Com s'ha dit més amunt, l'objectiu principal és el desenvolupament d'un sistema d'interfície natural d'usuari basat en gestos corporals. Una possible estratègia hauria estat fer servir càmeres d'espectre visible i algorismes existents basats en aprenentatge profund, com ara OpenPose o AlphaPose (Cao *et al*., 2019). No obstant això, es busca que el sistema capture els gestos de qualsevol persona, per la qual cosa, interessa que l'usuari se senta còmode fent-lo servir. Per aquesta raó, l'ús de càmeres de profunditat, que no registren la informació RGB, pot suposar un avantatge important perquè suposa menys preocupacions per la privacitat de la pròpia imatge.

A més a més, es podria treballar amb sistemes de reconeixement existents per a aconseguir una funcionalitat limitada o treballar amb algorismes que es basaren en una extracció de característiques dissenyada expressament per a la comesa. Tanmateix, es vol generar un sistema modular que puga fer-se servir per a controlar diferents dispositius i que, a més a més, puga incorporar gestos personalitzats de l'usuari. En efecte, un sistema rígid com el que s'ha ataüllat suposaria problemes de baixa generalització en la detecció gestos i dificultat en la generació de noves entrades. Més concretament, per exemple, un canvi de profunditat o d'orientació farien que la detecció fallara, i la inclusió d'un nou gest suposaria una nova fase d'extracció de característiques *ad hoc* que en faria inviable l'ús.

Ben al contrari, gràcies al paradigma d'aprenentatge automàtic, aquestes tasques es poden automatitzar fàcilment i la fase d'extracció de característiques i posterior classificació són transparents per a l'usuari, fets que permeten una adaptació ràpida alhora que robusta a canvis en les condicions de captura o a l'edició del conjunt de gestos a detectar. El procés complet d'aquest enfocament es pot observar en el diagrama de flux de la Il·lustració 1.



Il·lustració 1: diagrama de flux d'alt nivell del procés del projecte

Convé revisar en detall les tasques de baix nivell que s'han de dur a terme per a la consecució del projecte. Primerament, es farà una recerca bibliogràfica profunda en el camp de la interacció humà-màquina i, més concretament, pel que fa als sistemes d'interfície intel·ligent, particularment els d'interfície natural d'usuari, al mateix temps que s'instal·laran les llibreries necessàries per al treball amb les càmeres Intel RealSense de la sèrie 400 en la NVIDIA Jetson Nano i es farà la captura dels gestos corporals d'interés. Posteriorment, aquests arxius es processaran per tal d'obtindre fotogrames de profunditat representatius dels gestos. Tot seguit, es prepararà un entorn d'aprenentatge profund basat en Python, amb Jupyter Notebook i llibreries d'aprenentatge profund com TensorFlow i Keras, i s'obtindran models per a la classificació gestual. Després, se'n farà la validació i comparació amb mètriques d'avaluació ben establides. Per acabar, s'assajarà la inferència en temps real d'aquest sistema gràcies l'embolcall (wrapper) de l'SDK de RealSense per a Python (pyrealsense2). Tot aquest procés de desenvolupament tècnic anirà acompanyat, naturalment, de la redacció en paral·lel dels apartats corresponents de la memòria.

1.4 Planificació i recursos necessaris

Com s'ha descrit detalladament més amunt, els recursos necessaris per al desenvolupament del projecte són els següents:

Maquinari

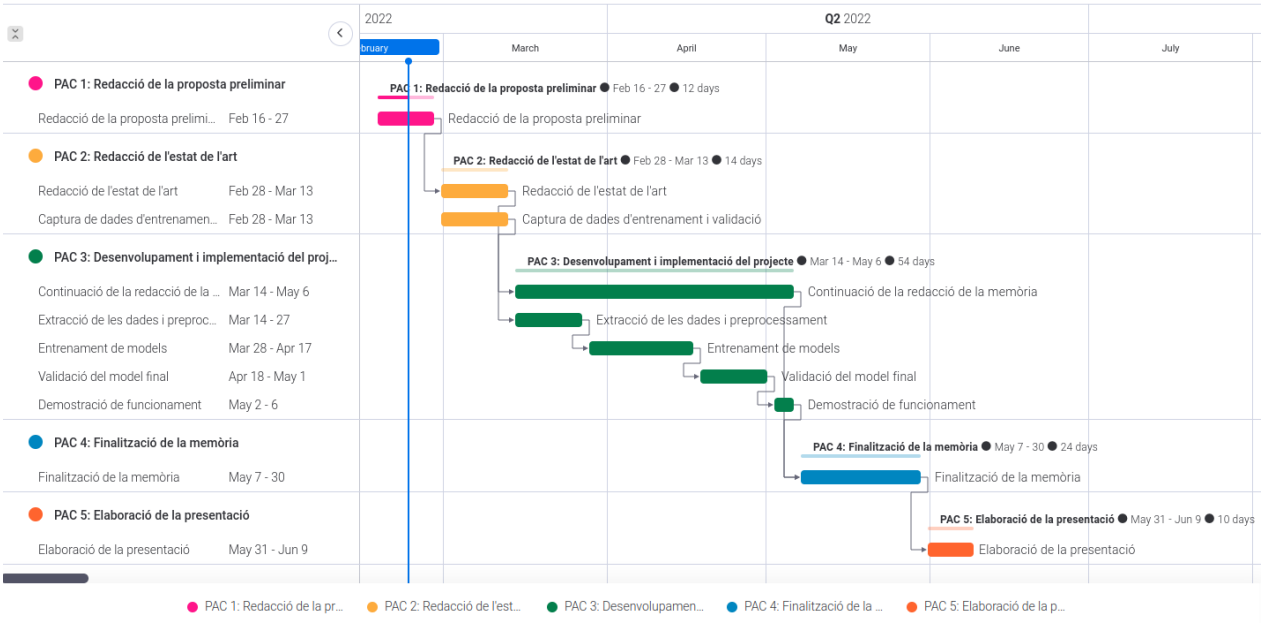
- Intel RealSense D435/D455
- NVIDIA Jetson Nano Development Kit
- Adaptador microUSB 5V 3A
- Cable USB 3.0
- Targeta microSD 128 GB
- Monitor HDMI

- Ordinador portàtil amb accés a GPU
- Ratolí
- Teclat

Programari

- Python 3.10
- Jupyter Notebook
- Intel RealSense SDK 2.0
- Biblioteques estàndard de Python (pyrealsense2, numpy, Keras, TensorFlow, pandas...)

Tot seguit, es presenta el diagrama de Gantt del desenvolupament del projecte en la Il·lustració 2 per a deixar-ne clara la planificació.



Il·lustració 2: diagrama de Gantt del desenvolupament del projecte

Més avall s'expliquen breument les tasques que hi ha al Gantt.

- **PAC 1: Redacció de la proposta preliminar (16/02–27/02)**
  - *Redacció de la proposta preliminar* (16/02–27/02): Redacció dels primers apartats de la memòria.

- **PAC 2: Redacció de l'estat de l'art (28/02–13/03)**
  - *Redacció de l'estat de l'art* (28/02–13/03): Cerca de referències rellevants, actualitzades i pertinents per al projecte, i preparació de l'apartat d'estat de l'art.

  - *Captura de dades d'entrenament i validació* (28/02–13/03): Instal·lació del programari per a les càmeres Intel RealSense D435/D455 (SDK 2.0) en la NVIDIA Jetson Nano i captura de les escenes gestuals necessàries.

- **PAC 3: Desenvolupament i implementació del projecte (14/03–06/05)**
  - *Continuació de la redacció de la memòria* (14/03–06/05): Compleció dels apartats de desenvolupament.
  - *Extracció de les dades i preprocessament* (14/03–27/03): Obtenció dels fotogrames de profunditat i preprocessament.
  - *Entrenament de models* (28/03–17/04): Engegada del procés d'aprenentatge automàtic amb xarxes convolucionals d'aprenentatge profund.
  - *Validació del model final* (18/04–01/05): Comparació entre models mitjançant mètriques estàndard i validació del millor model, que passarà a inferència.
  - *Demostració de funcionament* (02/05–06/05): Inferència en temps real fent servir l'embolcall de Python de l'SDK Intel Realsense 2.0 (pyrealsense2).

- **PAC 4: Finalització de la memòria (07/05–30/05)**
  - *Finalització de la memòria* (07/05–30/05): Compleció de la resta d'apartats i tancament de la versió definitiva.

- **PAC 5: Elaboració de la presentació (31/05–09/06)**
  - *Elaboració de la presentació* (31/05–09/06): Confecció de les diapositives per a la defensa del treball final.

1.5 Breu sumari de productes obtinguts

En acabar el projecte, es preveu haver obtingut un sistema d'interfície natural d'usuari amb personalització de gestos que permeta l'eventual interacció amb màquines. La captura es farà amb una càmera de profunditat lleugera i una plataforma de computació mòbil. Es tracta d'una solució portàtil, senzilla i de baix cost.

1.6 Breu descripció de la resta de capítols de la memòria

A continuació del capítol d'Introducció hi ha el capítol d'Estat de l'art, en el qual es presenten algunes referències necessàries en el camp i se n'analitzen les eines, tècniques i solucions actuals. Això inclou les interfícies naturals d'usuari,

els sensors de profunditat, les tècniques i eines d'aprenentatge automàtic i les opcions de maquinari de computació mòbil existents en el mercat.

Més avall, hi ha el capítol de Desenvolupament, central en la memòria, en el qual es descriu minuciosament el treball tècnic dut a terme en la consecució del projecte. S'hi troba la instal·lació del programari necessari pel que fa tant a la càmera com als models, la captura de les dades, la seua extracció i processament, l'entrenament dels models d'aprenentatge automàtic, la seua comparativa i validació i, finalment, una demostració de funcionament que permeta verificar que el producte obtingut té el comportament que s'espera.

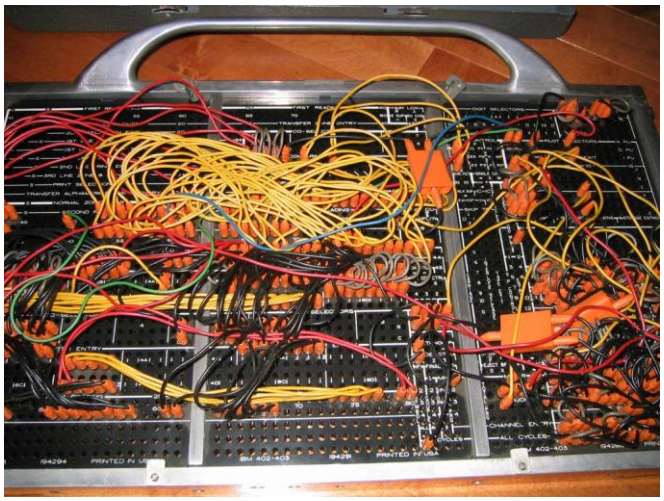
Seguidament, hi ha el capítol de Conclusions, en el qual s'analitza el que s'ha aprés gràcies al desenvolupament del projecte, però també en quina mesura s'han assolit els objectius marcats i fins a quin punt la planificació i la metodologia s'han respectat i han servit. Addicionalment, s'hi inclou un apartat de Treball futur en el qual es tracen possibles línies d'actuació que prenguen com a base el que s'ha aconseguit en aquest projecte.

Per acabar, hi ha els capítols de Glossari (per tal d'explicar els termes i abreviacions més emprats durant la memòria), Bibliografia (que conté les referències en què el projecte es recolza) i Annexos (on es deixa material autocontingut: el codi de programació generat).

2. Estat de l'art

2.1 Interfícies naturals d'usuari

El primer precursor d'aquest tipus d'interfície es pot trobar en els anys posteriors a la Segona Guerra Mundial en les anomenades *Human-Machine Interfaces* (HMI). En aquest moment es van començar a desenvolupar algunes interfícies bàsiques per a comunicar-se amb els rudimentaris ordinadors que hi havia (Myers, 1998). Aquests sistemes eren poc pràctics i s'adaptaven molt poc a l'usuari –de fet, era l'usuari qui s'havia d'adaptar als requeriments de la màquina, atés que es tractava de maximitzar la poca capacitat de càlcul disponible–. Es tractava, per tant, de l'època de les interfícies per lots, en la qual les targetes perforades i els taulers de connexions (vegeu la Il·lustració 3) servien com a entrada per als ordinadors primerencs, i el resultat sovint s'obtenia al cap d'hores o dies.



Il·lustració 3: tauler de connexions de la màquina de comptabilitat IBM 402[1]

Més endavant, a partir dels anys 70, es produí una evolució cap a les anomenades interfícies de línia d'ordres (en anglés, *Command-Line Interface*, CLI) que ja permetia manipular l'ordinador mitjançant la introducció de línies de text cru gràcies a l'ús de teletips primer i terminals de pantalla de vídeo després, tots dos amb teclat per a l'entrada però amb eixida mecànica en paper en el primer i en pantalla en el segon (Myers, 1998). Simultàniament, es van començar a introduir les primeres interfícies gràfiques d'usuari (*Graphical User Interface*, GUI) que feien ús de l'hipertext i de perifèrics com el monitor, el ratolí i el teclat, dispositius que han aplegat fins als nostres dies. En la Il·lustració 4 es poden veure un parell d'exemples paradigmàtics d'aquests dos tipus d'interfícies.

[1] Disponible en https://upload.wikimedia.org/wikipedia/commons/b/b7/IBM402plugboard.Shrigley.wireside.jpg?download

Il·lustració 4: CLI en terminal de vídeo DEC VT100[2] (esquerra) i GUI en Xerox Star 8010[3] (dreta)

Ja en la dècada dels 80, el nou paradigma d'interfície gràfica va permetre el desenvolupament de l'arquitectura de sistema i aplicacions (*System Application Architecture*, SAA), en el qual es basen la majoria de sistemes operatius de MS-DOS ençà, incloent-hi els principals a hores d'ara: Windows, Mac i la majoria de distribucions de Linux. A hores d'ara, no obstant això, els sistemes han evolucionat cap a les interfícies intel·ligents d'usuari (*Intelligent User Interfaces*, IUI), particularment les que són del tipus natural (*Natural User Interfaces*, NUI), amb caràcter general (Hvas, 2020). En la Il·lustració 5 s'aprecia l'evolució natural de les CLI a les GUI i, finalment, a les NUI.

Il·lustració 5: evolució de les interfícies d'usuari: textuals, gràfiques i naturals[4]

Aquestes interfícies naturals permeten a l'usuari comunicar-se amb la màquina de manera intuïtiva i amb accions que formen part del comportament quotidià de l'ésser humà (Hvas, 2020). Alguns exemples amplament estesos en la nostra vida diària són les pantalles tàctils que serveixen d'entrada i eixida de dades en aparells com ara les tauletes o els telèfons intel·ligents, o el reconeixement de la parla que permet de controlar la reproducció de música o elements de domòtica (persianes, il·luminació...) mitjançant un petit assistent amb altaveu i micròfon, connectat a Internet.

[2] Disponible en https://upload.wikimedia.org/wikipedia/commons/9/99/DEC_VT100_terminal.jpg?download
[3] Disponible en https://upload.wikimedia.org/wikipedia/en/1/1d/Xerox_Star_8010_workstations.jpg
[4] Disponible en https://www.interaction-design.org/literature/article/natural-user-interfaces-what-are-they-and-how-do-you-design-user-interfaces-that-feel-natural

Altres d'aquestes tecnologies, en canvi, no tenen encara un nivell d'implantació tan generalitzat. S'hi poden citar, per exemple, els sistemes de reconeixement de gestos, que detecten moviments o postures corporals i els interpreten per a establir una interacció amb la màquina, tant mitjançant anàlisi de dades de moviment (acceleròmetres, giroscopis...) com d'imatge (vegeu la Il·lustració 6), o les anomenades interfícies cervell-ordinador (*Brain-Computer Interface*, BCI), que aspiren a revolucionar la manera de comunicar-nos-hi perquè permeten d'interpretar directament el nostre pensament mitjançant la mesura de les ones cerebrals (Blažica, 2015).

El present treball final s'emmarca dins de la categoria del reconeixement de gestos i pretén oferir a l'usuari un sistema general basat en càmera de profunditat, mínimament invasiu, preparat per a la detecció d'un conjunt provat de gestos, però amb l'opció d'entrenar el sistema amb els gestos personalitzats que li escaiguen, fet que dota el sistema d'una gran adaptabilitat.

Il·lustració 6: exemple de NUI basada en gestos per al control de videojocs[5]

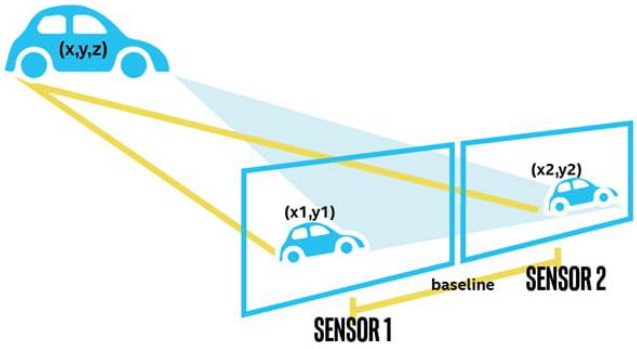
2.2 Sensors de profunditat

Aquests sensors es fan servir per a aconseguir imatges 2D que representen la distància a què es troben els punts de l'escena respecte del punt on es col·loquen. Encara que el resultat que es vol aconseguir és el mateix, el principi físic de funcionament pot ser molt diferent segons la tecnologia emprada en el sensor. Segons això, es pot parlar principalment de mòduls de llum estructurada/codificada, de profunditat estereoscòpics, de temps de vol (*Time of Flight*, ToF) i LiDAR, entre d'altres. Tot seguit, se'n presenten les característiques bàsiques (Intel RealSense, 2019).

[5] Disponible en https://www.interaction-design.org/literature/article/natural-user-interfaces-what-are-they-and-how-do-you-design-user-interfaces-that-feel-natural

Els sensors de llum estructurada/codificada es basen en l'emissió de bandes de llum infraroja que, en projectar-se damunt dels objectes, es deformen. Tot mesurant aquesta deformació i comparant-la amb una imatge de referència es pot aproximar la distància a què es troba cada punt de l'escena. L'ús d'aquests sensors es limita normalment a interiors, atés que són molt sensibles a interferències en la banda infraroja.

En canvi, els mòduls de profunditat estereoscòpics fan servir llum de qualsevol espectre. El muntatge disposa de dos sensors separats, normalment uns quants centímetres, de manera que per simple comparança entre totes dues imatges i coneixent la distància entre sensors es pot calcular la distància a què es troba cada punt per triangulació (vegeu la Il·lustració 7). Aquest és el principi de funcionament en què es basa la visió humana per a aproximar les distàncies als objectes. Els mòduls poden treballar tant dins d'edificis com a l'aire lliure perquè el sistema és robust davant interferències en qualsevol banda espectral.



Il·lustració 7: funcionament bàsic de les càmeres de profunditat estereoscòpiques[6]

Pel que fa als mòduls de temps de vol (ToF) i LiDAR, aquests es basen en l'emissió d'algun tipus de llum i, mitjançant la mesura del temps de retorn dels raigs al sensor i tenint en compte la velocitat de la llum, s'estima el mapa de distàncies a l'escena. Atés que poden fer servir llum de diverses freqüències, permeten mesurar a distàncies considerables, encara que són susceptibles davant la interferència perquè altres raigs poden falsejar les mesures, especialment en l'exterior. Normalment, els ToF es fan servir en espectre visible, mentre que els LiDAR fan servir làsers polsants per a obtindre núvols de punts (De Los Pobres, 2020). Per tant, els primers solen fer-se servir en sensors miniaturitzats amb necessitats més relaxades, com és el cas dels nous telèfons intel·ligents de gama alta en aplicacions de fotografia, mentre que els segons s'instal·len en equips experts amb requisits estrictes, amb el cotxe autònom com a principal exponent (Tillman, 2021).

[6] Disponible en https://www.intelrealsense.com/beginners-guide-to-depth/

Els sensors de profunditat eren en general molt cars fins fa poc més de 10 anys. Es tractava de sistemes experts que, a més a més, habitualment tenien resolucions molt baixes, de l'ordre dels 1.000 a 100.000 píxels, i amb marges d'error amples que en limitaven l'ús en nombroses aplicacions (He *et al*., 2017). Tanmateix, després d'algunes millores tecnològiques clau, a partir del 2005 va créixer l'expectació pel desenvolupament d'alguna d'aquestes càmeres de profunditat, especialment en el context de les plataformes de videojocs. El 2009 es va anunciar el projecte Natal, el qual cercava comercialitzar una solució modular i de baix cost per a la consola XBOX 360 de Microsoft. Finalment, el novembre de 2010 es llançava oficialment Kinect, un controlador de jocs de baix cost que permetia la interacció natural a l'usuari mitjançant mesura de profunditat en QVGA i control per veu (Lowensohn, 2011).

En poc de temps, les vendes de Kinect es van disparar –a tall d'exemple, als EUA es va aplegar al milió d'unitats venudes en menys de dues setmanes– i, al seu torn, va créixer l'interés per a l'ús d'aquest maquinari en aplicacions més enllà del que permetia el sistema de XBOX. Adafruit va oferir, per començar, un premi de 1.000 dòlars a qui generara un controlador per a Kinect mitjançant enginyeria inversa, quantitat que anava creixent amb el temps. En només una setmana, un programador va aconseguir 3.000 dòlars de premi per haver generat aquest programari. A partir d'ací, una infinitud de projectes de codi obert amb Kinect van començar a proliferar, especialment, per al reconeixement de gestos (Biswas i Basu, 2011; Li, 2012; Andersen *et al.*, 2012; Ren, Yuan, Meng i Zhang, 2013).

A pesar que la Kinect disposava d'unes bones prestacions per al moment en què fou creada, el cert és que la resolució limitada, el marge d'error ample i el rang de profunditat de treball estret en limitaven l'ús en nombroses aplicacions, com ara a l'aire lliure. Després, la Kinect 2 va millorar discretament alguns d'aquests registres, com es pot observar en la Taula 1 (Cai, Han, Liu i Shao, 2017). No obstant això, encara hi havia bastant marge de millora en dispositius de baix cost i el 2015, després del succés tan generalitzat de Kinect, Intel va presentar la seua divisió de càmeres de profunditat RealSense i la gama primerenca de càmeres de profunditat associada, amb registres de prestacions mitjanes-altes a baix cost, d'entre 100 i 400 dòlars.

Els anys subsegüents es van desenvolupar nous models de càmera Intel RealSense, entre els quals destaquen els mòduls D435 i D455, les característiques més rellevants dels quals es poden observar comparades amb les de Kinect en la Taula 1 (Intel RealSense, 2021; Intel Corporation, 2022a; Intel Corporation, 2022b). S'hi veu clarament que, amb preus que es mantenen per davall de 400 dòlars, són solucions de més altes prestacions en tots els paràmetres d'interés considerats però en una mida molt més compacta i lleugera –especialment apta per a l'ús en dispositius mòbils i a l'aire lliure–.

Microsoft, al seu torn, va generar un nou model de Kinect el 2018: l'Azure Kinect. Aquest mòdul està orientat a la intel·ligència artificial i fa servir múltiples micròfons, càmeres de profunditat i RGB per a treballar amb models. A canvi d'això, per contra, el mòdul és apreciablement més voluminós i pesat que les RealSense, valors que es mouen en el rang de les Kinect antigues (Tölgyessy, Dekan, Chovanec i Hubinský, 2021). Es pot establir una comparança visual de la mida de les càmeres Kinect 1, Kinect 2, RealSense D435 i Azure Kinect en la Il·lustració 8.

Com ja s'ha dit en seccions anteriors del present treball, per les raons adduïdes en aquesta anàlisi, els models triats per al desenvolupament i consecució dels objectius del projecte són les càmeres RealSense de la sèrie 400, vist que combinen unes característiques avançades amb un disseny modular compacte i lleuger, sense perdre la condició de dispositiu de baix cost.

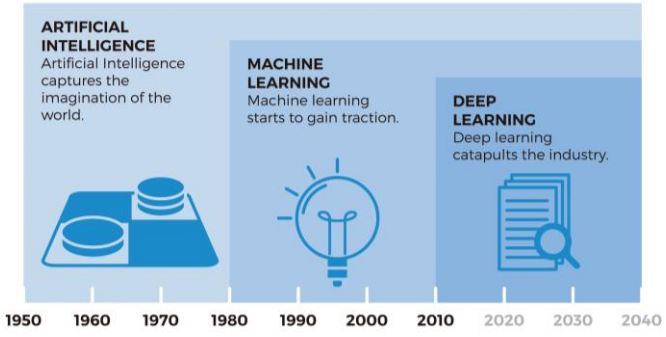
Taula 1: comparativa entre els principals models de càmera de profunditat (Cai, Han, Liu i Shao, 2017; Intel RealSense, 2021; Intel Corporation, 2022a; Intel Corporation, 2022b; Tölgyessy, Dekan, Chovanec i Hubinský, 2021)

Característica / Model	Kinect 1	Kinect 2	Intel RealSense D435	Microsoft Azure Kinect	Intel RealSense D455
Any de llançament	2010	2014	2018	2020	2020
Tecnologia de profunditat	Llum estructurada	Temps de vol (ToF)	Càmera estereoscòpica	Temps de vol (ToF)	Càmera estereoscòpica
Resolució de profunditat (màx.)	320 x 240	512 x 424	1280 x 720	1024 x 1024	1280 x 720
Fotogrames per segon (màx.)	30	30	90	30	90
Rang de distàncies (recomanat, m)	0.4 – 4	0.5 – 4.5	0.3 – 3	0.5 – 3	0.6 – 6
Camp de visió (màx., graus)	57 x 43	70 x 60	87 x 58	120 x 120	87 x 58
Dimensions (mm)	333 x 274 x 79	249 x 66 x 67	90 x 25 x 25	125 x 103 x 39	124 x 29 x 26
Pes (g)	430	610	72	440	84
Estàndard USB	2.0	3.0	3.0	3.1	3.1
Preu de llançament	150 $	200 $	289 $	399 $	389 $

Il·lustració 8: comparança visual entre diferents càmeres de profunditat: se'n destaca la Kinect 1 (darrere a l'esquerra), la Kinect 2 (darrere a la dreta), l'Intel RealSense D435 (davant a la dreta en el trípode) i l'Azure Kinect (davant a l'esquerra al costat del cactus)[7]

2.3 Tècniques i eines d'aprenentatge automàtic

Fa no més d'un parell de dècades, la intel·ligència artificial tenia un èxit limitat a causa que les màquines no aprenien de manera autònoma a partir d'exemples, sinó que els enginyers havien d'extraure les característiques d'interés en el problema de manera particular per en cada cas. Per exemple, si es volien detectar gestos de la mà, s'havien de programar, per exemple, la localització de la mà amb una binarització i la detecció de la zona connexa major de la imatge (Kurakin, Zhang i Liu, 2012). En aquesta aproximació, es podria dir que les màquines no aprenien en què s'havien de fixar, sinó que els humans les ensenyaven a localitzar i detectar determinats segments corporals (Liang i Zheng, 2015). Això no era pràctic ni àgil i, sobretot, no representava un enfocament general que permetera de resoldre problemes diversos.

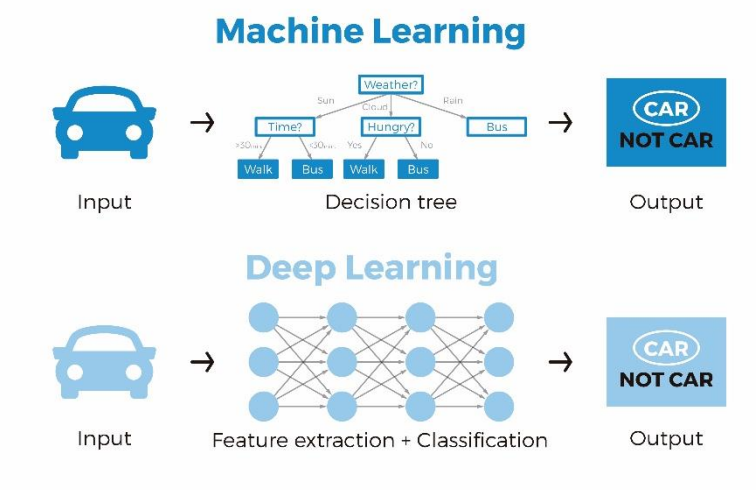


Il·lustració 9: evolució de la intel·ligència artificial en les darreres dècades[8]

En els anys més recents, però, l'aprenentatge automàtic ha representat un avanç sense precedents en el camp de la intel·ligència artificial perquè ha eliminat en

[7] Disponible en https://vvvv.org/blog/revvvvamped-support-for-depth-cameras
[8] Disponible en https://blog.bismart.com/diferencia-machine-learning-deep-learning

gran manera la necessitat d'un disseny específic de l'experimentació per a cada camp. En particular, el paradigma d'aprenentatge profund, que inclou les xarxes neuronals convolucionals i les xarxes recurrents, que prenen el funcionament del cervell com a inspiració, ha dut a la superació de les fites aconseguides en el passat (vegeu la Il·lustració 9) amb algorismes tradicionals en tasques tan diverses com el reconeixement d'objectes, la classificació en imatges o el reconeixement de la parla, entre molts d'altres (LeCun, Bengio i Hinton, 2015).



Il·lustració 10: diferència entre aprenentatge automàtic (ML) i aprenentatge profund (DL)[9]

En el context d'aquest TFG, el fet d'aplicar un paradigma d'aprenentatge automàtic de tipus profund implica no haver de dissenyar específicament un sistema que extraga la semàntica de la imatge de profunditat com ho faria una persona, sinó que, simplement, es deixa que la màquina aprenga en què s'ha de fixar en les imatges de profunditat per tal de maximitzar l'encert de classificació dels gestos (vegeu la Il·lustració 10). Un sistema així, basat en una xarxa neuronal convolucional, pot aplegar a superar el rendiment d'un humà en la tasca de reconeixement de gestos, com en moltes altres tasques.

El llenguatge de programació més habitual per a l'entrenament de models d'aprenentatge automàtic és Python i, en particular, les llibreries de més ús actualment en el paradigma d'aprenentatge profund són TensorFlow (amb la interfície de Keras) i PyTorch (Yalçin, 2021). A més a més, una alternativa a l'ús d'un entorn de desenvolupament integrat (IDE), com ara PyCharm, és l'ús d'un quadern de Jupyter, molt ràpid d'instal·lar perquè s'executa damunt del navegador i molt versàtil per al prototipatge ràpid perquè treballa de manera nadiua amb entorns virtuals –virtualenv– en els quals les llibreries d'interés es poden aïllar i independitzar de les d'altres projectes (Soloelectronicos, 2021). D'aquesta manera, es pot fer servir un paquet d'eines potent però simple de gestionar i, sobretot, totalment compatible i validat.

[9] Disponible en https://blog.bismart.com/diferencia-machine-learning-deep-learning

2.4 Maquinari de computació: *cloud* vs *edge*

Els primers ordinadors eren extremadament voluminosos i pesants, i ocupaven habitacions senceres, per la qual cosa, la computació estava lligada al maquinari. Després, cap als 80, l'adveniment dels ordinadors personals tenia com a missió principal deslligar l'activitat de cada usuari d'una unitat central de computació (Hayes, 2008). Aquesta translació va suposar el primer canvi de paradigma pel que fa al lloc on es duu a terme la computació.

Fins fa aproximadament 30 anys, els dispositius de computació personals eren generalment fixos, voluminosos i de prestacions baixes. A mesura que van anar passant els anys, els avanços tecnològics lligats a l'augment de capacitat de càlcul, a la miniaturització, a la reducció del consum energètic i a la millora de les bateries van conduir a la generalització de les tecnologies mòbils (ordinadors portàtils, agendes electròniques, telèfons...). Sense menyscapte d'això, a pesar que la velocitat de xarxa era encara modesta, la tendència anava en la direcció d'incrementar la potència de càlcul dels dispositius mòbils (millors processadors, busos de dades, memòries...) per tal d'oferir unes prestacions superiors.

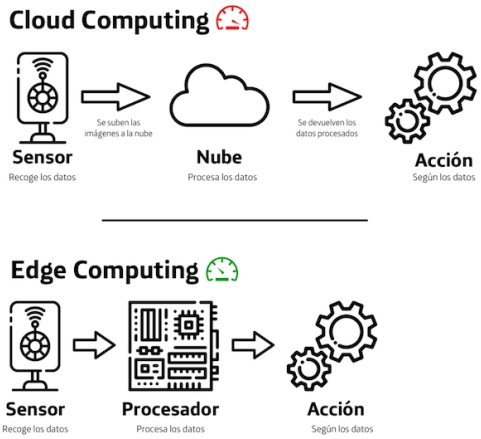
No obstant això, a principi dels 2000, l'augment de velocitat de xarxa (ADSL, fibra òptica, 3G...) permeté l'aparició de nombroses plataformes en línia (YouTube, Facebook, aplicacions de navegació...) que tornaren a basar-se en una unitat central que duu a terme les tasques pesades de computació i memòria (Euskaltel, 2015). Es tracta del paradigma de computació en núvol (*cloud computing*): començava a ser més important disposar d'una connexió d'amplada de banda elevada i latència reduïda que d'un maquinari molt potent en termes de processador i memòria.

Aquest paradigma va permetre de simplificar els dispositius finals, tot abaratint-los: la intel·ligència estava centralitzada al núvol, i s'hi accedia per Internet. L'efecte es va magnificar amb el progrés de les xarxes d'Internet: cablades (fibra òptica) i sense fil (4G). Es tracta del segon canvi de paradigma respecte de la ubicació de la càrrega computacional (Hayes, 2008).

Els darrers anys, però, per bé que la implementació de 5G ha donat encara més ales a la computació en núvol, s'ha anat fent evident que en determinades aplicacions crítiques sovint no es pot confiar en la xarxa o en un cert servei en núvol. L'enfocament operatiu en cas de fallada (*fail-operational*) requereix que el funcionament de determinats sistemes clau es base en càlculs efectuats en plataformes de computació físicament presents al lloc. Això és especialment evident en el cas dels vehicles autònoms (aeronaus, cotxes, camions...) perquè aquests puguen assolir nivells superiors d'automatització (Macher, Druml, Veledar i Reckenzaun, 2019), encara que també pot tindre connotacions de seguretat i integritat de les dades (Shi, Cao, Zhang, Li i Xu, 2016).

Aquest enfocament es coneix com a computació en la vora (*edge computing*). Tornen a guanyar importància la potència de càlcul dels dispositius, la velocitat dels busos de comunicació i la quantitat de memòria per a reduir el temps de resposta i dotar el sistema de robustesa, encara que això supose encarir els dispositius finals (Satyanarayanan, 2017).

Es pot dir que s'està produint un tercer canvi de paradigma, com s'observa en la Il·lustració 11, encara que no de manera generalitzada, atés que la computació en núvol continua de ser la més estesa en aplicacions no crítiques, sovint en combinació amb la computació en la vora. Algunes tecnologies incipients es basen en aquesta amalgama: la internet de les coses (*internet of things*, IoT) o la indústria 4.0 en són bons exemples (Red Hat, 2021).



Il·lustració 11: paradigma *cloud* vs paradigma *edge*[10]

En aquest enfocament, particularment en aplicacions d'intel·ligència artificial tolerants a fallades, és molt recomanable que els equips disposen d'una capacitat de càlcul elevada. A més a més, el fet d'haver d'acostar-les a la font de dades recomana l'ús de plataformes mòbils lleugeres, de mida reduïda i de baix cost. Després de la generalització de la programació i l'ús de microcontroladors per a sensorització i robòtica bàsica gràcies al món Arduino i la dels ordinadors de butxaca per a servidors web i robòtica més avançada amb Raspberry Pi al capdavant, NVIDIA va fer un pas endavant per a proporcionar ordinadors molt compactes i pensats per al món mòbil que permeten una computació més extensiva, requerida per les noves tècniques d'intel·ligència artificial. Aquesta línia de mòduls de computació s'anomena NVIDIA Jetson (NVIDIA, 2022) i les prioritats són un augment de la capacitat de càlcul en diversos ordres de magnitud (gràcies a l'ús de diverses CPU i GPU) però mantenint un consum energètic, una mida i un pes reduïts.

[10] Disponible en https://empresas.blogthinkbig.com/edge-computing-que-es/

Dins d'aquesta gamma de productes es poden citar, per ordre creixent de prestacions i preu, la Jetson Nano, la Jetson TX2, la Jetson Xavier NX i la Jetson AGX Xavier. En la Il·lustració 12 se'n mostra la gama en els respectius mòduls de desenvolupament (*developer kit*), i en la Taula 2 se'n pot apreciar la comparativa (Aufranc, 2020).

Il·lustració 12: mòduls de la gamma NVIDIA Jetson: Nano, TX2, Xavier NX i AGX Xavier[11]

Pel fet que les necessitats que es tenen són principalment d'inferència en aprenentatge profund i es podran admetre latències de centenars de mil·lisegons, en aquest projecte es treballa amb la NVIDIA Jetson Nano, que ja disposa de capacitat GPU per a computació extensiva i treball amb el mòdul RealSense. Aquest dispositiu ja permetrà la consecució dels objectius del projecte.

Taula 2: comparativa entre els principals models de la gamma NVIDIA Jetson (Aufranc, 2020)

Característica / Model	NVIDIA Jetson Nano	NVIDIA Jetson TX2	NVIDIA Jetson Xavier NX	NVIDIA Jetson AGX Xavier
Any de llançament	2019	2017	2018	2018
CPU	Processador de 4 nuclis ARM A57	Processador de 6 nuclis: 2 NVIDIA Denver 2 64-Bit i 4 ARM Cortex-A57	Processador de 6 nuclis NVIDIA Carmel ARM v8.2 64-bit, 6 MB L2 + 4 MB L3 cau	Processador de 8 nuclis ARM v8.2 64-Bit, 8 MB L2 + 4 MB L3 cau
GPU	Maxwell GPU de 128 nuclis	Pascal GPU de 256 nuclis	Volta GPU de 384 nuclis de CUDA i 48 nuclis de Tensor	Volta GPU de 512 nuclis de Tensor

[11] Disponible en https://www.cnx-software.com/2020/05/16/nvidia-jetson-developer-kits-comparison-nano-vs-tx2-vs-xavier-nx-vs-agx-xavier/

Característica / Model	NVIDIA Jetson Nano	NVIDIA Jetson TX2	NVIDIA Jetson Xavier NX	NVIDIA Jetson AGX Xavier
Acceleradors d'IA	No en té	No en té	2 Motors NVDLA i processador VLIW de 7 vies per a visió artificial	2 Motors NVDLA i processador VLIW de 7 vies per a visió artificial
Rendiment de càlcul	0,5 TFLOPS (FP16)	1,3 TFLOPS (FP16)	6 TFLOPS (FP16) o 21 TOPS (INT8)	5,5 – 11 TFLOPS (FP16) o 20 – 32 TOPS (INT8)
Memòria	4 GB 64-bit LPDDR4 a 25.6 GB/s	8GB 128-bit LPDDR4 a 59.7 GB/s	8 GB 128-bit LPDDR4x a 51.2 GB/s	32 GB 256-Bit LPDDR4x a 137 GB/s
Consum de potència	5 – 10 W	7,5 – 15 W	10 – 15 W	10 – 30 W
Dispositiu de ventilació	Dissipador	Dissipador + ventilador	Dissipador + ventilador	Dissipador + ventilador
Mida (mm)	100 x 80 x 26	170 x 170 x 16	103 x 91 x 31	105 x 105 x 55
Preu de llançament	99 $	599 $	1299 $	1499 $

2.5 Solucions i sistemes de reconeixement de gestos existents

Encara que el producte final que es vol obtindre en el present treball es pot considerar innovador pel fet de treballar exclusivament amb càmeres de profunditat, tot seguit se citen una sèrie de solucions de reconeixement de gestos existents que, si bé no són equivalents, sí que tenen alguns elements comuns, com ara un sistema de detecció basat en aprenentatge automàtic o una plataforma de còmput de baix cost.

Alguns treballs es basen en l'ús de la informació RGB i d'algorismes d'aprenentatge automàtic per a la detecció. Per exemple, Francke, Ruiz-del-Solar i Verschae (2007) desenvoluparen un sistema de comunicació humà-robot basat en vídeo RGB que es basava en modelar la pell per tal de segmentar les mans, fer-ne seguiment i decidir-ne la posició gràcies a un conjunt de classificadors tous (*soft*) combinats en cascada (*boosting*) i a un paradigma d'aprenentatge actiu per a reduir la càrrega d'intervenció humana en el procés d'entrenament. No obstant això, aquest sistema té un disseny *ad hoc* i, en conseqüència, la capacitat d'adaptació i generalització que mostra és molt limitada.

Al seu torn, el treball de Dardas i Georganas (2011) està orientat a la interacció humà-màquina en l'àmbit dels videojocs. El sistema està entrenat amb SIFT (transformació de característiques invariant en escala) i es basa en la detecció

d'uns punts estratègics de la mà i l'aplanament de les característiques en un vector unidimensional (llista de paraules). Després d'un agrupament per *k-means*, se'n fa el càlcul d'histogrames, els quals es prenen com a conjunt de característiques per a classificar-se amb un SVM (màquina de vectors de suport). Els autors hi aconseguiren una precisió de detecció superior al 96 % en sis gestos. Novament, el sistema té un disseny específic per a l'extracció i classificació, de manera que tampoc són directes l'adaptació a nous gestos o la generalització a diferents condicions de captura.

Poc després, Dardas i Petriu (2011) desenvoluparen una variant d'aquest sistema que mitjançant l'ús de PCA únicament (anàlisi de components principals) aconseguia vora un 93 % de precisió en la detecció de quatre gestos, si bé amb un temps d'inferència bastant superior. A pesar de la pèrdua de rendiment, s'observa que aquest sistema ja compta amb una versatilitat major a l'hora d'adaptar-se al canvi de condicions a causa de la simplicitat de la proposta.

En canvi, un nombre reduït de treballs fan servir informació de profunditat, encara que solen necessitar també vídeo RGB en un plantejament específic per a la detecció de punts. Reyes, Domínguez i Escalera (2011) usen un enfocament dinàmic: el mapa de distàncies de les articulacions detectades permet fer una extracció de característiques que, mitjançant DTW (*dynamic time warping*) per a treballar amb diferents velocitats i durades, constitueix un sistema de detecció de gestos en vídeos.

De manera similar, Neverova, Wolf, Paci *et al.* (2013) usen dades RGB, de profunditat i fins i tot de la parla per tal de detectar gestos de manera fiable. El desenvolupament es basa en l'aprenentatge automàtic en múltiples escales: separar el moviment a gran escala del cos sencer dels moviments subtils de la mà i, de manera similar, tractar els gestos dinàmics com un seguit de postures corporals. La detecció en aquest treball ja es basa en xarxes neuronals recurrents, per la qual cosa la capacitat de generalització és millor que en propostes anteriors. Després, els autors refinaren el sistema i aconseguiren el millor resultat del moment en una competició de detecció de gestos prenent com a informació les dades RGB i de profunditat (Neverova, Wolf, Taylor i Nebout, 2013).

Més recentment, Köpüklü, Gunduz, Kose i Rigoll (2019) aplicaren un paradigma d'aprenentatge profund a dades tant RGB com de profunditat, en experiments separats. La canonada de detecció es fa en dos fases: una xarxa lleugera CNN per a detectar ràpidament l'instant inicial del gest i una altra xarxa CNN més pesant per a classificar el tipus de gest en qüestió. Gràcies a aquest enfocament, els autors aconseguiren un nivell superior de rendiment de detecció. Especialment, aquests apunten que en sistemes d'aprenentatge profund, les dades de profunditat són més informatives que les d'espectre visible (RGB). A

pesar que aquest sistema és el que més s'assembla al que es desenvolupa en el present treball, ells fan servir una GPU d'alt rendiment (NVIDIA TITAN XP) tant per a entrenament com per a inferència.

Altres treballs se centren a perfeccionar el reconeixement de la mà per a gestos de detall (Song, Chen and Wang, 2020; Brock, Sabanovic, Nakamura i Gomez, 2020; Benitez-Garcia, Olivares-Mercado, Sanchez-Perez i Yanai, 2021; Nair i Noel, 2021).

En canvi, el desenvolupament que es duu a terme en aquest TFG té com a objectiu aconseguir la millor detecció de gestos generals possible fent servir només dades de profunditat per a garantir la privacitat i permetent que tot el procés (entrenament, inferència, reentrenament...) s'execute en una plataforma de baix cost i baix consum energètic que no té la capacitat de còmput d'una targeta gràfica d'alt rendiment com ara les NVIDIA de segment superior (GeForce, TITAN, V100, RTX, Tesla K80...).

3. Desenvolupament

3.1 Instal·lació del programari

Primerament, cal instal·lar tot el programari que es farà servir per al desenvolupament. Pel que fa a la NVIDIA Jetson, principalment, caldrà instal·lar-hi el sistema operatiu Ubuntu 18.04, el llenguatge Python, l'SDK d'Intel RealSense, el seu l'embolcall de Python (pyrealsense2) i les llibreries necessàries per a l'entrenament (Jupyter Notebook, TensorFlow, Keras, pandas i Matplotlib, fonamentalment, que s'encapsularan en un entorn virtual de Python per a aïllar-les de les dependències d'altres projectes). Alternativament, aquest programari s'instal·larà en una màquina Ubuntu 20.04 amb major capacitat computacional per a donar suport a l'experimentació. Tot seguit, es donen les claus principals per a aquesta instal·lació.

En primer lloc, per a instal·lar Ubuntu en la Jetson, es farà servir el tutorial oficial que es proporciona al web de NVIDIA[12] i que consisteix en un mètode habitual en aquestes plaques mòbils: flaixar la targeta amb una imatge del sistema operatiu. Una vegada fet, només caldrà inserir-la en el kit de desenvolupament mòbil Jetson i ja es podrà treballar directament amb la placa com un ordinador qualsevol (amb monitor HDMI i teclat i ratolí USB).

Tot seguit, s'hi fa la instal·lació de Python i de TensorFlow seguint les instruccions que es troben al web de documentació de NVIDIA[13] i, quan se n'ha verificat el funcionament, s'hi instal·la la resta de paquets de manera extremadament simple gràcies a pip:

```
sudo pip3 install jupyterlab keras pandas matplotlib
```

Una vegada s'ha completat el procés, només resta compilar i instal·lar l'SDK d'Intel RealSense seguint aquest tutorial especialitzat de JetsonHacks[14] i, quan el procés acaba, s'hi instal·la l'embolcall de Python novament mitjançant pip:

```
sudo pip3 install pyrealsense2
```

Amb la placa ja preparada, es repliquen les passes d'instal·lació per a la màquina Ubuntu 20.04, si bé algunes són més directes perquè, per exemple, no és estrictament necessari compilar l'SDK. Una vegada preparada aquesta màquina de suport, ja es té tot el sistema preparat per a la gravació de gestos, l'extracció

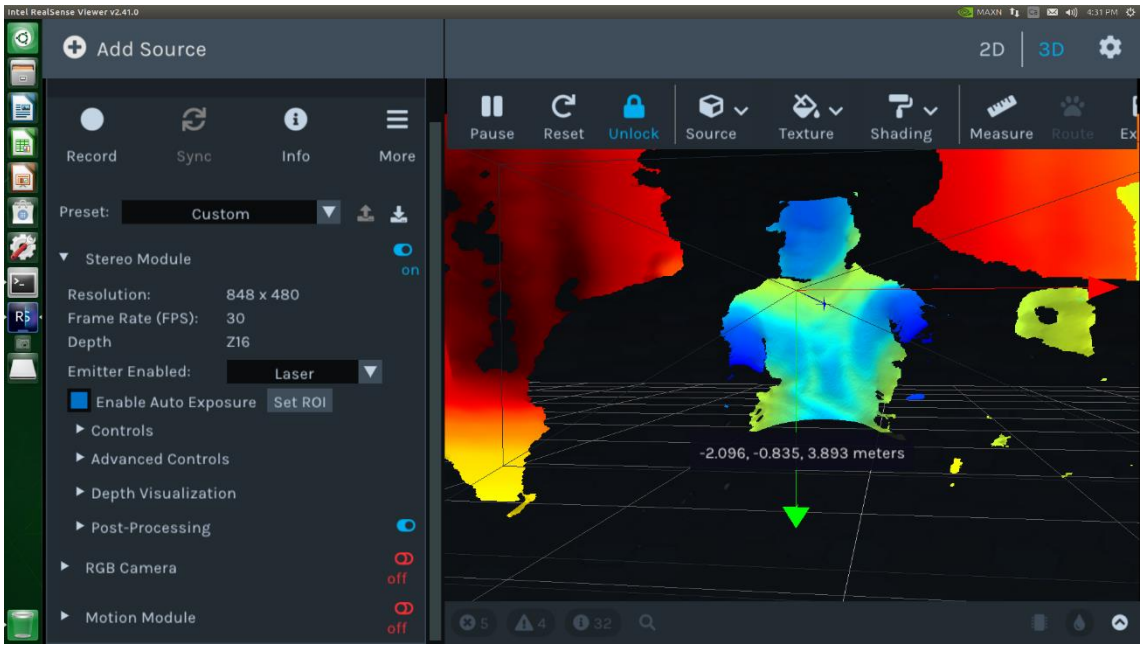
[12] Vegeu https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit#write
[13] Vegeu https://docs.nvidia.com/deeplearning/frameworks/install-tf-jetson-platform/index.html
[14] Vegeu https://github.com/jetsonhacks/installRealSenseSDK

de característiques, l'entrenament i validació de models, i la posada en producció del model final per a inferència en temps real.

3.2 Captura de dades de profunditat

Per a fer-ne la captura, es farà servir l'Intel RealSense Viewer que s'instal·la en companyia de l'SDK de la càmera de profunditat. Aquesta eina d'interfície gràfica d'usuari (GUI) permet dur a terme la gravació dels gestos en format vídeo, en el qual cada fotograma representa un mapa de profunditat 2D de l'escena capturada. Gràcies a aquesta forma de fer, es podran gravar ràpidament un conjunt ample de fotogrames que representen el mateix gest, amb modificacions que semànticament no l'afecten (moviments a esquerra i dreta, avant i arrere o girs lleus davant la càmera...).

En la Il·lustració 13 es mostra un exemple de visualització en temps real que ofereix aquesta eina. Com s'hi pot observar, una vegada generada una certa configuració pel que fa a paràmetres de captura (resolució, nombre de fotogrames per segon, tipus de càmera de captura...), aquesta es pot desar en format JSON i carregar de manera fàcil posteriorment per a agilitar les gravacions.



Il·lustració 13: exemple d'ús de la GUI Intel RealSense Viewer en el mòdul de NVIDIA Jetson Nano per a la captura de fotogrames de profunditat

Tot seguit, es presenten els gestos que es volen detectar prenent la informació de mapa de profunditat de la càmera RealSense. Es tracta d'un conjunt estàtic i de cos complet de vuit gestos més un de neutre, com s'observa a la Taula 3. Cal destacar que aquestes imatges monocromàtiques en escala de grisos d'exemple

s'han generat a partir dels vídeos de profunditat segons el que s'explicarà en el subapartat següent, d'extracció i preprocessament de dades.

Com s'ha vist més amunt, el muntatge necessari per a la captura es compon simplement del mòdul Jetson Nano, un monitor HDMI, un teclat i un ratolí USB i la càmera RealSense connectada també a un port USB 3.0. Se'n mostra una fotografia en la Il·lustració 14. Després de cada vídeo de profunditat gravat, es genera un fitxer binari `.bag` en la carpeta seleccionada que conté totes les dades en cru (punts de profunditat, marques de temps, paràmetres de configuració emprats, etcètera). A partir d'aquests fitxers, es confegirà la base de dades del treball amb fotogrames individuals de profunditat 2D per a entrenar i validar els models de classificació basats en aprenentatge profund.

Taula 3: gestos considerats en el sistema de detecció i imatges de profunditat associades d'exemple

Gest	Imatge de profunditat d'exemple
Posició estàndard en absència de gest (G0)	
Un braç alçat (G1)	
Dos braços alçats (G2)	

Gest	Imatge de profunditat d'exemple
Un braç estés (G3)	
Dos braços estesos (G4)	
Un braç al pit (G5)	
Dos braços al pit (G6)	
Un braç en gerra (G7)	

Gest	Imatge de profunditat d'exemple
Dos braços en gerra (G8)	

De cada gest es captura un vídeo d'uns 20-30 segons per sessió, i es fan un total de quatre sessions en què intervenen quatre usuaris diferents per a assegurar la generalització del model a qualsevol persona. En el subapartat següent es presentaran la lògica i el codi necessaris per a automatitzar l'extracció de les dades en cru als fotogrames en format d'imatge classificats per gest que alimentaran els models d'aprenentatge automàtic.

Il·lustració 14: muntatge necessari per a la gravació de gestos (mòdul de desenvolupament NVIDIA Jetson Nano, monitor HDMI, teclat i ratolí USB i càmera Intel RealSense D455 USB 3.0 en suport elevat)

3.3 Extracció i processament de dades

Una vegada han estat obtinguts els fitxers `.bag`, diversos per a cada gest per a garantir la generalització en diferents condicions de captura, se n'extrauran els fotogrames associats. Com s'ha tingut en compte a l'hora de fer-ne la gravació,

per cada vídeo de profunditat, tots els fotogrames representen el mateix gest. Aquest procés s'automatitza per a guanyar temps i pensant en el fet que calguera ampliar la base de dades o incloure gestos diferents en el futur.

L'extracció fa servir eines de l'SDK d'Intel RealSense, principalment l'ordre `convert`, que permet obtindre fitxers 3D (per exemple, en STL, per a edició i impressió 3D) o 2D (PNG o BMP en color real per a la càmera d'espectre visible o acolorida segons el camp de profunditat, per exemple). En el cas actual, tanmateix, per tal d'optimitzar l'extracció de la imatge de distància en temps i en espai d'emmagatzematge, s'ha preferit fer servir l'eixida 2D en CSV, de manera que cada fotograma consisteix en un fitxer de text separat per comes amb els valors que representen la distància de cada punt projectat al sensor.

La conversió que es cerca s'automatitza fàcilment amb aquest fitxer *shell* que executa ordres de terminal.

```
for file in *.bag
do
   file_no_ext=${file%.*}
   gesture=${file_no_ext: -2}
   mkdir ../../$gesture/${file_no_ext::-12}
   rs-convert -i $file -d -v ../../$gesture/${file_no_ext::-12}/
   ls -F ../../$gesture/${file_no_ext::-12}/*.csv | head -n 40 | xargs -r rm
   ls -F ../../$gesture/${file_no_ext::-12}/*.csv | tail -n 40 | xargs -r rm
   cd ../../$gesture/${file_no_ext::-12}/
   find . -name "*.txt" -type f –delete
   cd -
done

find . -name "*.log" -type f -delete
find . -name "*.bag" -type f –delete

mkdir ../../CSV/
mv ../../G* ../../CSV/
```

Si es té en compte que tots els fitxers `.bag` dels diferents gestos es troben en la mateixa carpeta, es pot interpretar el codi. En la primera línia, s'obri un bucle que recorre tots els fitxers binaris i, per cadascun, genera una carpeta amb un nom representatiu de la sessió dins de cada carpeta de gest (de G0 a G8) i hi deixa els fitxers CSV convertits segons el gest. Finalment, atesa la taxa de 15 fotogrames per segon en gravació, s'eliminen vora tres segons de fotogrames al principi i al final de cada vídeo de profunditat per a evitar incloure les primeres i les últimes captures, que corresponen a l'usuari posant en marxa i aturant la gravació, per la qual cosa, és molt probable que en alguns l'usuari no faça el gest que ha de representar eixe conjunt d'instàncies. Per acabar, es netegen els subproductes de la conversió (`.txt` de metadades i `.log` que no ens calen, en principi) i, per raons d'eficiència en l'ús de memòria, també s'eliminen els binaris.

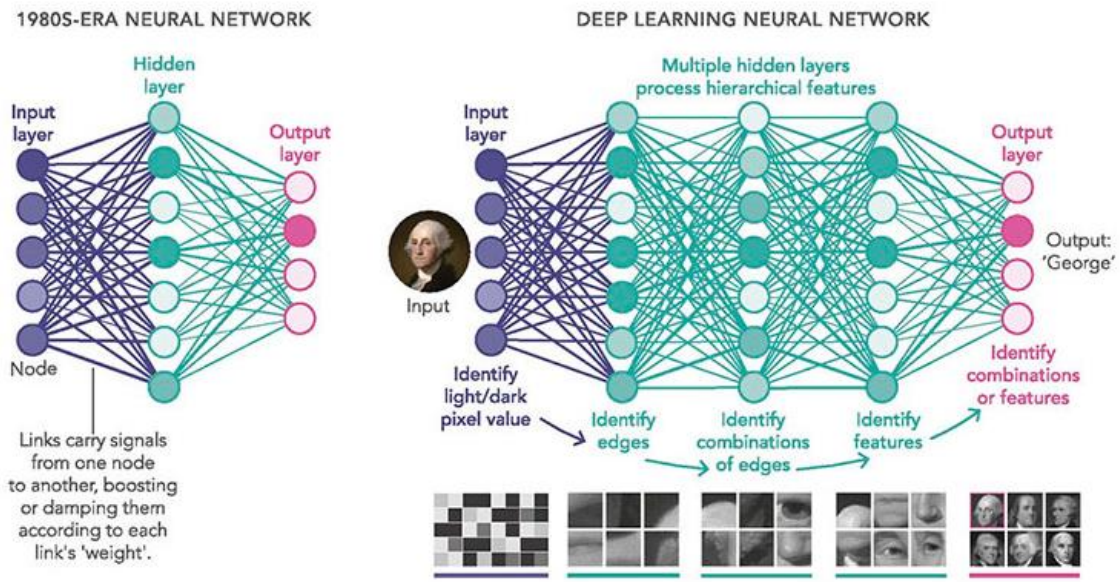
Una vegada es tenen els fitxers CSV individuals, s'executarà un fragment de codi que editarà el nom de tots els fitxers per a unir-los en una mateixa carpeta però de manera que continuen de ser representatius de l'origen i, posteriorment, amb un altre fragment de codi, es dividiran aleatòriament en tres grups (entrenament, validació i test) en un percentatge estàndard (per exemple, 75 % per a entrenament, 15 % per a validació i 10 % per a test) i es mouran les dades a les carpetes corresponents. A banda d'aquestes dades, es destinarà un conjunt de vídeos extra a efectuar una comprovació final en test amb dades completament noves.

Després, amb això fet, es preprocessaran aquests fitxers per tal de generar imatges PNG que permeten l'objectiu triple de tractar-los com a instàncies en la base de dades de gestos per a l'entrenament de models d'aprenentatge automàtic, de poder verificar a simple vista cada exemple i de reduir-ne la mida pel pas a INT8 (una reducció aproximadament en un factor 20: cada exemple passa d'uns 500 KB a uns 25 KB). Tot el codi necessari per a dur a terme aquestes conversions es pot trobar en l'annex del subapartat 7.1. Arribats a aquest punt, es pot començar la canonada (*pipeline*) d'aprenentatge automàtic.

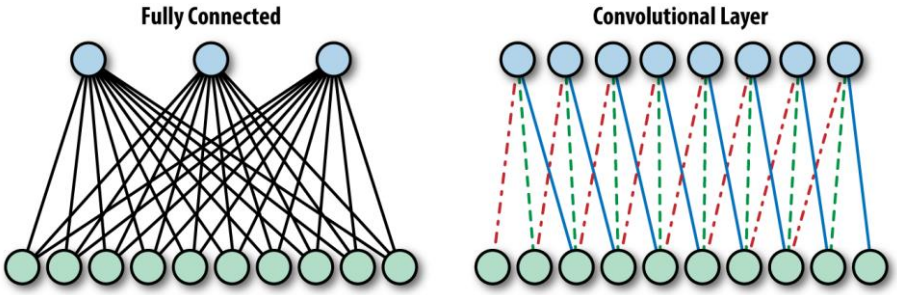
3.4 Entrenament de models d'aprenentatge automàtic

Per tal d'entrenar els models, es farà servir un enfocament típic en aprenentatge automàtic: es crearà un Jupyter Notebook i s'empraran llibreries Python estàndard, com ara PIL per a l'edició d'imatges, TensorFlow (2.8.0) en l'API de Keras per a l'entrenament de models d'aprenentatge profund o Matplotlib per al traçat de gràfics. Tot el codi que s'ha generat per a l'entrenament també està disponible en l'annex del subapartat 7.1. Com correspon a aquestes llibreries, els models d'aprenentatge automàtic que es faran servir seran xarxes neuronals convolucionals profundes, que tenen una capacitat de representació molt elevada i molta facilitat per a resoldre tasques complexes i variades.

En la Il·lustració 15 es pot apreciar l'estructura general que tenen aquestes xarxes profundes. Convé destacar que, a diferència de les xarxes amb capes totalment connectades, en les de tipus convolucional es troben capes les neurones de les quals es connecten només amb les neurones de la perifèria de la capa anterior en cada cas, com s'observa a la Il·lustració 16. Gràcies a això, es conserven la majoria de les prestacions de la xarxa però se'n redueix molt la mida i la càrrega computacional tant en entrenament com en inferència a causa de la reducció de paràmetres perquè cada capa en té un nombre pràcticament proporcional al nombre de neurones d'aquesta capa i de l'anterior.



Il·lustració 15: paradigma de models basats en xarxes neuronals tradicionals amb una capa oculta (esquerra) i les actuals, molt més profundes (dreta)[15]



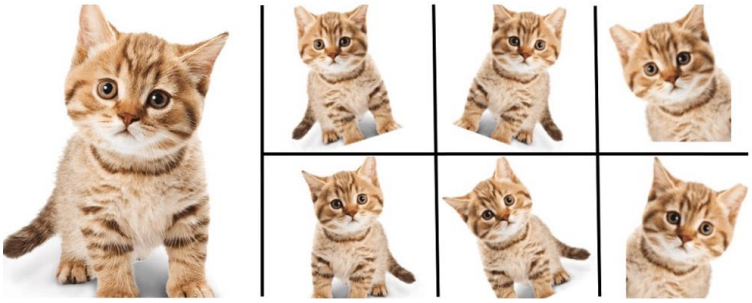
Il·lustració 16: diferència entre les capes totalment connectades (esquerra) i les convolucionals (dreta)[16]

Primerament, es programen els generadors de dades perquè vagen creant-se els lots a mesura que calguen tant en l'entrenament com en la posterior validació. En el primer cas, s'inclou augmentació de dades (*data augmentation*) com a primera tècnica de regularització perquè els models puguen generalitzar millor dades mai vistes. En la Il·lustració 17 es veu un exemple del tipus de modificacions que s'efectuen a les imatges d'entrenament: un cert nivell aleatori de rotació, desplaçament horitzontal/vertical i ampliació, i la possibilitat d'inversió horitzontal.

Cal remarcar que aquests canvis no han de modificar semànticament les imatges d'entrada perquè es falsejarien molts exemples d'algunes classes. Per exemple, si s'ampliara massa o es fes una translació excessiva es podrien perdre parts del cos essencials per al reconeixement del gest. De manera especialment interessant, si s'hagueren previst gestos diferents per al braç esquerre i per al

[15] Disponible en https://medium.com/swlh/an-overview-on-convolutional-neural-networks-ea48e76fb186
[16] Disponible en https://www.futurespace.es/redes-neuronales-y-deep-learning-capitulo-1-preludio/

dret, el fet d'invertir horitzontalment les imatges de manera aleatòria confondria completament la detecció d'aquestes dues categories. Tanmateix, en aquest treball només hi ha gestos invariants a inversió horitzontal, atés que es poden fer indistintament amb qualsevol dels dos braços.



Il·lustració 17: exemples d'augmentació de dades considerats al treball[17]

Passant als models utilitzats, i dins del camp de les xarxes neuronals profundes, segons les dades oficials proporcionades al web de Keras de resultats en ImageNet, mida dels pesos, nombre de paràmetres, nombre de capes i temps d'inferència, se n'ha seleccionat un conjunt que representa de la millor manera possible el conjunt de variants disponibles a hores d'ara i que permet d'executar-se en una plataforma de computació mòbil com la NVIDIA Jetson Nano (Keras, 2022). Aquests models són InceptionResNetV2 (Szegedy, Ioffe, Vanhoucke i Alemi, 2017), Xception (Chollet, 2017) i DenseNet121 (Huang, Liu, Van Der Maaten i Weinberger, 2017). En la Taula 4 se'n poden observar les característiques principals.

Taula 4: característiques principals de les xarxes profundes considerades

Model / Paràmetre	Mida (MB)	Precisió ImageNet (Top-1)	Precisió ImageNet (Top-5)	Nombre de paràmetres (milions)	Nombre de capes	Temps típic d'inferència en CPU / GPU (ms)
Inception ResNetV2	215	80,3 %	95,3 %	55,9	449	130,2 / 10,0
Xception	88	79,0 %	94,5 %	22,9	81	109,4 / 8,1
DenseNet121	33	75 %	92,3 %	8,1	242	77,1 / 5,4

L'enfocament habitual en aprenentatge profund per a cercar la generalització i l'estabilitat i accelerar l'entrenament és usar transferència d'aprenentatge (*transfer learning*) per a disposar d'una inicialització adient i per a aprofitar els pesos de les primeres capes dels models entrenats en ImageNet i entrenar només la resta de capes a continuació. Tanmateix, en el cas actual, havent comparat els resultats d'aquesta aproximació amb les prestacions dels models

[17] Disponible en https://franspg.wordpress.com/2020/01/27/generacion-de-datos-artificiales-data-augmentation/

entrenats des de zero, es prendrà la inicialització dels pesos d'ImageNet però s'entrenaran totes les capes de la xarxa a partir d'aquests.

És convenient parar esment que, a causa del plantejament amb imatges que representen profunditat, aquestes són monocromàtiques, per la qual cosa és possible que tornar a entrenar les primeres capes des de zero afine els resultats. Això ocorre perquè aquesta primera part de les xarxes, dedicada a l'extracció de característiques, en principi no hauria de tindre diferències en els pesos dels tres canals de color, entre d'altres consideracions. Consegüentment, si s'entrena la xarxa completa, aquesta podrà adaptar-se millor a imatges en escala de grisos que representen profunditat i no color, amb diferències semàntiques.

Al final de cada xarxa se substituirà la darrera capa per un conjunt de capes totalment connectades que acabaran en una capa amb un nombre de neurones igual al nombre de gestos a detectar i amb funció d'activació de tipus sigmoide, esquema típic de la classificació multiclasse. La funció de pèrdua que es farà servir serà, consegüentment, l'entropia creuada categòrica (*categorical cross-entropy*). Addicionalment, en el procés d'entrenament s'inclourà alguna mesura de regularització més enllà de l'augmentació de dades, com ara *drop-out*.

Per a optimitzar l'entrenament, s'aniran desant els pesos que donen un millor resultat per a evitar el sobreentrenament. A més a més, per a guanyar temps en l'experimentació, es farà una parada primerenca (*early stopping*) si el model no millora en un nombre determinat d'èpoques. Després, en inferència, només caldrà carregar l'estructura del model i els pesos en format binari.

Una vegada provat el funcionament del codi per a l'entrenament, aquest es deixarà en un fitxer de Python i així es podrà entrenar des de la línia d'ordres gràcies a la llibreria `argparse`. Si es llança l'ajuda hi apareixen els paràmetres que el codi d'entrenament espera en la crida.

```
> python3 TFG_train.py –h

usage: TFG.py [-h] [--model_type MODEL_TYPE] [--not_trainable NOT_TRAINABLE] [--
   epochs EPOCHS] [--learning_rate LEARNING_RATE] [--patience PATIENCE]
```

Els hiperparàmetres són:

BASE_PATH: directori on hi ha les dades d'entrenament
MODEL_TYPE: nom del model segons estructura
NOT_TRAINABLE: proporció de capes amb pesos fixats que no s'entrenen
EPOCHS: nombre màxim de passades completes per les dades d'entrenament
LEARNING_RATE: taxa d'aprenentatge per a l'actualització
PATIENCE: nombre màxim d'èpoques que es continuarà entrenant sense millora

Per tant, es pot començar un entrenament simplement així, on cada paràmetre ha de ser substituït pel seu valor:

```
> python3 TFG_train.py --b BASE_PATH --m MODEL_TYPE --n NOT_TRAINABLE --e EPOCHS --l
   LEARNING_RATE --p PATIENCE
```

El procés d'entrenament es repeteix per a tots els tipus d'estructura de xarxa. S'ha avaluat el conjunt d'hiperparàmetres que millor resultat han oferit, els quals es mostren davall. Així mateix, es valorarà si hi han aparegut problemes en l'entrenament (sobreentrenament o subentrenament, velocitat de convergència...).

NOT_TRAINABLE: 0 (s'entrenen tots els pesos a partir dels d'ImageNet)
EPOCHS: 50
LEARNING_RATE: 2e-5
PATIENCE: 8

Tot seguit, es pot veure un exemple resumit de l'eixida per terminal corresponent a un dels entrenaments de prova.

```
(ml_venv) vortiz@vortiz-desktop:~/gesture$ python3 TFG_train.py –b
    "/home/vortiz/gesture/TFG_PROVES/" --m "Xception" --n 0 --e 50 --l 2e-5 --p 8
Num GPUs Available:  1
Found 10042 images belonging to 9 classes.
Found 2008 images belonging to 9 classes.
Generadors creats correctament
Els pesos precarregats són de imagenet
El nombre total de capes de la xarxa és de 132
Model: "model"
_________________________________________________________________________________
Epoch 1/50
Epoch 00001: val_loss improved from inf to 1.47079, saving model to
   /home/vortiz/gesture/TFG_PROVES_ULTIMATE/MODELS/model_Xception_ultimate.h5
628/628 - 230s - loss: 1.9077 - categorical_accuracy: 0.2857 - val_loss: 1.4708 -
   val_categorical_accuracy: 0.3202 - 230s/epoch - 366ms/step

(...)

Epoch 00036: val_loss did not improve from 0.08942
628/628 - 209s - loss: 0.1383 - categorical_accuracy: 0.9824 - val_loss: 0.1050 -
val_categorical_accuracy: 0.9905 - 209s/epoch - 332ms/step
-----------------------------------------------
Valors que podem mostrar: dict_keys(['loss', 'categorical_accuracy', 'val_loss',
   'val_categorical_accuracy'])
-----------------------------------------------
Estructura i model final desats
```

Per a cada entrenament, es desen les corbes de pèrdua en funció de l'època.

3.5 Metodologia de comparativa i validació de models

En aquest subapartat, una vegada han estat entrenats els models, se'n farà una avaluació. Com s'ha dit en el subapartat d'extracció i processament, en la partició aleatòria de les dades s'ha destinat un 10 % dels fotogrames de profunditat, mai vistos en l'entrenament, a la partició de test. No obstant això, i per tal d'assegurar que el model generalitza correctament, a banda d'aquesta primera prova de rendiment del classificador s'analitzarà una partició de test extra, completament independent de la resta de dades, provinent d'un altre vídeo, en la qual es podria esperar una davallada lleugera en el rendiment de predicció.

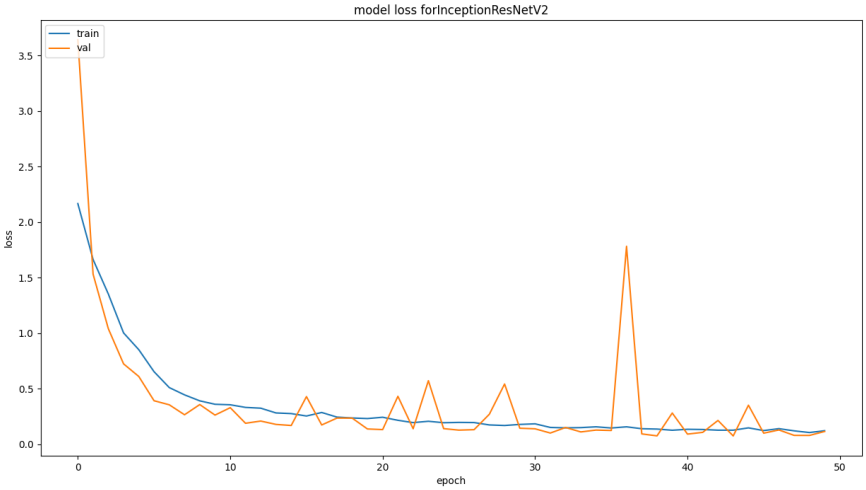
Encara que per a entrenar els models s'ha fet servir l'entropia creuada categòrica com a funció de pèrdua, per tal de valorar quantitativament el rendiment dels models, i donat que les classes són ben equilibrades quant al nombre d'exemples, l'avaluació es basarà principalment en una mesura més fàcilment interpretable, com ara la precisió de classificació multiclasse, que simplement representa el nombre d'exemples de gestos correctament classificats pel model respecte del total d'exemples analitzats.

Atés que la capa d'eixida del model té tantes neurones com gestos, cada exemple generarà un conjunt de nou nombres entre zero i u que donaran una idea de la probabilitat de correspondència a cada gest (decisió tova o *soft*). No obstant això, com el classificador ha de donar una categoria predita, simplement, se'n prendrà el màxim per a binaritzar-ne la decisió (dura o *hard*). Després, es farà servir aquesta predicció més probable per a comparar amb la classe real i verificar la precisió mitjana del classificador, entre d'altres mètriques.

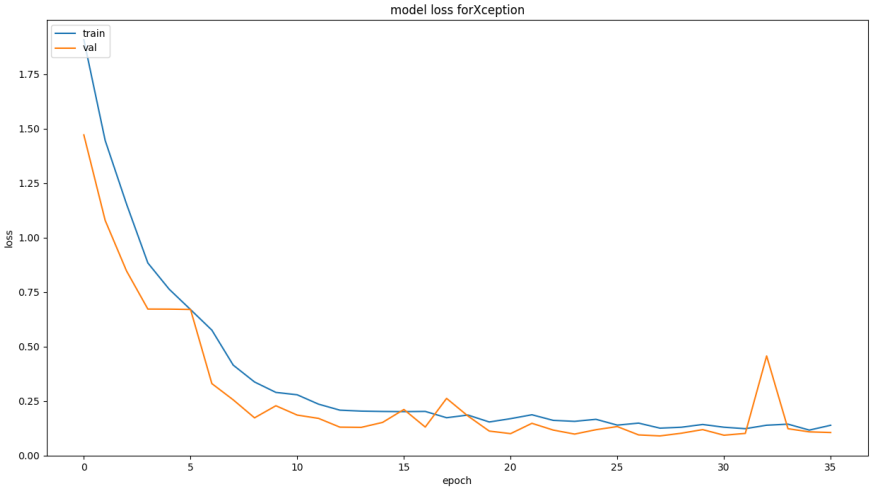
3.6 Descripció dels resultats obtinguts

En l'annex del subapartat 7.1 es pot veure el codi associat a l'avaluació de models. Primerament, es mostren les corbes d'aprenentatge amb els valors de pèrdua per als tres models entrenats (InceptionResNetV2 en la Il·lustració 18, Xception en la Il·lustració 19 i DenseNet121 en la Il·lustració 20).

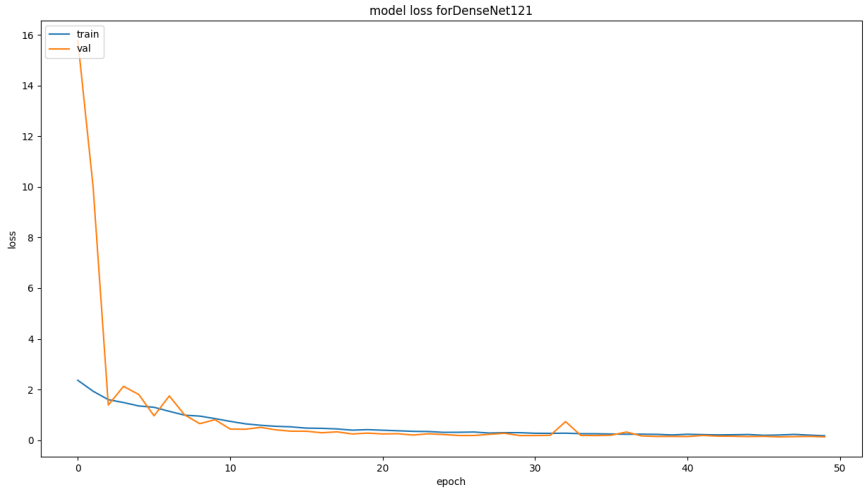
S'observa que l'entrenament convergeix com a màxim en unes 30 èpoques en tots els casos, amb posteriors refinaments mínims. Així mateix, totes les xarxes són capaces d'aprendre de les dades amb precisió atenent al descens profund i continu en la pèrdua d'entrenament, i sembla que tots els models tenen un descens més o menys estable en validació, fet que indica la robustesa de l'entrenament.



Il·lustració 18: corbes d'entrenament amb les funcions de pèrdua en els conjunts d'entrenament i validació del model InceptionResNetV2



Il·lustració 19: corbes d'entrenament amb les funcions de pèrdua en els conjunts d'entrenament i validació del model Xception

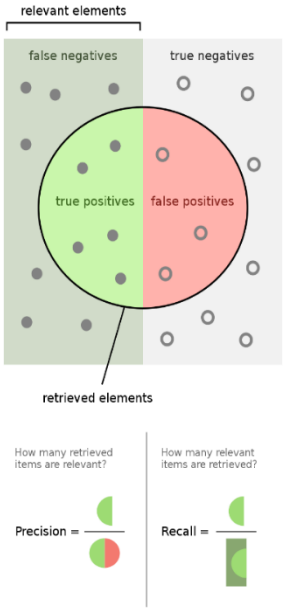


Il·lustració 20: corbes d'entrenament amb les funcions de pèrdua en els conjunts d'entrenament i validació del model DenseNet121

Així mateix, es pot veure que la tendència és a convergir dins del nombre d'èpoques descrit i, com que es desa el millor model en validació, s'evita el sobreentrenament en tots els casos. A més a més, el fet de tindre pràcticament tota la corba de pèrdua en validació per davall de la d'entrenament en tots tres casos i el d'obtindre valors de pèrdua reduïts indiquen alguns aspectes importants, els quals es descriuen a continuació.

Primerament, la relació entre les corbes confirma que hi ha una bona generalització dels models a les dades no vistes en tots tres models (s'evita el sobreentrenament). En segon lloc, el fet que s'aconseguisquen valors menuts de pèrdua en validació i test en tots els casos –valors d'entropia creuada categòrica al voltant de 0,1– permet de descartar la presència de subentrenament. Finalment, l'aparent contradicció que s'aconseguisquen millors valors en dades no vistes que en les d'entrenament es deu a la inclusió del *drop-out* en entrenament, el qual fa perdre una miqueta de rendiment de classificació a la xarxa a costa d'augmentar la generalització (en validació, com en test, el *drop-out* no està present, i es treballa amb la xarxa completa).

Tot seguit, es passa a avaluar el funcionament dels models en el conjunt de test. El primer que cal fer és carregar cada model del fitxer binari on se n'han desat estructura i pesos. Una vegada regenerats els models, es farà servir la funció de predicció de Keras i un parell de funcions d'avaluació de scikit-learn –un resum de mètriques de classificació i el traçat de la matriu de confusió– per a conéixer el rendiment de classificació dels models en detall.



Il·lustració 21: interpretació geomètrica i analítica amb diagrames de Venn de les mètriques precisió i reclam –*recall*– per al cas de classificació binària[18]

[18] Disponible en https://ca.wikipedia.org/wiki/Precisi%C3%B3_i_reclam#/media/Fitxer:Precisionrecall.svg

Abans de presentar els valors obtinguts, però, es farà una breu presentació del significat de les mètriques d'interés. En primer lloc, es pot veure la interpretació gràfica i analítica de la precisió i el reclam –*recall*– per a tasques de classificació binària en la Il·lustració 21. S'hi pot veure clarament que la precisió i el reclam en una tasca binària es defineixen així:

$$precisió = \frac{VP}{VP + FP} \qquad reclam = \frac{VP}{VP + FN}$$

on VP significa vertader positiu (un positiu real que es classifica com a positiu), FP significa fals positiu (un negatiu real que es classifica com a positiu) i FN significa fals negatiu (un positiu real que classifica com a negatiu). La interpretació d'aquestes dues mesures és simple: la precisió informa de la proporció de casos classificats com a positius que són realment positius i el reclam, de la proporció de casos realment positius que són classificats com a positius.

Amb la combinació d'aquestes dues mètriques es calcula l'anomenat valor-F1 (*F1-score*), que no és més que una mitjana harmònica equilibrada entre totes dues –hi atorga la mateixa importància–, com s'observa a continuació.

$$F_1 = 2 \cdot \frac{precisió \cdot reclam}{precisió + reclam}$$

En el cas que es troba en aquest treball, les mètriques són similars a les presentades més amunt però amb una extensió al cas multiclasse. Per això, per cada classe, se'n calcula la precisió i el reclam amb els valors de VP, FP i FN considerant els exemples d'eixa classe com a positius i tots els de les altres com a negatius.

Seguidament, es presenta el valor de precisió global (nombre de casos encertats pel classificador dividit pel nombre total d'instàncies) i els valors de precisió, reclam i F1 en versió *macro avg* (en la qual es fa la mitjana dels valors per classe sense ponderar per nombre d'instàncies) i *weighted avg* (en la qual es fa la mitjana ponderada pel nombre d'instàncies dels valors per classe). En el nostre cas, però, aquestes mètriques seran similars, ja que les classes estan bastant equilibrades en nombre d'exemples. Tota aquesta informació està especificada al web de scikit-learn[19].

Finalment, la matriu de confusió dona una informació més detallada del funcionament del classificador, atés que permet de conéixer no només quants exemples han estat ben classificats per classe i quants han estat confosos pel

[19] Vegeu https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html

classificador, sinó també quina classe els ha atorgat. Així, es podrà saber quins gestos són més semblants o més complexos, per exemple. A més a més, els valors de VP, FP i FN per classe són molt fàcils de visualitzar a la matriu:

- VP: elements de la diagonal
- FP: suma dels elements de la columna excepte el de la diagonal
- FN: suma dels elements de la fila excepte el de la diagonal

Novament, aquesta informació es pot consultar detalladament al web de scikit-learn[20]. Tot seguit, es veuen els valors de les mètriques que s'obtenen per als tres models finals mitjançant l'execució del codi de test per terminal, com s'observa davall.

```
> python3 TFG_test.py --b BASE_PATH --m MODEL_PATH
```

InceptionResNetV2

	precision	recall	f1-score	support
G0	0.9935	0.9872	0.9904	156
G1	0.9865	1.0000	0.9932	146
G2	1.0000	1.0000	1.0000	138
G3	0.9923	1.0000	0.9961	129
G4	1.0000	0.9930	0.9965	142
G5	0.9932	1.0000	0.9966	145
G6	1.0000	0.9940	0.9970	167
G7	0.9938	0.9938	0.9938	162
G8	1.0000	0.9935	0.9968	155
accuracy			0.9955	1340
macro avg	0.9955	0.9957	0.9956	1340
weighted avg	0.9955	0.9955	0.9955	1340

Xception

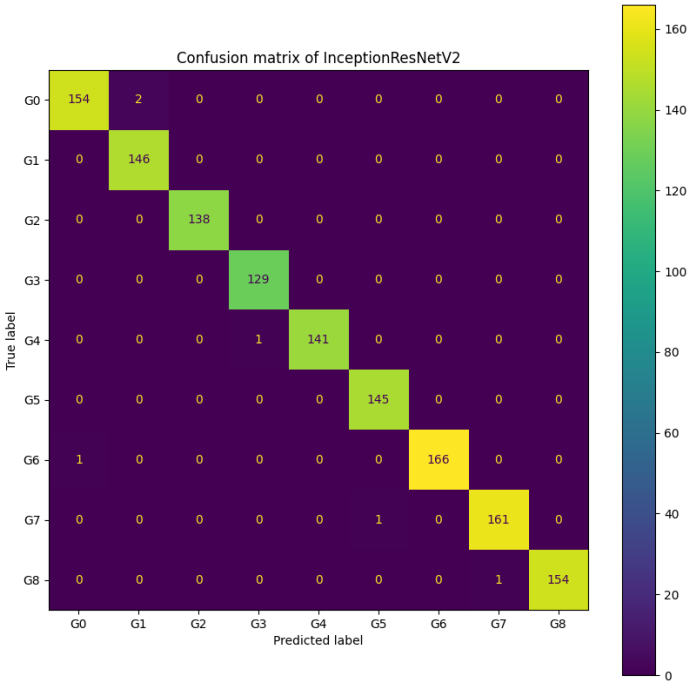
	precision	recall	f1-score	support
G0	0.9935	0.9872	0.9904	156
G1	0.9932	1.0000	0.9966	146
G2	1.0000	1.0000	1.0000	138
G3	1.0000	1.0000	1.0000	129
G4	1.0000	1.0000	1.0000	142
G5	1.0000	0.9931	0.9965	145
G6	0.9940	1.0000	0.9970	167
G7	0.9938	0.9938	0.9938	162
G8	1.0000	1.0000	1.0000	155
accuracy			0.9970	1340
macro avg	0.9972	0.9971	0.9971	1340
weighted avg	0.9970	0.9970	0.9970	1340

[20] Vegeu https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html

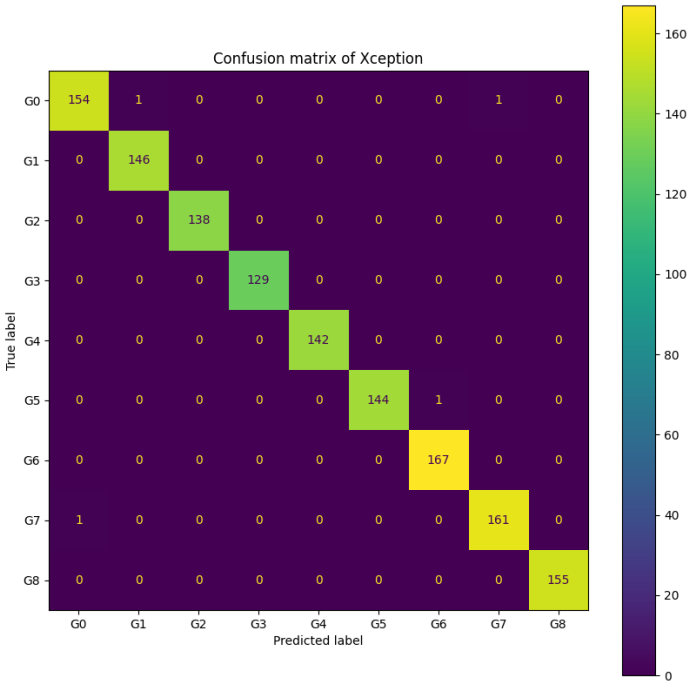
DenseNet121

	precision	recall	f1-score	support
G0	0.9810	0.9936	0.9873	156
G1	1.0000	0.9932	0.9966	146
G2	1.0000	1.0000	1.0000	138
G3	1.0000	1.0000	1.0000	129
G4	1.0000	1.0000	1.0000	142
G5	0.9724	0.9724	0.9724	145
G6	0.9702	0.9760	0.9731	167
G7	1.0000	0.9877	0.9938	162
G8	1.0000	1.0000	1.0000	155
accuracy			0.9910	1340
macro avg	0.9915	0.9914	0.9915	1340
weighted avg	0.9911	0.9910	0.9911	1340

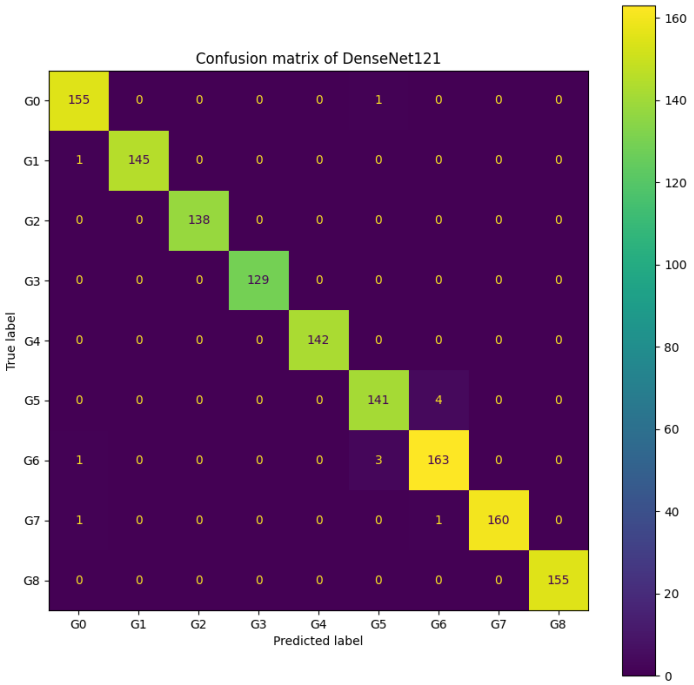
Seguidament, es presenten les matrius de confusió per als tres models entrenats –InceptionResNetV2 en la Il·lustració 22, Xception en la Il·lustració 23 i DenseNet121 en la Il·lustració 24–.



Il·lustració 22: matriu de confusió per al conjunt de test del model InceptionResNetV2



Il·lustració 23: matriu de confusió per al conjunt de test del model Xception



Il·lustració 24: matriu de confusió per al conjunt de test del model DenseNet121

Es pot afirmar que els valors de les mètriques principals d'interés de tots tres models (precisió, reclam i *F1-score*) són molt elevats, fet que confirma la bona tendència d'entrenament vista durant el procés d'entrenament i en les corbes de pèrdua presentades més amunt. No obstant això, com s'ha dit, es farà una comprovació addicional amb un conjunt de test extra, provinent d'un vídeo d'un

dia diferent, amb roba i un fons una mica diferents, per tal de verificar que el funcionament global siga prou bo per a passar a la fase d'inferència en temps real. Tot seguit se'n presenten els resultats.

InceptionResNetV2

	precision	recall	f1-score	support
G0	1.0000	1.0000	1.0000	93
G1	1.0000	1.0000	1.0000	89
G2	1.0000	1.0000	1.0000	99
G3	1.0000	1.0000	1.0000	103
G4	1.0000	1.0000	1.0000	119
G5	1.0000	1.0000	1.0000	109
G6	1.0000	1.0000	1.0000	103
G7	1.0000	0.8588	0.9241	85
G8	0.8919	1.0000	0.9429	99
accuracy			0.9867	899
macro avg	0.9880	0.9843	0.9852	899
weighted avg	0.9881	0.9867	0.9865	899

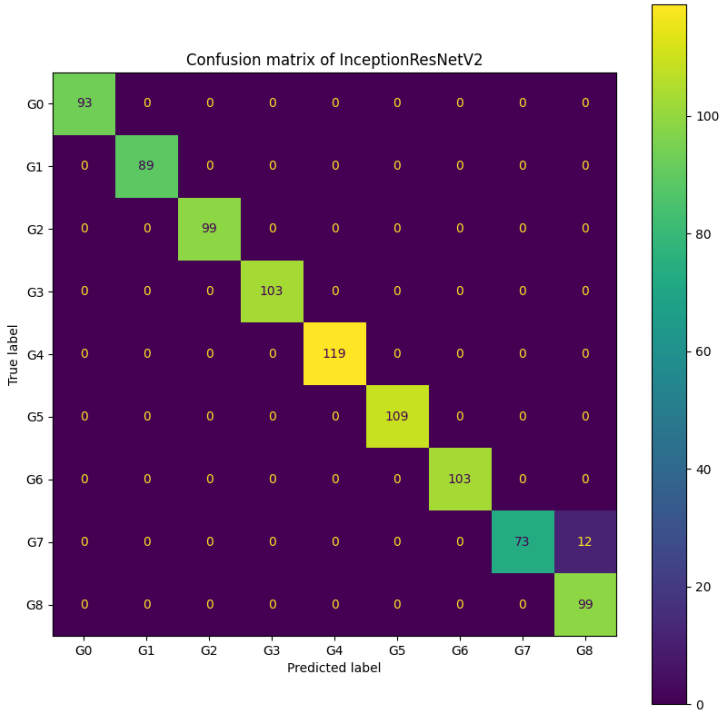
Xception

	precision	recall	f1-score	support
G0	1.0000	1.0000	1.0000	93
G1	1.0000	1.0000	1.0000	89
G2	1.0000	1.0000	1.0000	99
G3	1.0000	1.0000	1.0000	103
G4	1.0000	1.0000	1.0000	119
G5	1.0000	1.0000	1.0000	109
G6	1.0000	1.0000	1.0000	103
G7	1.0000	0.9765	0.9881	85
G8	0.9802	1.0000	0.9900	99
accuracy			0.9978	899
macro avg	0.9978	0.9974	0.9976	899
weighted avg	0.9978	0.9978	0.9978	899

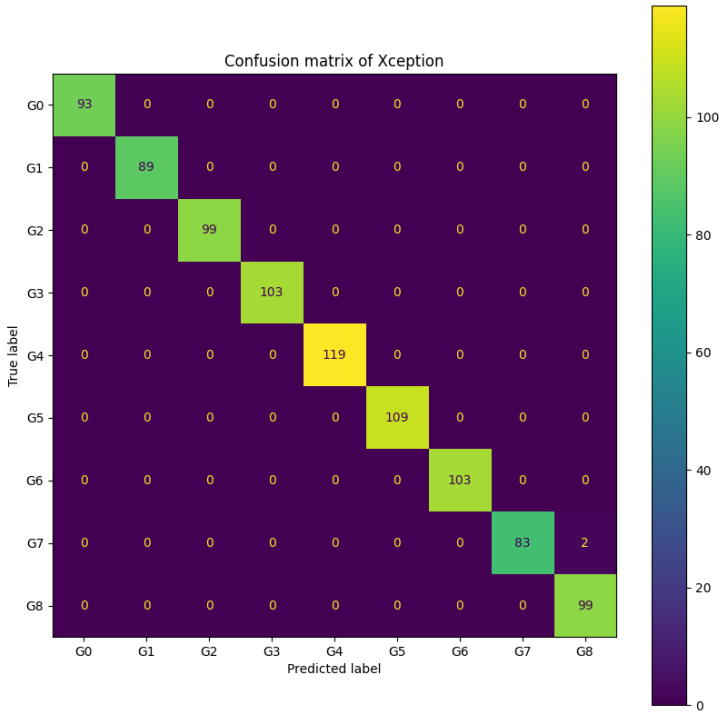
DenseNet121

	precision	recall	f1-score	support
G0	0.9894	1.0000	0.9947	93
G1	1.0000	1.0000	1.0000	89
G2	1.0000	1.0000	1.0000	99
G3	1.0000	1.0000	1.0000	103
G4	1.0000	1.0000	1.0000	119
G5	1.0000	1.0000	1.0000	109
G6	1.0000	1.0000	1.0000	103
G7	1.0000	0.9647	0.9820	85
G8	0.9802	1.0000	0.9900	99
accuracy			0.9967	899
macro avg	0.9966	0.9961	0.9963	899
weighted avg	0.9967	0.9967	0.9966	899

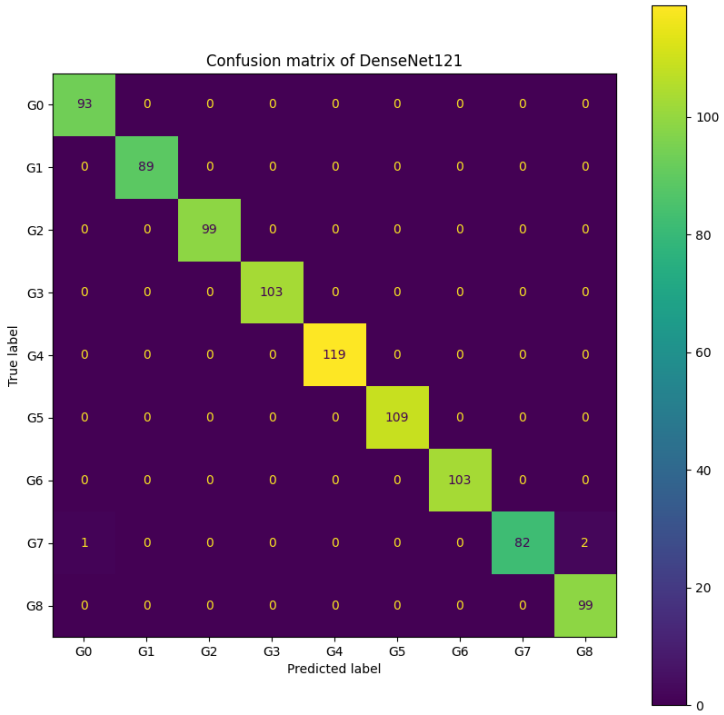
Com abans, es presenten les matrius de confusió per al conjunt de test extra i els tres models entrenats –InceptionResNetV2 en la Il·lustració 25, Xception en la Il·lustració 26 i DenseNet121 en la Il·lustració 27–.



Il·lustració 25: matriu de confusió per al conjunt de test extra del model InceptionResNetV2



Il·lustració 26: matriu de confusió per al conjunt de test extra del model Xception



Il·lustració 27: matriu de confusió per al conjunt de test extra del model DenseNet121

S'observa que els resultats són molt similars en la detecció dels fotogrames del vídeo de profunditat totalment independent de les dades i, com es podia esperar, el nivell de precisió continua de ser elevat. En conseqüència, es pot considerar que tots tres models generalitzen correctament les dades mai vistes en test.

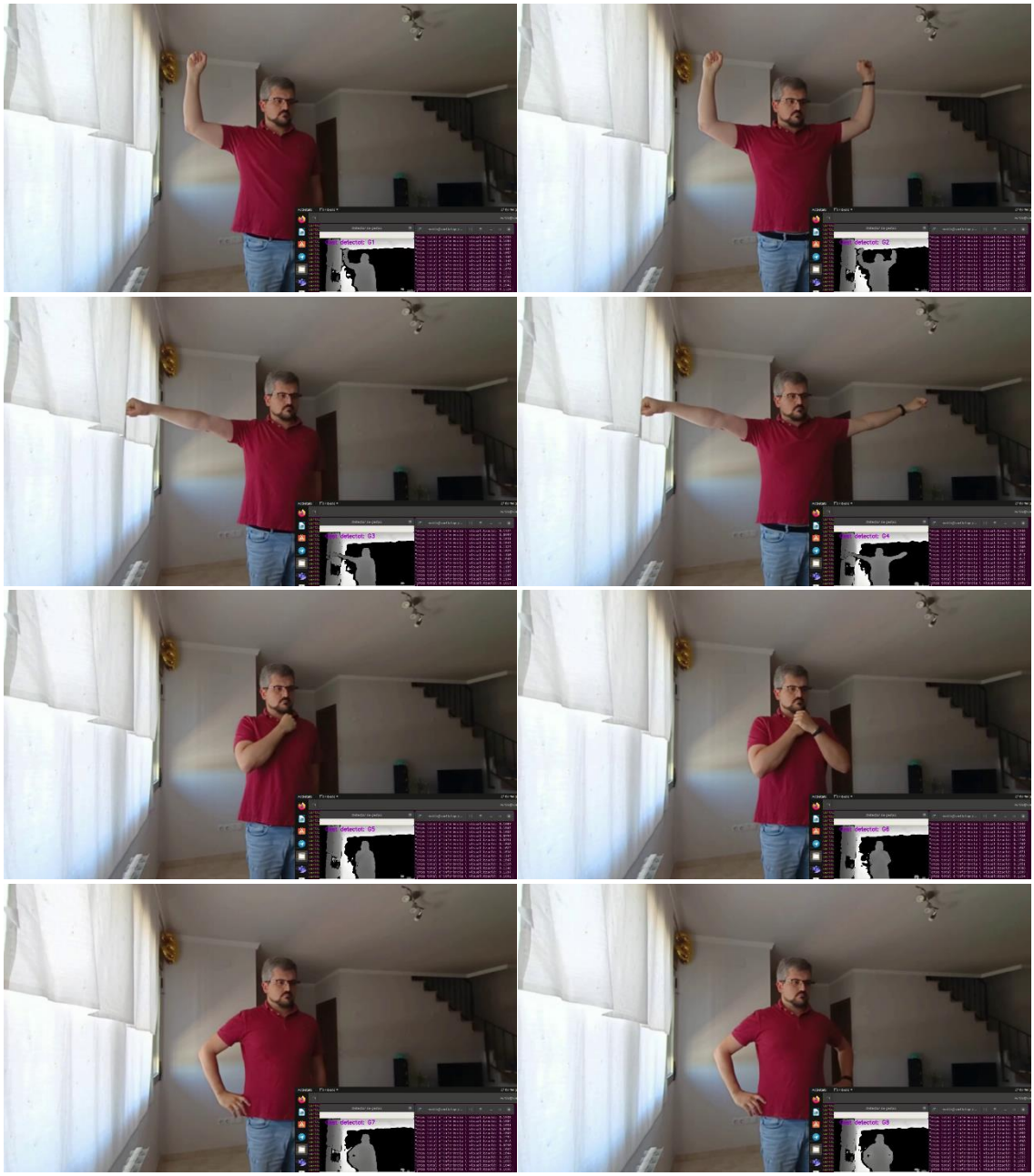
Comptat i debatut, aquests resultats permeten de fer-los servir amb confiança per a la inferència en el sistema final. Tanmateix, atesa la minsa diferència de rendiment que hi ha entre aquests models, probablement la millor opció és optar per la xarxa més lleugera tant pel que fa a temps d'inferència com a mida en memòria –DenseNet121– per a obtindre una càrrega computacional mínima.

3.7 Demostració de funcionament del sistema

Finalment, es farà una prova de concepte del sistema posant en inferència el model final seleccionat –DenseNet121– en el sistema de detecció de gestos compost pel mòdul de computació mòbil NVIDIA Jetson Nano i la càmera de profunditat Intel RealSense, en un esquema idèntic al que s'ha fet servir en la captura de la base de dades d'entrenament.

El codi inclou la càrrega del model final i un bucle infinit (fins que no es prem la tecla ESC) que espera fins que hi ha un fotograma disponible provinent de la càmera, el processa de la mateixa manera que en les imatges d'entrenament, validació i test, i l'envia al model per tal d'obtindre'n la predicció, la qual es mostra

en la finestra superposada a la imatge monocromàtica de profunditat. Com abans, el codi d'inferència es troba en l'annex del subapartat 7.1.



Il·lustració 28: exemples d'inferència en el sistema final fent servir el model DenseNet121 (d'esquerra a dreta i de dalt a baix: de G1 a G8)

En la Il·lustració 28 es presenten una sèrie de captures preses en temps real pel sistema en inferència i l'anàlisi que en fa el sistema d'aprenentatge profund en poques dècimes de segon (entre 0,1 i 0,25 segons en funció del model i de l'estat del processador), la immensa majoria de les quals són classificades correctament. Aquests fotogrames s'han extret d'un dels vídeos d'exemple –un per cada model– que s'han preparat com a demostració del sistema, i en aquestes seqüències es pot verificar la validesa de la solució desenvolupada.

4. Conclusions

4.1 Assoliment dels objectius

Una vegada s'ha aconseguit la consecució del projecte, es recorden els objectius generals del treball que es van exposar al principi del desenvolupament:

- Obtindre els fotogrames de profunditat del conjunt de gestos corporals
- Entrenar diversos models d'aprenentatge profund per a detecció de gestos i validar-los
- Implementar el model per a inferència en el sistema d'interfície natural d'usuari

S'observa que aquestes tres passes són les principals en la canonada d'aprenentatge automàtic que s'ha implementat, per la qual cosa, sembla clar que s'han assolit completament, incloent-hi un muntatge com a prova del sistema final per a la detecció de gestos.

Més en detall, primerament, s'han gravat les seqüències per gest amb diversos usuaris i se n'han extret els fotogrames anotats. Després, s'ha programat una canonada d'aprenentatge automàtic en les fases d'entrenament (obtenció de models) i test (verificació del rendiment d'aquests models). Finalment, s'ha implementat el model més lleuger per a inferència i se n'ha fet una demostració en temps real. Per tant, els objectius específics també s'han aconseguit.

4.2 Lliçons apreses

Al llarg de l'elaboració del projecte s'han extret moltes ensenyances ben útils en el dia a dia d'un enginyer que es dedique a l'aprenentatge automàtic. En primer lloc, un fet principal que s'ha aprés és que si les dades es prenen d'una font contínua (per exemple, un vídeo amb valors de FPS elevats), encara que s'assegure que cap fotograma no es fa servir tant en entrenament com en test, igualment es podria arribar a obtindre una estimació optimista del funcionament real dels models en predicció perquè les dades d'entrenament i test poden assemblar-se molt, encara que no siguen exactament iguals. Per tant, sempre es recomana de dur a terme alguna prova addicional amb dades completament noves, mai vistes pel model, per tal de conéixer-ne de manera fidedigna el rendiment predictiu.

Un altre element aprés ha estat la dificultat que suposa el treball en alguns entorns de desenvolupament com ara les plaques NVIDIA Jetson. En efecte, encara que la instal·lació i ús del sistema operatiu és trivial gràcies a les

instruccions preparades pel fabricant, no passa el mateix amb algunes llibreries a causa que es tracta de mòduls basats en processadors ARM (també passa el mateix amb els models de Raspberry Pi, per exemple). Conseqüentment, aquests no fan servir les mateixes instruccions que els processadors de propòsit general, per la qual cosa pot caldre una compilació específica d'alguns elements de programari.

Finalment, un darrer fet remarcable ha estat la poca diferència en el rendiment dels models a pesar de tindre un nombre de paràmetres molt diferent (amb una variació de fins a 6 vegades més, segons el cas). En models més pesants es troben millores molt marginals –o fins i tot, inexistents– en el rendiment de classificació i, per tant, no sol pagar la pena de fer-los servir si no es té una aplicació en la qual aprofitar qualsevol increment de predicció del model és imprescindible.

4.3 Adequació de la metodologia i de la planificació

La metodologia que s'ha fet servir finalment ha estat la prevista. Els pilars bàsics del treball s'han mantingut –ús de càmera de profunditat per a respectar la privacitat, establiment d'un sistema modular amb possibilitat de reutilització o inclusió de més gestos, paradigma d'aprenentatge automàtic basat concretament en aprenentatge profund–. A més a més, les fases generals del projecte han anat emparellades amb les quatre fases del procés d'obtenció del sistema: extracció i preprocessament, entrenament, validació i implementació i inferència, si bé amb alguna realimentació puntual cap al final del projecte.

Pel que fa a la planificació, en termes generals, es pot dir que el diagrama de Gantt del subapartat 1.4 s'ha complit de manera estricta, amb l'única excepció que la fase de finalització de la memòria s'ha aprofitat no només per a aquesta comesa, sinó també per a refinar el sistema de detecció, especialment quant a test i inferència. Aquest fet es deriva, principalment, de les complicacions mencionades més amunt trobades en la instal·lació d'alguns paquets en la placa Jetson, però també del fet d'haver hagut d'obtindre un conjunt de dades addicional i independent per a test.

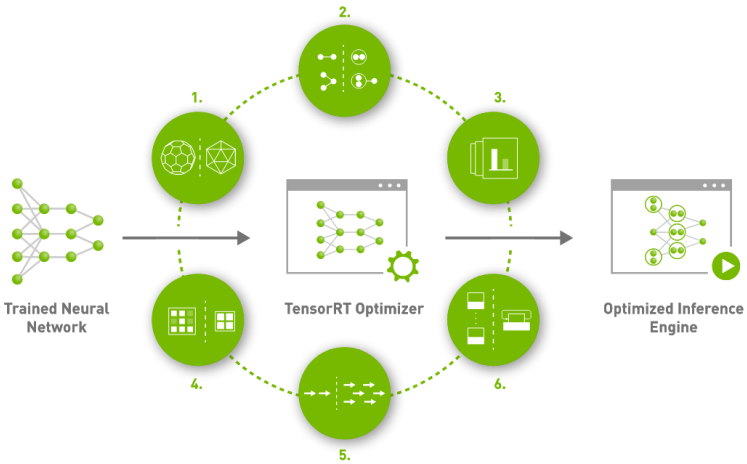
4.4 Treball futur

En aquest treball s'ha desenvolupat tota la canonada d'aprenentatge automàtic que va de l'arreplega i el preprocessament de dades, l'entrenament i la validació de models al test final i la implementació d'una demostració per a inferència. Tanmateix, hi ha un procés opcional addicional que podria ser interessant de valorar: l'optimització en inferència. Aquest procés consisteix a reduir la precisió dels nombres que representen els pesos entrenats dels models per tal de

disminuir la càrrega computacional dels càlculs matricials i, en conseqüència, incrementar-ne la velocitat d'execució.

Il·lustració 29: optimitzacions d'una xarxa SSD Inception-V2 en una targeta gràfica NVIDIA (Fierval, 2019)

Un enfocament típic en les plaques de desenvolupament de NVIDIA és l'ús de TensorRT (NVIDIA Developer, 2022), que permet el pas de FP32 (tipus de dades habitual en coma flotant de 32 bits) a FP16 (coma flotant de 16 bits) o a INT8 (enter de 8 bits), per bé que aquest últim tipus no és compatible amb les plaques Jetson. Aquesta optimització es basa en la càrrega de models en format nadiu (TensorFlow, PyTorch o Matlab) o, de manera més general, en format universal ONNX, per la qual cosa, un pas previ habitual és la conversió del model desat en format binari de TensorFlow (per exemple `.h5`) a aquest format.



Il·lustració 30: funcionament d'alt nivell de TensorRT (NVIDIA Developer, 2022)

En la Il·lustració 29 (Fierval, 2019) es pot veure un exemple d'aquest tipus d'optimització i el guany tan gran que se n'obté. Així mateix, encara que aquesta aproximació provoca a vegades una lleu davallada en el rendiment, sovint es tracta d'una pèrdua anecdòtica que en les aplicacions del món real gairebé sempre és assumible (Lee, 2017). Per tant, un treball futur recomanable seria

exportar el model final obtingut a ONNX i importar-lo en TensorRT per tal d'optimitzar-ne l'execució i obtindre un motor d'inferència optimitzat, tal com es mostra a la Il·lustració 30.

5. Glossari

Tot seguit, es presenten els termes i acrònims més importants que apareixen al llarg d'aquesta memòria. Les definicions han estat elaborades i completades per l'autor prenent com a base algunes fonts, entre les quals destaquen el Termcat[21], la Wikipedia[22] (en anglés) i la Viquipèdia[23] (en català).

- Aprenentatge automàtic (*machine learning*, ML): Procés del camp de la intel·ligència artificial dedicat a l'anàlisi, el disseny i el desenvolupament d'algorismes i tècniques que permeten que les màquines evolucionen i milloren el seu comportament a partir de l'estudi d'observacions o de les experiències pròpies.

- Aprenentatge profund (*deep learning*, DL): Tipus de procés d'aprenentatge automàtic basat en l'extracció i transformació de noves característiques del processament de la informació, per mitjà d'algoritmes que funcionen en un sistema per capes que s'inspira el funcionament bàsic del sistema neuronal del cervell.

- Augmentació de dades (*data augmentation*): Tècnica per a augmentar la quantitat de dades d'entrada mitjançant la qual es generen còpies sintètiques lleugerament modificades de les dades originals de manera que es conserven les característiques semàntiques necessàries per a la tasca.

- Canonada (*pipeline*): Paradigma molt habitual en aprenentatge automàtic que permet encadenar els processos de tractament de dades de manera que l'execució de les diverses fases de cada procés és contínua i eficient.

- Computació a la vora (*edge computing*): Sistema d'emmagatzematge i ús de recursos informàtics dut a terme a prop de la ubicació física d'un dispositiu o en la font de dades.

- Computació en núvol (*cloud computing*): Sistema d'emmagatzematge i ús de recursos informàtics basat en el servei en xarxa, que desplaça la càrrega de les tasques a un servidor central.

- Embolcall *(wrapper)*: Programari que facilita l'adaptació d'un programa a un sistema operatiu o entorn diferent d'aquell per al qual ha estat concebut.

- Època (*epoch*): Nombre de lots que constitueixen el conjunt de dades d'entrenament complet quan passen una vegada per la xarxa.

- Entorn de desenvolupament integrat (*integrated development environment*, IDE): Programari que serveix per a escriure, compilar i depurar components o aplicacions, de vegades incorporat a altres productes de programari.

[21] Disponible en https://www.termcat.cat/ca
[22] Disponible en https://en.wikipedia.org/
[23] Disponible en https://ca.wikipedia.org/

- **Equip de desenvolupament de programari (*software development kit*, SDK):** Conjunt d'eines de desenvolupament de programari que permeten al programador crear aplicacions per a un sistema concret.

- **Espectre infraroig (*infrared*, IR):** Banda de l'espectre electromagnètic que comprèn l'interval que va des de la llum visible fins a les microones, amb longituds d'ona compreses entre 0,75 i 1.000 μm.

- **Espectre visible:** Banda de l'espectre electromagnètic que correspon a la pròpia de la visió humana, amb longituds d'ona compreses entre 0,3 i 0,7 μm, aproximadament.

- **Funció d'activació:** Funció no lineal que s'aplica després del càlcul de multiplicació i suma en cada neurona d'una xarxa neuronal i que permet que la xarxa aproxime funcions arbitràriament complexes.

- **Funció de pèrdua:** Mètrica clau per al procés d'aprenentatge automàtic que representa la desviació d'una determinada predicció efectuada per un algorisme respecte del valor real cercat.

- **Hiperparàmetre:** Cadascun dels paràmetres del model i del procés d'entrenament que no s'entrenen, sinó que es defineixen de primeres.

- **Intel·ligència artificial (*artificial intelligence*, AI):** Part de la informàtica dedicada al desenvolupament d'algorismes que permeten a una màquina, basant-se en la percepció del seu entorn lògic, prendre decisions percebudes com a intel·ligents pels humans, típicament per a maximitzar les seves possibilitats d'èxit en una o més tasques.

- **Interfície humà-màquina (*human-machine interface*, HMI):** Mitjà que permet la relació i entre els humans i els equips informàtics mitjançant diverses funcions o característiques.

- **Interfície intel·ligent d'usuari (*intelligent user interface*, IUI):** Tipus d'interfície d'usuari que involucra algun aspecte d'intel·ligència artificial.

- **Interfície natural d'usuari (*natural user interface*, NUI):** Tipus d'interfície d'usuari en el qual s'interactua amb un sistema sense utilitzar sistemes de comandament o dispositius d'entrada i, en comptes d'això, típicament es fa ús de gestos corporals o reconeixement de la parla.

- **Internet de les coses (*internet of things*, IoT):** Xarxa formada per un conjunt d'objectes connectats a internet que es poden comunicar entre si i també amb humans, i així transmetre i tractar dades amb intervenció humana o sense.

- **Llum estructurada:** Llum que es projecta a través d'una trama que forma un patró geomètric sobre la superfície d'un objecte que s'ha de fotografiar, amb l'objectiu de proporcionar textura a superfícies que no en tenen o millorar la presa de punts tridimensionals i la determinació automàtica de punts homòlegs.

- **Lot (*batch*):** Conjunt de dades que es fa passar per la xarxa en una mateixa iteració.

- **Model:** Algorisme unitari d'aprenentatge automàtic que pren una o diverses entrades de dades i les processa per a obtindre unes prediccions (típicament d'agrupament, classificació, regressió o detecció).

- Parada primerenca (*early stopping*): Mecanisme que permet d'aturar l'entrenament d'un model d'aprenentatge automàtic si no es produeixen millores en un nombre determinat d'èpoques.

- Pesos (*weights*): Conjunt de coeficients que pren un determinat model després d'haver-se entrenat.

- Taxa d'entrenament (*learning rate*, LR): Escalar, generalment molt inferior a 1, pel qual es multiplica el pas teòric d'aprenentatge d'una xarxa neuronal per a permetre'n la convergència i l'estabilitat.

- Transferència d'aprenentatge (*transfer learning*): Paradigma d'aprenentatge automàtic que consisteix a aprofitar el coneixement generat en la resolució d'un problema per a acostar-se a la resolució d'altres de similars.

6. Bibliografia

Andersen, M. R. [Michael Riis], Jensen, T. [Thomas], Lisouski, P. [Pavel], Mortensen, A. K. [Anders Krogh], Hansen, M. [Mikkel], Gregersen, T. [Torben] i Ahrendt, P. [Peter] (2012). *Kinect depth sensor evaluation for computer vision applications*. Aarhus University. https://tidsskrift.dk/ece/article/download/21221/18710/48374

Aufranc, J. L. [Jean-Luc]. (2020, 16 de maig). NVIDIA Jetson Developer Kits Comparison – Nano vs TX2 vs Xavier NX vs AGX Xavier [entrada de blog]. *CNX Software – Embedded Systems News*. https://www.cnx-software.com/2020/05/16/nvidia-jetson-developer-kits-comparison-nano-vs-tx2-vs-xavier-nx-vs-agx-xavier/

Benitez-Garcia, G. [Gibran], Olivares-Mercado, J. [Jesus], Sanchez-Perez, G. [Gabriel] i Yanai, K. [Keiji]. (2021). IPN hand: A video dataset and benchmark for real-time continuous hand gesture recognition. A *25th International Conference on Pattern Recognition 2020* (pp. 4340-4347). https://doi.org/10.1109/ICPR48806.2021.9412317

Biswas, K. K. [Kanad Kishore] i Basu, S. K. [Saurav Kumar]. (2011). Gesture recognition using Microsoft Kinect®. A *5th International Conference on Automation, Robotics and Applications* (pp. 100-103). https://doi.org/10.1109/ICARA.2011.6144864

Blažica, B. [Bojan]. (2015, 8 de desembre). Natural user interfaces – NUI [entrada de blog]. *HCI Slovenia.* https://hci.si/2015/12/08/natural-user-interfaces-nui/

Brock, H. [Heike], Sabanovic, S. [Selma], Nakamura, K. [Keisuke] i Gomez, R. [Randy]. (2020). Robust real-time hand gestural recognition for non-verbal communication with tabletop robot Haru. A *29th IEEE international conference on robot and human interactive communication 2020* (pp. 891-898). https://doi.org/10.1109/RO-MAN47096.2020.9223566

Buntine, W. [Wray]. (2020). Machine learning after the deep learning revolution. *Frontiers of Computer Science, 14*(6), 1-3. https://doi.org/10.1007/s11704-020-0800-8

Cai, Z. [Ziyun], Han, J. [Jungong], Liu, L. [Li] i Shao, L. [Ling]. (2017). RGB-D datasets using Microsoft Kinect or similar sensors: a survey. *Multimedia Tools and Applications, 76*(3), 4313-4355. https://doi.org/10.1007/s11042-016-3374-6

Cao, Z. [Zhe], Simon, T. [Tomas], Wei, S. E. [Shih-En] i Sheikh, Y. [Yaser] (2017). Realtime multi-person 2d pose estimation using part affinity fields. A *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 7291-7299). https://arxiv.org/abs/1611.08050

Chollet, F. [François]. (2017). Xception: Deep learning with depthwise separable convolutions. A *Proceedings of the IEEE conference on computer vision and pattern recognition* (p. 1251-1258). https://doi.org/10.48550/arXiv.1610.02357

Dardas, N. [Nasser] i Georganas, N. [Nicolas]. (2011). Real-time hand gesture detection and recognition using bag-of-features and support vector machine techniques. *IEEE Transactions on Instrumentation and measurement, 60*(11), 3592-3607. https://doi.org/10.1109/TIM.2011.2161140

Dardas, N. [Nasser] i Petriu, E. [Emil]. (2011). Hand gesture detection and recognition using principal component analysis. A 2011 *Proceedings of the IEEE International conference on computational intelligence for measurement systems and applications* (pp. 1-6). https://doi.org/10.1109/CIMSA.2011.6059935

De Los Pobres, A. [Aranzulla]. (2020, 19 de novembre). ToF vs. LiDAR: ¿Cuál es la diferencia? [entrada de blog]. *TecnoLoco*. https://tecnoloco.istocks.club/tof-vs-lidar-cual-es-la-diferencia/2020-11-19/

Euskaltel. (2015, 3 de novembre). Las edades de la tecnologia móvil: 1G, 2G, 3G, 4G... [entrada de blog]. *Euskaltel Blog*. https://blog.euskaltel.com/las-edades-de-la-tecnologia-movil-1g-2g-3g-4g/

Fierval. (2019, 25 de març). Supercharging Object Detection in Video: TensorRT 5 [entrada de blog]. *Viral F#: F#, CUDA, Computer Vision, Deep Learning*. https://viralfsharp.com/2019/03/25/supercharging-object-detection-in-video-tensorrt-5/

Francke, H. [Hardy], Ruiz-del-Solar, J. [Javier] i Verschae, R. [Rodrigo]. (2007). Real-time hand gesture detection and recognition using boosted classifiers and active learning. A *Pacific-rim symposium on image and video technology* (pp. 533-547). https://doi.org/10.1007/978-3-540-77129-6_47

Hayes, B. [Brian]. (2008). Cloud Computing. *Communications of the ACM, 51*(7), 9-11. https://dl.acm.org/doi/pdf/10.1145/1364782.1364786

He, Y. [Ying], Liang, B. [Bin], Zou, Y. [Yu], He, J. [Jin] i Yang, J. [Jun]. (2017). Depth errors analysis and correction for time-of-flight (ToF) cameras. *Sensors, 17*(1), 92. https://doi.org/10.3390/s17010092

Huang, G. [Gao], Liu, Z. [Zhuang], Van Der Maaten, L. [Laurens] i Weinberger, K. [Kilian]. (2017). Densely connected convolutional networks. A *Proceedings of the IEEE conference on computer vision and pattern recognition* (p. 4700-4708). https://doi.org/10.48550/arXiv.1608.06993

Hvas, D. [Ditte]. (2020, 5 de juliol). Natural User Interfaces – What are they and how do you design user interfaces that feel natural? [entrada de blog]. *Interaction Design Foundation.* https://www.interaction-design.org/literature/article/natural-user-interfaces-what-are-they-and-how-do-you-design-user-interfaces-that-feel-natural

Intel Corporation. (2019, 15 de juliol). Beginners guide to depth (updated) [entrada de blog]. *Intel RealSense*. https://www.intelrealsense.com/beginners-guide-to-depth/

Intel Corporation. (2022a). *Intel® RealSense™ Depth Camera D435*. https://ark.intel.com/content/www/us/en/ark/products/128255/intel-realsense-depth-camera-d435.html

Intel RealSense. (2019, 15 de juliol). Beginner's guide to depth (updated) [entrada de blog]. *Intel RealSense Blog*. https://www.intelrealsense.com/beginners-guide-to-depth/

Intel RealSense. (2022). *Compare cameras: contrast and compare our full lineup of quality computer vision devices*. https://www.intelrealsense.com/compare-depth-cameras/

Intel Corporation. (2022b). *Intel® RealSense™ Depth Camera D455*. https://ark.intel.com/content/www/us/en/ark/products/205847/intel-realsense-depth-camera-d455.html

Keras. (2022). *Keras Applications.* https://keras.io/api/applications/

Kurakin, A. [Alexei], Zhang, Z. [Zhengyou] i Liu, Z. [Zicheng]. (2012). A real time system for dynamic hand gesture recognition with a depth sensor. A *2012 Proceedings of the 20th European signal processing conference (EUSIPCO)* (pp. 1975-1979). https://doi.org/10.5281/zenodo.42817

LeCun, Y. [Yann], Bengio, Y. [Yoshua] i Hinton, G. [Geoffrey]. (2015). Deep learning. *Nature, 521*(7553), 436-444. https://doi.org/10.1038/nature14539

Li, Y. [Yi]. (2012). Hand gesture recognition using Kinect. A *2012 IEEE International Conference on Computer Science and Automation Engineering* (pp. 196-199). https://doi.org/10.1109/ICSESS.2012.6269439

Lee, J. [Joohoon]. (2017, 11 de desembre). Fast INT8 Inference for Autonomous Vehicles with TensorRT 3 [entrada de blog]. *NVIDIA Developer: Technical Blog*. https://developer.nvidia.com/blog/int8-inference-autonomous-vehicles-tensorrt/

Liang, B [Bin] i Zheng, L. [Lihong]. (2015). A survey on human action recognition using depth sensors. A *2015 International conference on digital image computing: techniques and applications (DICTA)* (pp. 1-8). https://doi.org/10.1109/DICTA.2015.7371223

Lowensohn, J. [Josh]. (2011, 23 de febrer). Timeline: A look back at Kinect's history [entrada de blog]. *CNET Tech*. https://www.cnet.com/tech/tech-industry/timeline-a-look-back-at-kinects-history/

Macher, G. [George], Druml, N. [Norbert], Veledar, O. [Omar] i Reckenzaun, J. [Jakob]. (2019). Safety and security aspects of fail-operational urban surround perceptION (FUSION). A *International Symposium on Model-Based Safety and Assessment* (pp. 286-300). https://doi.org/10.1007/978-3-030-32872-6_19

Mejia-Trujillo, J. D. [Jeison David], Castaño-Pino, Y. J. [Yor Jaggy], Navarro, A. [Andrés], Arango-Paredes, J. D. [Juan David], Rincón, D. [Domiciano], Valderrama, J. [Jaime], Muñoz, B. [Beatriz] i Orozco, J. L. [Jose Luis]. (2019). Kinect™ and Intel RealSense™ D435 comparison: a preliminary study for motion analysis. A *2019 IEEE International Conference on E-health Networking, Application & Services (HealthCom)* (pp. 1-4). https://doi.org/10.1109/HealthCom46333.2019.9009433

Myers, B. [Brad]. (1998). A brief history of human-computer interaction technology. *Interactions, 5*(2), 44-54.

Nair, M. [Mukul] i Noel, S. [Sherly]. (2021). Hand Gesture Recognition using Double CNN and Transfer Learning. A *International Conference on Communication, Control and Information Sciences 2021 Vol. 1* (pp. 1-6). https://doi.org/10.1109/ICCISc52257.2021.9485015

Neverova, N. [Natalia], Wolf, C. [Christian], Paci, G. [Giulio], Sommavilla, G. [Giacomo], Taylor, G. [Graham] i Nebout, F. [Florian]. (2013). A multi-scale approach to gesture detection and recognition. A *Proceedings of the IEEE International Conference on Computer Vision Workshops 2013* (pp. 484-491). https://doi.org/10.1109/ICCVW.2013.69

Neverova, N. [Natalia], Wolf, C. [Christian], Taylor, G. [Graham] i Nebout, F. [Florian]. (2014). Multi-scale deep learning for gesture detection and localization. A *European Conference on Computer Vision 2014* (pp. 474-490). https://doi.org/10.1007/978-3-319-16178-5_33

NVIDIA. (2022). *Jetson Store*. https://www.nvidia.com/en-us/autonomous-machines/jetson-store/

NVIDIA Developer. (2022). *NVIDIA TensorRT*. https://developer.nvidia.com/tensorrt

Red Hat. (2021, 14 de maig). ¿Qué es la arquitectura de edge computing? [entrada de blog]. *Red Hat*. https://www.redhat.com/es/topics/edge-computing/what-is-edge-architecture?

Ren, Z. [Zhou], Yuan, J. [Junsong], Meng, J. [Jingjing] i Zhang, Z. [Zhengyou]. (2013). *Robust part-based hand gesture recognition using Kinect sensor. IEEE transactions on multimedia, 15*(5), 1110-1120. https://doi.org/10.1109/TMM.2013.2246148

Reyes, M. [Miguel], Dominguez, G. [Gabriel] i Escalera, S. [Sergio]. (2011). Featureweighting in dynamic timewarping for gesture recognition in depth data. A *2011 IEEE International Conference on Computer Vision Workshops* (pp. 1182-1188). https://doi.org/10.1109/ICCVW.2011.6130384

Satyanarayanan, M. [Mahadev]. (2017). The emergence of edge computing. *Computer, 50*(1), 30-39. https://doi.org/10.1109/MC.2017.9

Shi, W. [Weisong], Cao, J. [Jie], Zhang, Q. [Quan], Li, Y. [Youhuizi] i Xu, L. [Lanyu]. (2016). Edge computing: vision and challenges. *IEEE Internet of Things journal, 3*(5), 637-646. https://doi.org/10.1109/JIOT.2016.2579198

Soloelectronicos. (2021, 29 d'agost). Primeros pasos con Jupyter Notebook [entrada de blog]. *soloelectronicos.com*. https://soloelectronicos.com/2021/08/29/primeros-pasos-con-jupyter-norebook/

Song, X. [Xiaoyu], Chen, H. [Hong] i Wang, Q. [Qing]. (2020). A Gesture Recognition Approach Using Multimodal Neural Network. A *Journal of Physics: Conference Series Vol. 1544, No. 1* (p. 12127-12132). https://www.doi.org/10.1088/1742-6596/1544/1/012127

Szegedy, C. [Christian], Ioffe, S. [Sergey], Vanhoucke, V. [Vincent] i Alemi, A. [Alexander]. (2017). Inception-V4, Inception-ResNet and the impact of residual connections on learning. A *Thirty-first AAAI Conference on Artificial Intelligence* (p. 4278-4284). https://doi.org/10.48550/arXiv.1602.07261

Szymczyk, M. [Matthew]. (2014, 12 de juny). How does the Kinect 2 compare to the Kinect 2? [entrada de blog]. *Zugara*. http://zugara.com/how-does-the-kinect-2-compare-to-the-kinect-1

Tillman, M. [Maggie]. (2021, 9 d'abril). What is a ToF camera? Time-of-flight sensor and photography explained [entrada de blog]. *Pocket-lint*. https://www.pocket-lint.com/phones/news/147024-what-is-a-time-of-flight-camera-and-which-phones-have-it

Tölgyessy, M. [Michal], Dekan, M. [Martin], Chovanec, Ľ. [Ľuboš] i Hubinský, P. [Peter]. (2021). Evaluation of the Azure Kinect and its comparison to Kinect V1 and Kinect V2. *Sensors, 21*(2), 413. https://doi.org/10.3390/s21020413

Yalçin, O. [Orhan]. (2021, 9 de gener). Top 5 Deep Learning Frameworks to Watch in 2021 and Why TensorFlow [entrada de blog]. *Towards Data Science.* https://towardsdatascience.com/top-5-deep-learning-frameworks-to-watch-in-2021-and-why-tensorflow-98d8d6667351

7. Annexos

7.1 Codi desenvolupat

Càrrega de llibreries

```
from PIL import Image
import os
import shutil
import sys
import math
import numpy as np
import random
import glob
import argparse

from keras.utils import np_utils
from tensorflow.keras.models import Model
from tensorflow.keras.applications import *
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Model, load_model, model_from_json
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D, Dropout,
   GlobalAveragePooling2D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras import regularizers
from tensorflow.python.keras.callbacks import EarlyStopping, ModelCheckpoint
from tensorflow.python.framework.ops import disable_eager_execution

from sklearn.metrics import classification_report
import matplotlib.pyplot as plt
from tqdm.keras import TqdmCallback

disable_eager_execution()

from numpy.random import seed; seed(42)
import tensorflow; tensorflow.random.set_seed(42)

print("Num GPUs Available: ", len(tensorflow.config.list_physical_devices('GPU')))
```

Unió de dades de diferents sessions de gravació

```
base_path = "/media/vortiz/MicroSD-128/TFG_PROVES/"

all_csv = glob.glob(base_path+"*/*/*/*.csv")

# Mou els fitxer considerats de validació
for elem in all_csv:
    if not os.path.exists(base_path + "CSV_temp/" + elem.split("/")[-3]):
        os.makedirs(base_path + "CSV_temp/" + elem.split("/")[-3])
    os.rename(elem, base_path+"CSV_temp/"+elem.split("/")[-3]+"/"+elem.split("/")[-
   2]+elem.split("/")[-1])

# Esborra la carpeta primitiva
if os.path.exists(base_path+"CSV/"):
```

```
    shutil.rmtree(base_path+"CSV/")

# Deixa la nova carpeta amb el nom original
os.rename(base_path+"CSV_temp/", base_path+"CSV/")

print("Dades de diversos experiments unides")
```

Separació de particions d'entrenament/validació/test

```
base_path = "/media/vortiz/MicroSD-128/TFG_PROVES/"
random.seed(42)
train_proportion = 0.75
validation_proportion = 0.15

for root, dirs, files in os.walk(base_path+"CSV/", topdown=False):
    # Selecciona conjunts d'entrenament, validació i test
    train_list = random.sample(files, round(train_proportion*len(files)))
    not_train_list = [elem for elem in files if elem not in train_list]
    validation_list = not_train_list[:round(validation_proportion*len(files))]
    test_list = [elem for elem in not_train_list if elem not in validation_list]

    # Crea les carpetes d'entrenament, validació i test
    if not os.path.exists(base_path + "CSV_temp/train/" + root.split("/")[-1]):
        os.makedirs(base_path + "CSV_temp/train/" + root.split("/")[-1])
    if not os.path.exists(base_path + "CSV_temp/validation/" + root.split("/")[-1]):
        os.makedirs(base_path + "CSV_temp/validation/" + root.split("/")[-1])
    if not os.path.exists(base_path + "CSV_temp/test/" + root.split("/")[-1]):
        os.makedirs(base_path + "CSV_temp/test/" + root.split("/")[-1])

    # Mou els fitxers considerats d'entrenament, validació i test
    for elem in train_list:
        os.rename(base_path+"CSV/"+root.split("/")[-1]+"/"+elem,
                  base_path+"CSV_temp/train/"+root.split("/")[-1]+"/"+elem)
    for elem in validation_list:
        os.rename(base_path+"CSV/"+root.split("/")[-1]+"/"+elem,
                  base_path+"CSV_temp/validation/"+root.split("/")[-1]+"/"+elem)
    for elem in test_list:
        os.rename(base_path+"CSV/"+root.split("/")[-1]+"/"+elem,
                  base_path+"CSV_temp/test/"+root.split("/")[-1]+"/"+elem)

# Esborra la carpeta primitiva
if os.path.exists(base_path+"CSV/"):
    shutil.rmtree(base_path+"CSV/")

# Deixa la nova carpeta amb el nom original
os.rename(base_path+"CSV_temp/", base_path+"CSV/")

print("Subconjunts d'entrenament, validació i test generats correctament")
```

Càrrega de CSV i conversió a imatges

```
base_path = "/media/vortiz/MicroSD-128/TFG_PROVES/"

# Màxima distància de detecció
d_max_m = 2.55 # metres
```

```
for root, dirs, files in os.walk(base_path+"CSV/", topdown=False):
    for name in files:
        if ".csv" in name:
            # Carrega cada CSV com un array, converteix-lo a imatge i desa'l
            csv_as_array = np.genfromtxt(root + "/" + name, delimiter=',')
            # Esborra punts més llunyans que d_max_m
            csv_as_array[csv_as_array > d_max_m] = 0
            uint_img = np.round(np.absolute(np.array((csv_as_array/d_max_m)*255)))
                .astype('uint8')
            img = Image.fromarray(uint_img)
            if img.mode != 'RGB':
                img = img.convert('RGB')
            #img.show()
            current_path = root.replace("CSV", "IMG") + "/"
            if not os.path.exists(current_path):
                os.makedirs(current_path)
            img.save(current_path + name.replace("csv", "png"))

print("Imatges de profunditat creades correctament")ñ
```

Funció de preprocessament (identitat)

```
def depth_preprocess(image):
    '''Preprocessa la imatge'''
    return image
```

Definició de generadors de dades d'entrenament

```
def create_train_generators(base_path="/home/vortiz/gesture/TFG_PROVES/"):
    '''Crea generadors de dades d'entrenament i validació'''

    img_width, img_height = 424, 240
    img_shape = (img_height, img_width, 3)
    class_mode ='categorical'
    batch_size = 4

    train_setup = ImageDataGenerator(
        rescale=1./255, # màxim de profunditat en cm
        shear_range=0.15,
        zoom_range=0.15,
        rotation_range=20,
        width_shift_range=0.15,
        height_shift_range=0.15,
        horizontal_flip=True,
        vertical_flip=False,
        preprocessing_function=depth_preprocess)

    no_train_setup = ImageDataGenerator(rescale=1./255,
        preprocessing_function=depth_preprocess)

    train_generator = train_setup.flow_from_directory(
        base_path + "IMG/train",
        target_size=img_shape[:-1],
        batch_size=batch_size,
        class_mode=class_mode,
        seed=42)
```

```
    validation_generator = no_train_setup.flow_from_directory(
        base_path + "IMG/validation",
        target_size=img_shape[:-1],
        batch_size=batch_size,
        class_mode=class_mode,
        seed=42)

    num_classes = len(set(train_generator.classes))

    print("Generadors creats correctament")

    return img_shape, class_mode, batch_size, train_generator, validation_generator,
   num_classes
```

Funció de definició dels entrenaments

```
def train(model_type="InceptionResNetV2", not_trainable=0.95, epochs=20,
   learning_rate=1e-5, patience=5):
    '''Entrena els models'''

    # Carreguem el model preentrenat amb ImageNet (transfer learning)
    possibles = globals().copy()
    possibles.update(locals())
    model_name = possibles.get(model_type)
    if not model_name:
        raise NotImplementedError("Aquest model, (%s), no està previst en
                l'entrenament" % model_type)
    else:
        source = "imagenet" if "NASNet" not in model_type else None
        print("Els pesos precarregats són de", source)
    base_model = model_name(weights=source, include_top=False,
                                input_shape=img_shape)

    # Nombre total de capes
    total_layers = len(base_model.layers)
    print("El nombre total de capes de la xarxa és de", total_layers)

    # Ajustem les capes d'InceptionV3 que restaran fixes i les que entrenarem,
    # al final
    for i, layer in enumerate(base_model.layers):
        # Les primeres capes no s'entrenen i les darreres, sí
        if (i < not_trainable * total_layers):
            # Aquestes no les toquem (per defecte, el valor de trainable és de True)
            layer.trainable = False

    # Afegim noves capes al final per a adaptar el model al nostre problema
    # amb dropout per a evitar el sobreajust i una capa totalment connectada
    x = base_model.output
    x = GlobalAveragePooling2D()(x)
    x = Dropout(0.5)(x)
    x = Dense(512, activation='relu', kernel_regularizer=regularizers.l1_l2(l1=1e-5,
        l2=1e-4), bias_regularizer=regularizers.l2(1e-4),
        activity_regularizer=regularizers.l2(1e-5))(x)
    x = Dropout(0.5)(x)
    x = Dense(128, activation='relu', kernel_regularizer=regularizers.l1_l2(l1=1e-5,
        l2=1e-4), bias_regularizer=regularizers.l2(1e-4),
```

```
    activity_regularizer=regularizers.l2(1e-5))(x)
x = Dropout(0.5)(x)
x = Dense(32, activation='relu', kernel_regularizer=regularizers.l1_l2(l1=1e-5,
    l2=1e-4), bias_regularizer=regularizers.l2(1e-4),
    activity_regularizer=regularizers.l2(1e-5))(x)

# Afegim una última capa totalment connectada, d'eixida, amb num_classes
# neurones, cadascuna representa un possible gest
predictions = Dense(num_classes, activation='sigmoid')(x)

# Creem el model final i el compilem
model = Model(inputs=[base_model.input], outputs=[predictions])
#model.summary()   # representació en mode text del model

# Creem l'optimitzador de tipus Adam amb definició de taxa d'aprentatge
adam_opt = Adam(learning_rate=learning_rate)

# Funció de pèrdua per a problemes de classificació multietiqueta; amb definició
# de taxa d'aprentatge i mètrica de precisió categòrica per al problema de
# classificació multiclasse
model.compile(loss='categorical_crossentropy',
              optimizer=Adam(),
              metrics=['categorical_accuracy'])

# Configurem Eearly stopping perquè l'entrenament es detinga en cas que no
# hi haja millora; també hi ha un checkpoint que permet desar el millor model
early_stopping_callback = EarlyStopping(monitor='val_loss', patience=patience,
    restore_best_weights=True)
checkpoint_callback = ModelCheckpoint(base_path + "MODELS/model_" + model_type +
    "_ultimate.h5", monitor='val_loss', verbose=1, save_best_only=True,
    mode='min')

# Entrenem el model amb les dades dels generadors, mostrem informació de
# l'error en acabar cada època i barres de progrés per batch
history = model.fit(train_generator,
                    epochs=epochs,
                    verbose=2,
                    steps_per_epoch=math.ceil(train_generator.samples/batch_size),
                    validation_data=validation_generator,
                    validation_steps=math.ceil(
                        validation_generator.samples/batch_size),
                    callbacks=[early_stopping_callback, checkpoint_callback])

# Ací hi ha les variables que es desen en cada època; les podrem mostrar al final
print("-------------------------------------------------")
print("Valors que podem mostrar:", history.history.keys())
print("-------------------------------------------------")

# Tracem la gràfica de pèrdua (binary cross-entropy) en entrenament i validació
fig = plt.figure(figsize=(15,8))
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss for'+model_type)
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```

```
    fig.savefig(base_path+'MODELS/training_history_'+model_type+'.png', dpi=fig.dpi)

    # Desem el darrer model en format binari HDF5
    # (el de millor valor en validació ja s'ha desat gràcies al checkpoint)
    if not os.path.exists(base_path + "MODELS/"):
        os.makedirs(base_path + "MODELS/")
    model.save(base_path + "MODELS/model_" + model_type + "_latest.h5")
    print("Estructura i model final desats")
```

Crida dels entrenaments

```
# Execució de l'entrenament amb hiperparàmetres
if __name__ == "__main__":
    '''
    base_path = string(sys.argv[1])
    model_type = string(sys.argv[2]) #"InceptionResNetV2" #"EfficientNetV2S"
        #"EfficientNetV2L" #"Xception" #"NASNetMobile"
    not_trainable = float(sys.argv[3])
    epochs = int(sys.argv[4])
    learning_rate = float(sys.argv[5])
    patience = int(sys.argv[6])
    '''

    parser = argparse.ArgumentParser(description='Executable script to train models.
                Syntax: --base_path --model_type --not_trainable –epochs
                        --learning_rate --patience')
    parser.add_argument("--base_path", type=str,
        default="/home/vortiz/gesture/TFG_PROVES/")
    parser.add_argument("--model_type", type=str, default="Xception")
    parser.add_argument("--not_trainable", type=float, default=0)
    parser.add_argument("--epochs", type=float, default=25)
    parser.add_argument("--learning_rate", type=float, default=1e-5)
    parser.add_argument("--patience", type=int, default=5)

    args = parser.parse_args()

    base_path = args.base_path
    model_type = args.model_type
    not_trainable = args.not_trainable
    epochs = int(args.epochs)
    learning_rate = args.learning_rate
    patience = int(args.patience)

    # Creem els generadors de dades
    img_shape, class_mode, batch_size, train_generator, validation_generator,
        num_classes = create_train_generators(base_path)

    # Engeguem l'entrenament
    train(model_type, not_trainable, epochs, learning_rate, patience)
```

Definició de generadors de dades de test

```
def create_test_generators(base_path="/home/vortiz/gesture/TFG_PROVES/"):
    '''Crea generadors de dades de test'''

    img_width, img_height = 424, 240
```

```
    img_shape = (img_height, img_width, 3)
    class_mode ='categorical'
    batch_size = 4

    no_train_setup = ImageDataGenerator(rescale=1./255,
        preprocessing_function=depth_preprocess)

    test_generator = no_train_setup.flow_from_directory(
        base_path + "IMG/test",
        target_size=img_shape[:-1],
        batch_size=batch_size,
        class_mode=class_mode,
        shuffle=False)

    num_classes = len(set(test_generator.classes))

    print("Generadors creats correctament")

    return img_shape, class_mode, batch_size, test_generator, num_classes
```

Funció de càrrega de models i definició de mètriques d'avaluació

```
def test(model_path):
    '''Avalua el millor model en el conjunt de test'''

    model = load_model(model_path)
    y_pred_soft = model.predict(test_generator, verbose=2)
    #print("Probabilitats:", y_pred_soft)
    y_pred = [np.argmax(elem) for elem in y_pred_soft]
    #print("Prediccions més probables:", y_pred)
    y_true = test_generator.classes
    #print("Etiquetes reals:", y_true)

    #test_loss = log_loss(y_true, y_pred_soft)
    test_acc = accuracy_score(y_true, y_pred)
    print("Precisió en test: %.4f" % test_acc)
    print("-------------------------------------------------")

    target_names = ["G"+str(i) for i in range(num_classes)]
    report = classification_report(y_true, y_pred, target_names=target_names,
        digits=4)
    print(report)
    with open(base_path+'MODELS/classification_report_'+
            model_path.split("/")[-1][:-3]+'.txt', 'w') as f:
        f.write(report)

    cm = confusion_matrix(y_true, y_pred)
    cmpl = ConfusionMatrixDisplay(cm, display_labels=target_names)
    fig, ax = plt.subplots(figsize=(10,10))
    ax.set_title('Confusion matrix of '+model_path.split("/")[-1].split("_")[-2])
    cmpl.plot(ax=ax)
    fig.savefig(base_path+'MODELS/confusion_matrix_'+
            model_path.split("/")[-1][:-3]+'.png', dpi=fig.dpi)
```

Crida d'avaluació

```
# Execució de l'avaluació del model
if __name__ == "__main__":
    '''
    base_path = string(sys.argv[1])
    model_path = string(sys.argv[2])
    '''

    parser = argparse.ArgumentParser(description='Executable script to test models.
    Syntax: --base_path --model_path')
    parser.add_argument("--base_path", type=str,
        default="/home/vortiz/gesture/TFG_PROVES/")
    parser.add_argument("--model_path", type=str, default=
        "/home/vortiz/gesture/TFG_PROVES/MODELS/model_InceptionResNetV2_ultimate.h5")

    args = parser.parse_args()

    base_path = args.base_path
    model_path = args.model_path

    # Creem el generador de dades
    img_shape, class_mode, batch_size, test_generator, num_classes =
        create_test_generators(base_path)

    # Engeguem el test
    test(model_path)
```

Codi d'inferència

```
from PIL import Image
import os
import shutil
import sys
import math
import numpy as np
import cv2
import random
import glob
import argparse
import time

from keras.utils import np_utils
from tensorflow.keras.models import Model
from tensorflow.keras.models import Model, load_model, model_from_json
from tensorflow.python.framework.ops import disable_eager_execution

disable_eager_execution()

from numpy.random import seed; seed(42)
import tensorflow; tensorflow.random.set_seed(42)
import pyrealsense2 as rs

print("Num GPUs Available: ", len(tensorflow.config.list_physical_devices('GPU')))

#######################################
```

```
def inference(model_path):
    '''Fa inferència en la RealSense'''

    # Carreguem el model per a inferència
    model = load_model(model_path)

    # Defineix distància màxima de detecció i temps complet de retard per frame
    d_max_mm = 2550 #mm
    t_delay_full = 250 #ms

    # Inicialitza el pipeline de la càmera
    pipe = rs.pipeline()

    # Configura el corrent de dades de profunditat
    config = rs.config()
    config.enable_stream(rs.stream.depth, 424, 240, rs.format.z16, 5)
    profile = pipe.start(config)

    try:
        while True:
            # Mesura inicial de temps total d'inferència + visualització
            start = time.time()
            # Agafa una captura de la càmera de profunditat
            frames = pipe.wait_for_frames()
            depth_frame = frames.get_depth_frame()
            if not depth_frame: continue
            depth_image = np.asanyarray(depth_frame.get_data())
            # Esborra punts més profunds que d_max_mm
            depth_image[depth_image > d_max_mm] = 0
            # Calcula la imatge monocromàtica
            uint_img = np.round(np.absolute(np.array((depth_image/d_max_mm)*255)))
                .astype('uint8')
            # Redimensiona a 3 canals i estandarditza valors al rang 0-1
            stacked_img = np.stack((uint_img,)*3, axis=-1)
            stacked_img_std = stacked_img / 255.0
            # Passa pel model
            y_pred_soft = model.predict(np.expand_dims(stacked_img_std, axis=0))
            y_pred_text = "Gest detectat: G"+str([np.argmax(elem) for elem in
                y_pred_soft][0])
            # Posa el text a la imatge
            font, org, fontScale = cv2.FONT_HERSHEY_SIMPLEX, (25,25), 0.8
            fontColor, thickness, lineType = (227,0,181), 2, 2
            cv2.putText(stacked_img, y_pred_text, org, font, fontScale,
                fontColor, thickness, lineType)
            # Mostra la imatge
            cv2.imshow("Detector de gestos", stacked_img)
            # Mesura final de temps total d'inferència + visualització
            end = time.time()
            print("Temps total d'inferència i visualització: %.4f" %(end-start))
            # Temps per a mantindre uns FPS constants; si premem ESC, eixim del bucle
            key = cv2.waitKey(max(1, round(t_delay_full-1000*(end-start))))
            if key == 27: # Codi tecla ESC
                cv2.destroyAllWindows()
                break

    except Exception as e:
```

```
        print(e)
        pass

    finally:
        pipe.stop()


#########################################################

# Execució de la inferència en RealSense
if __name__ == "__main__":
    '''
    model_path = string(sys.argv[1])
    '''

    parser = argparse.ArgumentParser(description='Executable script to make inference
        from models. Syntax: --model_path')
    parser.add_argument("--model_path", type=str,
      default="/home/vortiz/gesture/TFG_PROVES/MODELS/model_DenseNet121_ultimate.h5")

    args = parser.parse_args()

    model_path = args.model_path

    # Engeguem la inferència
    inference(model_path)
```