UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

TREBALL FI DE GRAU

Grau en Enginyeria Electrònica Industrial i Automàtica

PROTOTIP D'UN DISPOSITIU PER AL RECONEIXEMENT AUTOMÀTIC D'IMATGES DE CÈL·LULES SANGUÍNIES

Memòria i Annex

Autor: Martí Reig Grau
Director: Edwin Santiago Alférez Baquero
Co-Director: José Julian Rodellar Benede

Convocatòria: 2018-2019

I. Resum

Aquest treball consisteix en el disseny i desenvolupament d'un prototip per al reconeixement de cèl·lules sanguínies, per a realitzar un diagnòstic temprà, i l'estudi de tots els conceptes teòrics necessaris per a la comprensió total del treball. Aquest procés comença en l'estudi del camp de la intel·ligència artificial dedicat a les xarxes neuronals i el món del *machine learning*, camp que en els darrers anys ha guanyat importància. Analitzant els diferents tipus de problemes possibles i la manera més adient d'atacar-los, entendre que són les xarxes neuronals i les metodologies que utilitzen. Un cop entesa la base teòrica, s'analitza un model de classificació de cèl·lules ja existent: que fa, com ho fa i els resultats obtinguts, així com el model preentrenat Xception utilitzat en aquest. Familiaritzats amb el model, es desenvolupa una aplicació web de caràcter local, en l'entorn de treball Dash, per a ordinadors o similars, capaç d'interactuar amb el model, presentant així una eina de treball per a personal mèdic per poder aprofitar tot el potencial del model d'una manera ràpida i eficaç. Finalment, l'aplicació s'integra dins d'una placa de desenvolupament prou potent per a operar models de xarxes neuronals amb una mida molt reduïda, esdevenint el prototip de dispositiu mèdic portàtil per a la classificació de cèl·lules.

II. Resumen

Este trabajo consiste en el diseño y desarrollo de un prototipo para el reconocimiento de células sanguíneas para realizar un diagnóstico temprano y el estudio de todos los conceptos teóricos necesarios para comprensión total del trabajo. Este proceso comienza en el estudio del campo de la inteligencia artificial dedicado a las redes neuronales y el mundo del *machine learning*, campo que en los últimos años ha ganado importancia. Se analizan los diferentes tipos de problemas posibles y la manera más adecuada de atacarlos, entender qué son las redes neuronales y las metodologías que utilizan. Una vez entendida la base teórica, se analiza un modelo de clasificación de células ya existente: qué hace, cómo lo hace y los resultados obtenidos, así como el modelo preentrenado Xception utilizado en este. Familiarizados con el modelo, se desarrolla una aplicación web de carácter local, en el entorno de trabajo Dash, para ordenadores o similares, capaz de interactuar con el modelo, presentando así una herramienta de trabajo para personal médico para poder aprovechar todo el potencial del modelo de una manera rápida y eficaz. Finalmente, la aplicación se integra dentro de una placa de desarrollo lo suficientemente potente como para operar modelos de redes neuronales con un tamaño muy reducido, convirtiéndose en el prototipo de dispositivo médico portátil para la clasificación de células.

III. Abstract

This work consists in the design and development of a prototype for the recognition of blood cells to perform an early diagnosis and the study of all the theoretical concepts necessary for total understanding of the work. This process begins with the study of the field of artificial intelligence devoted to neuronal networks and the world of machine learning, a field that has gained importance in recent years. Analyzing the different types of problems and the most appropriate way to solve them and understanding what neuronal networks are and what methodologies they use. Once the theoretical base is understood, an existing cell classification model is analyzed: what it does, how it does it and the results obtained, as well as the pre-trained Xception model used in it. Familiarized with the model, a web application of a local host, in Dash framework, for computers or similar, is developed, capable of interacting with the model, presenting a work tool for medical personnel to take advantage of the full potential of the model of 'a fast and effective way. Finally, the application is integrated into a development plate that is sufficiently powerful to operate neural network models with a very small size, becoming the prototype of a portable medical device for the classification of cells.

IV. Agraïments

M'agradaria donar les gràcies al Santiago Alférez per la seva dedicació i esforç amb el treball, per haver-me orientat sobretot el que havia d'estudiar i entendre i per estar a la meva disposició per a qualsevol dubte o problema.

Al meu germà per haver-me orientat i ajudat a entendre conceptes difícils i a la meva parella per haver estat al meu costat i ajudar-me a no decaure.

M'agradaria, a més, donar les gràcies a l'equip format per José Rodellar i Santiago Alférez per proporcionar-me els materials necessaris per a poder dur a terme el projecte i a la Dra. Anna Merino, del Laboratori Core del Hospital Clínic de Barcelona, per donar-nos accés a les imatges de frotis.

V. Glossari

***Machine leargning**:* aprenentatge autònom d'algoritmes.

Deep learning: aprenentatge autònom profund, complex.

Xarxa neuronal artificial: model computacional del machine learning.

Xarxa neuronal convolucional: model computacional del deep learning, s'enfoca a analitzar imatges.

Fine tunning: ajust fi, tècnica per a desenvolupar models.

***Inception**:* un tipus de model preentrenat.

***Xception**:* un tipus de model preentrenat.

***Depthwise**:* tècnica de càlcul utilitzada en les xarxes neuronals convolucionals

***Pointwise**:* tècnica de càlcul utilitzada en les xarxes neuronals convolucionals

Índex

I. RESUM 3

II. RESUMEN 4

III. ABSTRACT 5

IV. AGRAÏMENTS 6

V. GLOSSARI 7

1. PREFACI 13

1.1. Contingut de la memòria 13
1.2. Origen del treball 13
1.3. Motivació 13
1.4. Requeriments previs 13

2. INTRODUCCIÓ 15

2.1. Introducció 15
2.2. Objectius del treball 15
2.3. Abast del treball 16

3. BASES TEÒRIQUES 17

3.1. Estat de l'art del machine learning 17
3.2. Que és l'aprenentatge automàtic 18
3.3. Tipus d'aprenentatges 18
3.3.1. Aprenentatge supervisat 19
3.3.2. Aprenentatge no supervisat 19
3.3.3. Aprenentatge autosupervisat 20
3.3.4. Aprenentatge reforçat 20
3.4. Teoria de les xarxes neuronals artificials 20
3.4.1. Neurona biològica 20
3.4.2. Neurona artificial 21
3.4.3. Xarxa neuronal artificial 22
3.4.4. Xarxa neuronal convolucional 23

4. MODEL DE RECONEIXEMENT DE CÈL·LULES EN SANG 27

4.1. Bases biològiques 27

4.1.1. La sang 27
4.1.2. Els glòbuls blancs 28
4.2. Plantejament de l'algoritme de classificació 30
4.2.1. Què classificar, perquè i com 30
4.2.2. Adequació de les dades 30
4.2.3. Desequilibri de dades 31
4.2.4. Grups d'entrenament, validació i testeig 31
4.3. Estructura d'una xarxa neuronal 32
4.3.1. Procés de crear un model 32
4.3.2. Creació d'un model des de zero 32
4.3.3. Utilització d'un model preentrenat 33
4.3.4. Ajust fi 33
4.4. Model final de classificació d'imatges en cèl·lules sanguínies 34
4.4.1. Resultats del model creat des de zero 34
4.4.2. Resultats amb models preentrenats 34
4.4.3. Resultats amb ajust fi 35
4.4.4. Model Xception 36
4.4.5. Conclusions del model 38

5. HARDWARE I SOFTWARE DEL PROTOTIP 39
5.1. Hardware 39
5.2. Software 41

6. DISSENY I DESENVOLUPAMENT DE L'APLICACIÓ DASH 43
6.1. Introducció al Dash 43
6.2. Plantejament de l'aplicació 44
6.2.1. Interfície 44
6.2.2. Eines per a desenvolupar l'aplicació 45
6.3. Desenvolupament de l'aplicació 47
6.3.1. Descripció de l'aplicació 47
6.3.2. Llibreries necessàries 48
6.3.3. Ordres prèvies 49
6.3.4. Disseny de la interfície 49
6.3.5. Callbacks i Funcions 52
6.4. Resultats de l'aplicació 59

7. ANÀLISI DE L'IMPACTE AMBIENTAL 61

8. ANÀLISI ECONÒMICA 63

8.1. Material 63
8.2. Software 63
8.3. Formació 63
8.4. Costos d'enginyeria 64

9. CONCLUSIONS 65

9.1. Conclusions 65
9.2. Possibles millores 66

BIBLIOGRAFIA 67

Referències de la literatura 67
Referències en la xarxa 67
Software i Hardware 68
Bibliografia d'imatges 69

ANNEX 71

A1. Codi complet de l'aplicació Dash 71

1. Prefaci

1.1. Contingut de la memòria

En aquest primer capítol es defineix el contingut de la memòria, l'origen del treball i la seva motivació.

1.2. Origen del treball

En aquest treball es busca simplificar el reconeixement i classificació d'imatges de cèl·lules sanguínies, obtingudes amb un microscopi òptic, fent ús dels resultats de dos treballs anteriors: la tesi doctoral "*Methodology for Automatic Classification of Atypical Lymphoid Cells from Peripherial Blood Cell Images*", del grup d'investigació CodaLab, presentada pel Santiago Alférez Baquero l'any 2015 en el Programa d'Enginyeria Biomèdica de la Universitat Politècnica de Catalunya [1] i el treball de fi de grau "*Reconocimiento automático de cèl·lules malignas en sangre periférica a partir de imágenes digitales y utilizando redes neuronales convolucionales*" presentat per la Rebeca Villarraso Jiménez l'any 2018 en el Grau d'Enginyeria Biomèdica de la EEBE [2].

1.3. Motivació

Aquest treball està motivat pel desig d'augmentar l'eficàcia de la sanitat. La correcta implementació d'un algoritme de classificació en una placa portàtil pot suposar una gran millora a l'hora de realitzar anàlisis de frotis sanguínies, aconseguint observar i classificar les diferents mostres de manera més ràpida i eficaç. Es vol convertir la necessitat de tenir a personal mèdic analitzant una a una les imatges durant hores a requerir menys d'una hora per a fer-ho. Això suposaria menys personal, menys cansament i menys hores invertides que es poden invertir en altres feines més profitoses.

1.4. Requeriments previs

Per poder realitzar aquest treball va ser necessari aprendre prèviament coneixements de camps que no s'han estudiat al llarg de la carrera i que eren necessaris per a poder entendre i desenvolupar el projecte.

Entre aquests coneixements previs s'inclou entendre que volen dir els termes *machine learning* i *deep learning*. En concret va ser necessari entendre les xarxes neuronals artificials i les xarxes neuronals convolucionals.

Per adquirir aquests coneixements es va cercar informació principalment en el mateix treball de reconeixement de cèl·lules malignes [2] i en el llibre *"Deep Learning with Python"*, de François Chollet, publicat l'any 2017 per l'editorial MANNING [3], entre altres.

Aquest llibre també va ser utilitzat per aprendre les bases de Python i algunes de les seves llibreries com Numpy, Keras, Tensorflow, Pandas i la més important, ja que el treball gira entorn d'aquesta llibreria, Dash.

Per a estudiar a fons aquesta llibreria es va realitzar el curs en línia, de l'empresa Udemy, *"Interactive Python Dashboards with Plotly and Dash" [i]*.

Un cop obtinguts els coneixements per a poder realitzar l'aplicació es va haver d'instal·lar tots els programes, llibreries i sistema operatiu necessaris:

- Sistema operatiu: Ubuntu 16.04. Descarregat gratuïtament des de la pàgina oficial d'Ubuntu [xiii].
- Programa de desenvolupament Atom. Descarregat gratuïtament de la pàgina oficial d'Atom [xiv].
- Python 3.6.6. Descarregat de la pàgina oficial de Python [xvi].
- Plotly Dash.
- Tensorflow 1.9 [xviii].
- Keras 2.2 [xix].

Aquest treball ha sigut desenvolupat amb un ordinador portàtil amb les següents característiques:

- Equip: Acer Aspire VX5-591G-54FT
- Sistema operatiu: Ubuntu 16.04 64 bits
- Processador: Intel® CoreTM i7 – 7300HQ CPU @ 3,50GHz 6MB
- Memòria RAM: 8GB DDR4 SDRAM
- Targeta gràfica: Nvidia GeForce® GTX 1050 4GB GDDR5

Tot el software utilitzat és de lliure accés i gratuït.

2. Introducció

2.1. Introducció

Actualment, un dels objectius més importants del desenvolupament tecnològic és aconseguir realitzar tasques ja existents de manera més ràpida i/o senzilla. De tal forma que es puguin realitzar moltes més tasques en el mateix transcurs de temps o inclús poder arribar a reduir el nombre de personal necessari per a dur a terme certa tasca. En l'àmbit mèdic sol ser el primer cas, ja que al ser un camp de treball delicat, sempre es necessita personal mèdic per supervisar que no hi hagi possibles errors que podrien posar en perill la vida de pacients.

Quan a un pacient se sospita que pot patir càncer hematològic, en altres paraules, càncer que s'origina en el teixit que forma la sang com la medul·la òssia o les cèl·lules del sistema immunitari, una de les proves més comunes és realitzar una anàlisi visual de les cèl·lules presents en la sang. Això es realitza extraient una petita mostra de sang del pacient a analitzar i col·locant-la en un frotis per a ser observada seguidament a través d'un microscopi òptic. Un cop obtingudes les imatges del frotis, un metge especialitzat en detecció de cèl·lules sanguínies es disposa a analitzar-les una per una i classificar-les segons presentin cèl·lules canceroses, i de quin tipus, o no.

Per tal de facilitar aquesta feina al personal mèdic, en aquest treball es dissenyarà i desenvoluparà una aplicació web fàcil d'entendre i senzilla d'utilitzar que s'encarregui de realitzar una preclassificació de tal manera que aquest personal només hagi de comprovar i corregir els resultats obtinguts.

2.2. Objectius del treball

L'objectiu resumit és la realització d'un prototip d'un dispositiu portàtil per al reconeixement automàtic d'imatges de cèl·lules sanguínies a partir d'un model ja realitzat. Per aconseguir aquest objectiu se separa el treball en diferents parts, cada una amb els seus objectius específics corresponents.

En primer lloc, introduir-nos en el camp de l'aprenentatge automàtic (*machine learning*), les xarxes neuronals artificials i les xarxes neuronals convolucionals. Entendre com funcionen aquestes des del punt de vista conceptual i quines tècniques i eines de treball utilitzen per a poder generar models de predicció capaços d'aprendre pel seu compte.

En segon lloc, entendre el model de classificació de cèl·lules malignes en sang, realitzat per la Rebeca Villarraso [2]. Aprendre què classifica i com des del punt biològic, com fa servir les xarxes neuronals convolucionals per realitzar aquesta classificació i el procés que s'ha realitzat fins a aconseguir un model capaç de realitzar aquesta classificació amb un rendiment òptim.

En tercer lloc, dissenyar i desenvolupar una aplicació web de caràcter local que, fent ús del model anterior, permeti a personal mèdic realitzar una classificació d'imatges de cèl·lules de frotis de sang de manera molt més ràpida i eficient.

Finalment, seleccionar una placa de desenvolupament capaç de corre el model de predicció de manera ràpida i òptima, implementar tant l'aplicació com el model en aquesta i avaluar el correcte funcionament de totes les parts (model, aplicació i placa) en conjunt.

2.3. Abast del treball

Les cèl·lules de la sang es poden diferenciar amb una anàlisi visual. Fent ús de la visió artificial es poden diferenciar les cèl·lules amb una morfologia característica que pot indicar la presència d'algun tipus de patologia en el pacient, de manera que es minimitza la necessitat de tenir personal mèdic realitzant aquesta anàlisi. Fent ús d'aquesta tecnologia es pot realitzar un model de xarxa neuronal convolucional capaç de classificar un gran nombre d'imatges de cèl·lules en sang en 4 categories diferents amb una precisió acceptable. Ara bé, cal desenvolupar una eina que permeti a aquest personal treballar amb ell.

Utilitzant l'entorn de treball Dash de Plotly es pot realitzar una aplicació web de caràcter local que permeti a aquest personal interactuar amb el model de manera còmoda i ràpida.

A més, actualment hi ha plaques de desenvolupament prou potents per a poder corre xarxes neuronals, pel que es pot integrar l'aplicació i el model en elles. Aconseguint de resultat un prototip d'estació mèdica portàtil per a l'anàlisi de cèl·lules sanguínies.

3. Bases teòriques

Aquest primer capítol consisteix en una breu introducció a la teoria fonamental del treball com entendre què és el *machine learning* així com les seves categories més importants, amb alguns exemples, i l'esquelet que l'ha fet possible.

3.1. Estat de l'art del machine learning

El *machine learning* ha anat guanyant força en els darrers anys en diversos camps. En aquests, el seu ús ha significat millorar els serveis aportats d'una manera que no hauria sigut possible humanament. És a dir, que per molt personal que s'incorporés no es podria arribar a realitzar la feina que fan aquests algoritmes.

Entre aquests camps podem trobar aspectes del nostre dia a dia com serien els cercadors, en els quals observen les paraules introduïdes i, en funció a la següent acció que fa l'individu, aquest algoritme aprèn. Si la següent acció realitzada és anar al primer enllaç, aportat pel cercador, significa que la cerca ha sigut satisfactòria. Per altra banda, si després, l'individu torna a realitzar una cerca, amb diferents paraules o afegint-ne de noves, significa que el cercador no ha mostrat resultats que volia aquest usuari i aleshores, l'algoritme aprèn del seu error [ii].

Un altre camp, del qual forma part aquest treball, seria el camp mèdic, en el que analitza imatges mèdiques de pacients i reconeix patrons que l'ull humà no podria o simplement reconeix imatges que un humà podria però a una velocitat per imatge molt més elevada que aquests. A més, s'utilitza per calcular els factors de risc de malalties en la població en general [ii].

Ara bé, el camp on el *machine learning* pren més força de tots és la mineria de dades, de l'anglès *Data mining*. La mineria de dades és el procés d'extraure informació significativa de grans bases de dades, dades que poden aportar informació a través de factors ocults. Prediu la situació futura de l'empresa, cosa que ajuda a aquesta a prendre millors decisions, proporcionant un avantatge competitiu. Un exemple seria el seu ús en els comerços locals, en els quals analitzen els patrons de compra conjunta per identificar associacions de productes i decidir com col·locar-los en els estants dels passadissos [ii], [5].

Altres camps on s'aplica el *machine learning* són detecció de programes maliciosos, seguretat personal, detecció de fraus i l'actual punt de mira, cotxes autopilotables, entre molts més [ii].

3.2. Que és l'aprenentatge automàtic

Aprenentatge automàtic, de l'anglès *machine learning*, és un camp de la intel·ligència artificial que, mitjançant complexos algoritmes, permet a les màquines aprendre. Aquest aprenentatge s'aconsegueix entregant al programa una quantitat elevada de dades que poden anar acompanyades de classificacions d'aquestes o no. La màquina sola trobarà patrons que relacionin les dades amb les classificacions que se'ls hi ha donat o buscarà patrons ocults que puguin agrupar les dades en categories sense nom i elaborarà una sèrie de regles que permetran al mateix algoritme determinar la classificació o agrupació de nova informació que rebi. Aquest procés s'anomena entrenament [3], [5].

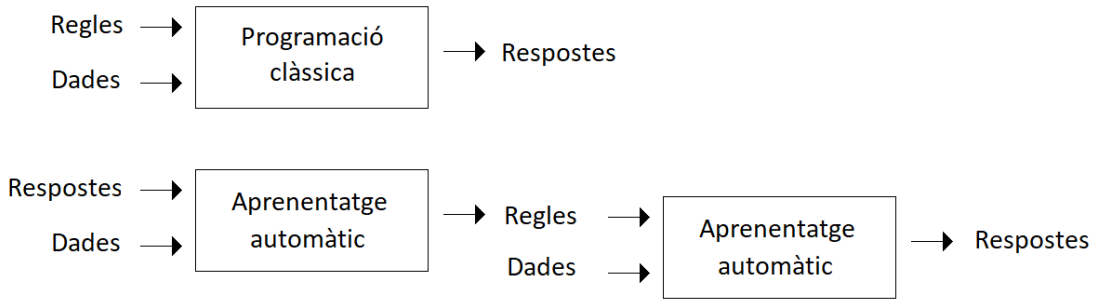

Figura 3.1.- Simplificació de la diferència entre la programació clàssica i l'aprenentatge automàtic. (Font: [3])

Un model entrenat seria aquell que ja se li ha entregat les dades, ha analitzat i ha arribat a generar una sèrie de regles per a un camp determinat. Per exemple, es podria obtenir un model entrenat capaç de reconèixer gats i gossos en imatges per a visió artificial però aquest no podria ser utilitzar per fer una predicció del trànsit [5].

Per evitar aquest problema, a l'hora de generar un model s'han de provar diferents funcions de predicció adients a la temàtica del problema a tractar; no s'utilitzen els mateixos algoritmes per models de reconeixement d'imatges que per models de classificació o predicció. Dins de cada temàtica inclús hi ha diferents, i el procediment és anar provant cada un d'ells amb les seves petites variacions o modificacions i observar la seva exactitud, de l'anglès *accuracy*, quedant-nos amb el més exacte [2].

3.3. Tipus d'aprenentatges

El *machine learning* és un camp molt complex amb multitud de subcamps dins seu. Els seus algoritmes es poden catalogar, generalment, en dues grans categories, que són l'aprenentatge supervisat i l'aprenentatge no supervisat. Encara que hi ha més categories, de menor presència actualment, com l'aprenentatge auto supervisat i l'aprenentatge reforçat. A continuació es farà un breu resum de cada tipus [3].

3.3.1. Aprenentatge supervisat

L'aprenentatge supervisat és el tipus d'aprenentatge més utilitzat de tots gràcies a la seva naturalesa clara i concisa. Sol ser el tipus d'aprenentatge que encaixa més amb els problemes als quals s'aplica el *machine learning* en l'actualitat. En un futur, quan els altres tipus guanyin popularitat i descobrim usos per aquests que encara no ens imaginem la balança s'equilibrarà més però de moment el *machine learning* consisteix en la seva majoria de problemes supervisats [3], [4].

Dins de l'aprenentatge supervisat hi ha dos grans tipus de problemes: els problemes de classificació i els problemes de regressió.

Els problemes de classificació i regressió consisteixen a entrenar un model a partir d'un volum de dades d'entrada gran amb les seves etiquetes o classificacions pertinents, anomenat conjunt d'entrenament, de tal manera que, al conèixer les entrades i les sortides que aquestes han de tenir, pot generar la sèrie de regles fàcilment. A continuació, quan ja s'han generat les regles, se li entrega un altre gran volum de dades d'entrada, amb les seves pertinents classificacions també, perquè comprovi l'eficàcia de les regles que ha creat. Aquest és el conjunt de validació. Tanmateix, el model no utilitzarà aquestes etiquetes més que per contrastar el resultat que ha obtingut aplicant les seves regles i el resultat que havien de tenir [3], [4].

La diferència que hi ha entre classificació i regressió és que un és de possibles respostes discretes i l'altre de possibles respostes contínues. En altres paraules, els problemes de classificació són aquells tal que els resultats possibles formen part d'un conjunt de respostes possibles i no hi ha possibilitat d'una resposta entremig, un exemple seria un problema per reconèixer matrícules. Per altra banda, els problemes de regressió són aquells tal que les respostes poden prendre infinitat de valors dins d'un interval, un exemple seria la predicció de la temperatura, que sempre pot haver-hi un valor més ínfim [3].

3.3.2. Aprenentatge no supervisat

L'aprenentatge no supervisat és un tipus d'aprenentatge que busca poder agrupar o relacionar grans volums de dades que no aporten cap informació sobre quines categories o etiquetes han de tenir. Es diferència de l'aprenentatge supervisat en què no busca les condicions que ha de complir una dada per formar part d'una categoria o d'una altra, ja que les dades d'entrada no tenen possibles categories. Aleshores, els algoritmes han de trobar la manera de poder agrupar aquest volum de dades basant-se en relacions o patrons comuns que hi pugui haver entre aquestes [3].

El mètode més utilitzat per a tractar aquest tipus de problemes és la clusterització de dades, de l'anglès *cluster analysis* [3].

El *clúster analysis* consisteix a realitzar aquestes agrupacions basant-se en diferents factors com la densitat o la distància entre les observacions en l'espai de dades, entre altres. Aquest espai és la representació visual del pes o valor de les dades d'entrada que es genera a partir d'aplicar certes fórmules o mètodes de valoració [3].

3.3.3. Aprenentatge autosupervisat

L'aprenentatge autosupervisat és molt semblant al supervisat. La diferència és que les classes o etiquetes de les dades no les introdueix una persona, sinó que és el mateix algoritme el que inventa aquestes classes. Aquesta categoria se situa entre les dues primeres comentades. No obstant això, no hi ha una frontera fixa entre aquestes tres categories pel que pot ser difícil distingir-les. Un aprenentatge autosupervisat pot ser reinterpretat com a supervisat o com a no supervisat, depèn del cas [3].

3.3.4. Aprenentatge reforçat

L'aprenentatge reforçat consisteix a generar un model sense dades d'entrada i basant-se en un sistema de recompenses de manera que l'algoritme sol aprengui a base de moltes iteracions. Un exemple aplicat seria un algoritme que aprengui a jugar al *Pong*. L'algoritme realitza accions aleatòries i aquestes poden aconseguir un premi, obtenir un punt, o una penalització, el contrincant obté un punt. Aquest tipus d'aprenentatge encara està en estat molt experimental i no té usos reals encara. Ara bé, es tenen grans expectatives per aquesta categoria en camps com robòtica i vehicles autònoms entre altres [3].

3.4. Teoria de les xarxes neuronals artificials

Fins ara s'ha explicat què és el *machine learning*, les categories que componen aquest i alguns exemples d'aquestes però falta entendre com és possible aplicar tota aquesta teoria a la pràctica. La tecnologia responsable de què el *machine learning* sigui possible és les xarxes neuronals artificials, de l'anglès *Neuronal Artificial Networks*.

Per a poder entendre què és una xarxa neuronal artificial s'ha d'entendre prèviament la idea d'una neurona artificial.

3.4.1. Neurona biològica

Una neurona és una cèl·lula del cos humà, concretament del sistema nerviós, que s'encarrega de rebre i processar informació d'entrada i enviar una resposta en funció d'aquesta.

Aquestes neurones reben informació d'altres neurones a través de les dendrites. A continuació, suma totes aquestes entrades i, si el resultat és superior a un llindar, envia un senyal a les altres neurones connectades a aquesta a través de l'axó [iii].

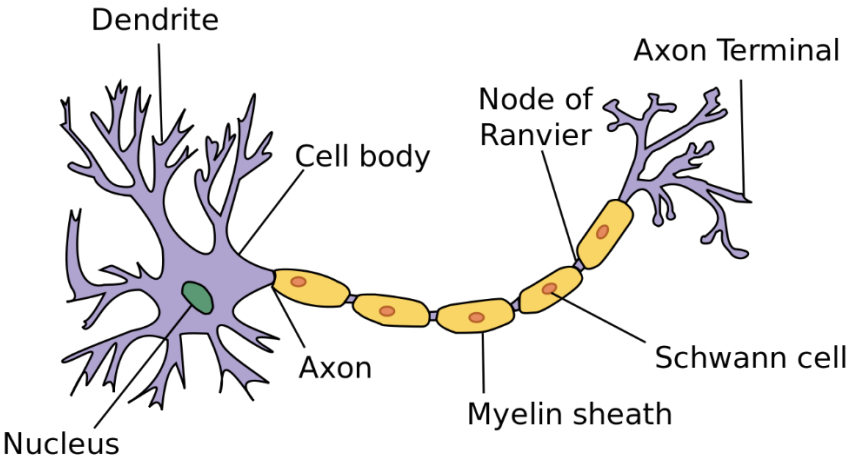

Figura 3.2.- Model de neurona cerebral.

(Font: [iii])

3.4.2. Neurona artificial

Una neurona artificial és un model de funció matemàtica basat en el funcionament de les neurones cerebrals. La base del seu funcionament és el mateix, rep informació d'entrada a través del perceptró, que seria l'equivalent a les dendrites i, si la suma de cada entrada multiplicada pel seu pes supera el valor de llindar, enviarà una resposta [3], [iii].

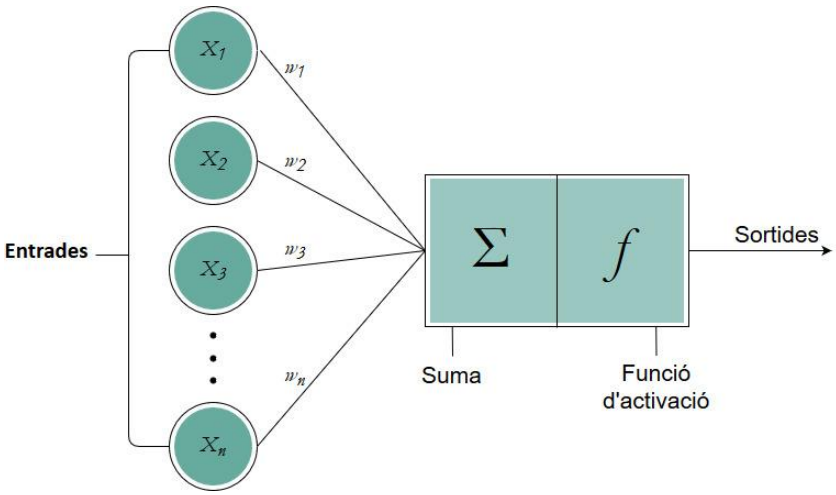

Figura 3.3.- Model de neurona artificial.

(Font: [iii])

Totes les entrades estan associades a un pes individual que pot ser diferent per a cada una.

3.4.3. Xarxa neuronal artificial

D'igual manera que un cervell humà fa ús de moltes neurones connectades entre elles per processar estímuls i generar respostes que seran enviades a la resta del cos, una xarxa neuronal artificial fa ús de moltes neurones artificials per tal de poder processar informació d'entrada per poder generar una resposta en funció a aquesta [iii], [ix].

Les neurones es connecten en forma de capes de manera no cíclica, de manera que les sortides d'unes poden ser les entrades d'unes altres. Normalment, per a problemes de xarxes neuronals artificials el model més comú emprat és el de capes completament connectades [ix].

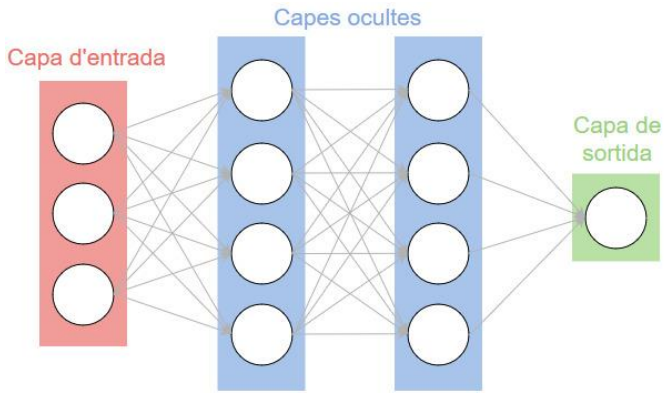

Figura 3.4.- Exemple de model neuronal de capes completament connectades.
(Font: [ix])

Prenent com a exemple el model de la Figura 3.4, s'observa que aquesta xarxa neuronal es compon de 3 capes que a la vegada estan formades per diverses neurones. Les neurones de cada capa estan connectades amb totes les neurones de les capes adjacents a aquesta.

El terme per referir-se al nombre de capes del que està compost un model de xarxa neuronal és el de profunditat, de l'anglès *Depth* [3].

A l'hora de crear un model s'ha de tenir en compte el nombre de paràmetres que haurà d'aprendre. Prenent una altra vegada d'exemple la Figura 3.4, per calcular el nombre de paràmetres que pot aprendre el nostre model s'ha de sumar el nombre de connexions entre les neurones de les capes, excloent les connexions d'entrada, i seguidament sumar un biaix. Aleshores tindríem que el nostre model és capaç d'aprendre 26 paràmetres, 20 (3 * 4 + 4 * 2) més 6 (4 + 2) [2].

Hi ha problemes que requereixen models amb més capes per a poder ser tractats correctament. Aleshores estem parlant d'aprenentatge profund, de l'anglès *deep Learning*. Aquest és un subgrup del *machine learning* que fa referència únicament als models de xarxes neuronals artificials formats per moltes capes. En altres paraules, que són molt profunds, d'aquí el seu nom [3].

Les xarxes neuronals artificials s'estructuren en tres tipus de capes:

Capa d'entrada: S'encarrega de transmetre la informació de l'exterior (entrada de dades) a la xarxa. No realitza cap càlcul computacional en ella, només passa la informació [iv], [2].

Capes ocultes: El seu nom es deu al fet que no tenen contacte amb l'exterior de la xarxa neuronal. S'encarreguen de processar la informació rebuda per la capa d'entrada i entreguen els resultats a la capa de sortida [iv], [2].

Capa de sortida: Aquesta és l'última capa totes. S'encarrega d'analitzar la informació rebuda per les capes ocultes i entregar una resposta al món exterior a la xarxa [iv], [2].

3.4.4. Xarxa neuronal convolucional

Les xarxes neuronals convolucionals formen part del deep learning, es caracteritzen per estar formades per moltes capes que poden ser disposades en forma de matriu bidimensional. Això fa que es pugui augmentar en molt la quantitat de paràmetres que pot aprendre un model.

Aprofitant aquesta característica, l'ús més comú de les xarxes neuronals convolucionals és el processament d'imatges per visió artificial. Es necessiten moltes capes amb moltes neurones per poder analitzar una imatge, ja que aquestes poden arribar a aportar milions de paràmetres a analitzar al tenir informació de cada píxel. Si tenim, per exemple, una imatge de 1000 × 1000 píxels amb els 3 colors ens sortirien 3 milions de paràmetres a aprendre.

Ara bé, un podria preguntar-se: "No podríem, simplement, afegir més capes a la xarxa neuronal artificial i/o augmentar el nombre de neurones en cada una?". La resposta és no, això ens portaria dos problemes: un problema és que no tenim capacitat computacional il·limitada com per operar amb aquests tipus de xarxes de manera massiva, l'altre problema és l'aparició del sobre ajust, de l'anglès *overfitting*. El sobre ajust és un efecte que apareix a l'entrenar massa un model o a l'entrenar-lo amb dades estranyes que no guarden gaire relació amb la funció objectiu que idealment volem aconseguir [3].

A part d'utilitzar capes bidimensionals, les xarxes neuronals convolucionals utilitzen diferents tipus d'agrupacions de capes per tal de crear un conjunt que s'encarregui de realitzar una tasca en concret, de manera que cada grup és format per diverses capes bidimensionals. Per a tractar amb els problemes de visió artificial utilitzen típicament tres tipus diferents de capes (sense contar les capes d'entrada i sortida) [v]:

Capes convolucionals: Les neurones convolucionals es caracteritzen per analitzar la imatge en forma de matriu en comptes d'individualment. En altres paraules, cada neurona s'encarrega d'observar una zona de pocs píxels quadrats de la imatge, així fan les altres neurones per a les altres zones de la imatge. D'aquesta manera s'aconsegueix reduir la càrrega computacional considerablement. Aprofitant aquesta virtut, les neurones determinen característiques d'una zona de la imatge com podria ser la detecció de vores. No obstant això, cal comentar que els nuclis convolucionals solen estar més entrenats de manera més complexa per tal d'extraure característiques més abstractes i no tan trivials [2], [iv] [v].

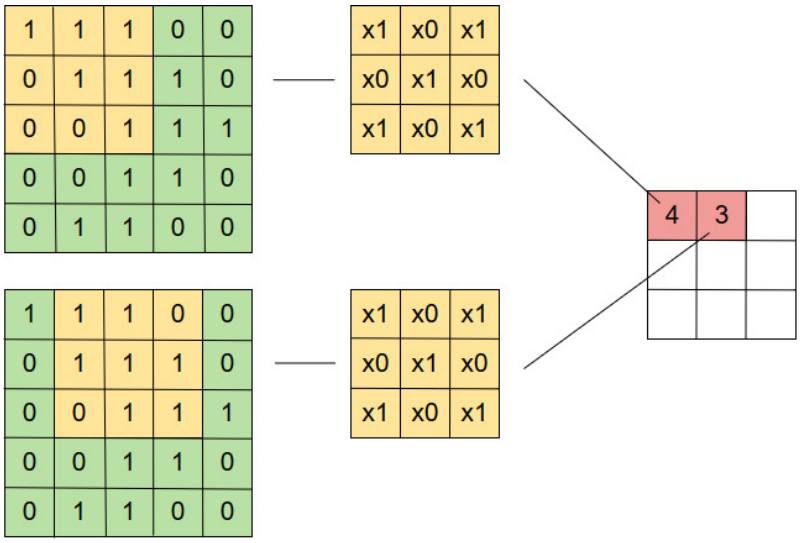

Figura 3.5.- Exemple de convolució on s'analitza una imatge 5 × 5 amb un escombrat 3 × 3. (Font: [v])

Capes d'agrupament: Les capes d'agrupament, de l'anglès *Pool layer*, s'encarreguen de reduir la dimensió d'informació rebuda per les capes convolucionals. Això ho fan combinant la resposta entregada per aquestes en una sola neurona. Les capes d'agrupament poden processar la informació de les capes convolucionals, entre altres maneres, utilitzant màxims i mitjanes. És a dir, es quedaran amb les característiques de més pes o amb les més reconegudes [v], [ix].

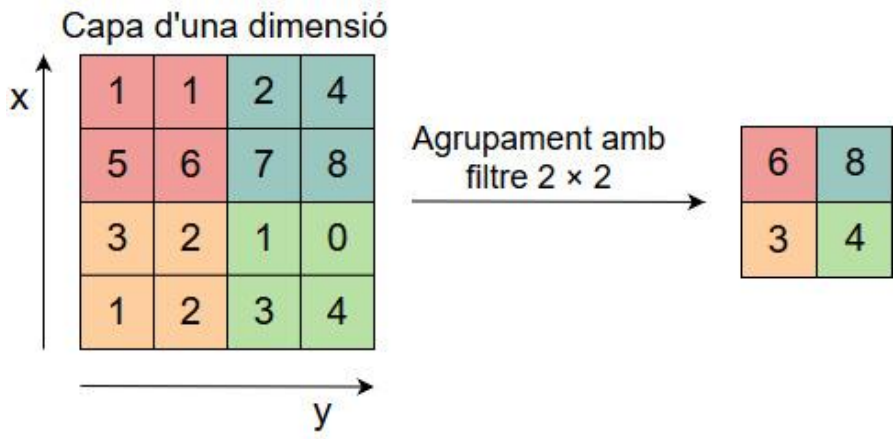

Figura 3.6.- Exemple de reducció de mida mitjançant un agrupament 2 × 2.
(Font*:* [ix])

Capes de regularització: Les capes de regularització s'encarreguen de controlar el sobre ajust per tal de reduir els seus efectes. Un exemple seria la regularització *dropout*, que apaga algunes neurones a l'atzar de manera que no aprenguin [2], [v].

Capes completament connectades: Al final de les capes d'agrupament es converteix la informació, que està disposada de forma bidimensional, a un vector d'una dimensió i a continuació s'analitza com es fa en les xarxes neuronals artificials, amb capes de neurones connectades a totes les neurones de les capes adjacents. Les capes completament connectades són les últimes prèvies a la capa de sortida [v].

Entremig d'aquestes capes es poden fer servir funcions d'activació, que actuen com a filtres que entreguen una resposta o no en funció a l'entrada. Una d'aquestes podria ser la ReLU, un rectificador que entrega resposta sempre que l'entrada sigui positiva, en cas contrari entregarà un 0 [2].

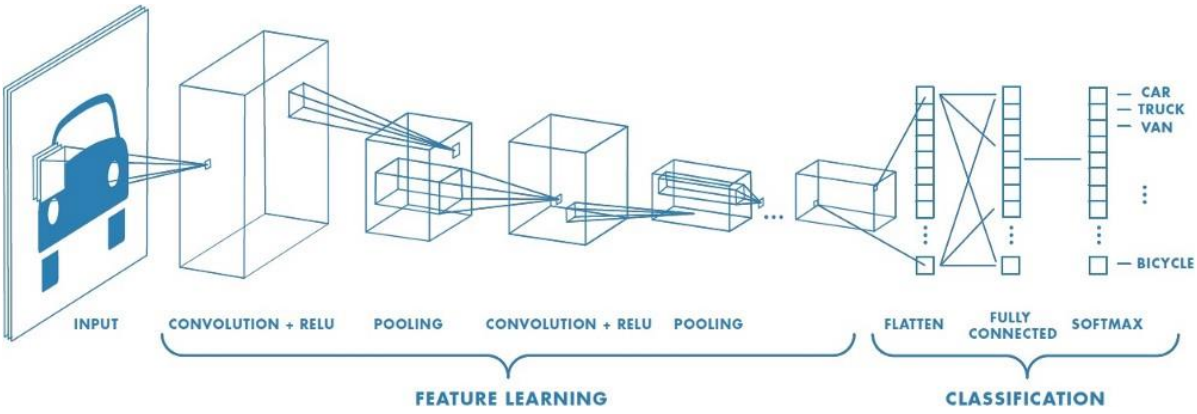

Figura 3.7.- Exemple de model convolucional amb la seva disposició en blocs i capes.
(Font: [v])

Si observem l'exemple de model convolucional que ens presenta la Figura 3.8, podem observar el procés d'aquest. En un principi tenim una imatge d'entrada que a través de la capa d'entrada arriba de forma òptima per a treballar a les capes convolucionals. Després de les capes convolucionals hi ha les capes *pool*, que simplifiquen la informació entregada per aquestes. Es pot observar que després de la primera capa d'agrupament hi ha un altre de convolucional. Això és degut al fet que els models convolucionals habituals són més complexes que utilitzar un tipus de capa i ja està, utilitzen diverses capes intercalades de manera que el model en concret realitzi una funció en concret. De manera que podem trobar diverses estructures convolucionals al mercat.

4. Model de reconeixement de cèl·lules en sang

En aquest capítol es responen a preguntes fonamentals com: què classifica, perquè ho classifica i com ho fa.

En primer lloc entendrem la idea darrere del model de reconeixement de cèl·lules sanguines des del punt de vista biològic i en segon lloc s'explicà la realització del model de xarxa neuronal capaç de dur a terme aquesta classificació així com les seves característiques més importants.

4.1. Bases biològiques

4.1.1. La sang

La sang humana està formada per plasma i cèl·lules, aquestes poden ser glòbuls vermells (eritròcits), glòbuls blancs (leucòcits) o plaquetes (trombòcits). Aquestes cèl·lules s'originen a partir d'una cèl·lula mare (hemocitoblast) que varia la seva estructura fins a arribar a convertir-se en una d'aquestes cèl·lules a través del procés anomenat hematopoesis. Una cèl·lula immadura es diu Blast [2].

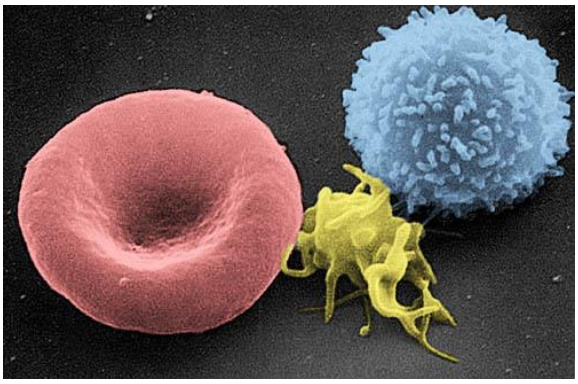

Figura 4.1.- D'esquerra a dreta: Glòbul vermell, plaqueta i glòbul blanc.
(Font: [xxii])

A l'hora de madurar, pot fer-ho per dues vies diferents: la via mieloide, que deriva a glòbuls vermells, plaquetes i alguns tipus de glòbuls blancs i la via limfoide, que deriva als altres tipus de glòbuls blancs i a les cèl·lules NK (de l'anglès *natural killer*) [2], [vi].

En aquest procés de maduració pot passar que les cèl·lules no es formin correctament i resultin en cèl·lules disfuncionals o que no es formin les suficients, això pot suposar un problema, ja que pot derivar en patologies i malalties perjudicials per a l'individu. Per exemple, una presència insuficient de glòbuls vermells pot derivar a anèmia, mentre que una malformació de les plaquetes pot derivar a problemes de coagulació [2].

En concret, aquest model se centra en la detecció i classificació dels glòbuls blancs, dels quals parlarem a continuació.

4.1.2. Els glòbuls blancs

Els glòbuls blancs són les cèl·lules encarregades de la defensa de l'organisme davant diferents amenaces: bacteris, virus, cèl·lules malignes, etc.

Els glòbuls blancs es cataloguen en dos subgrups: els granulats i els no granulats. Aquests subgrups fan referència a la seva morfologia. Els granulats presenten petits gràsnuls dins de les seves membranes cel·lulars mentre que els no granulats no en presenten [2], [vi].

El subgrup granulat està comprès per tres tipus de glòbuls: Eosinòfils, Neutròfils i Basòfils.

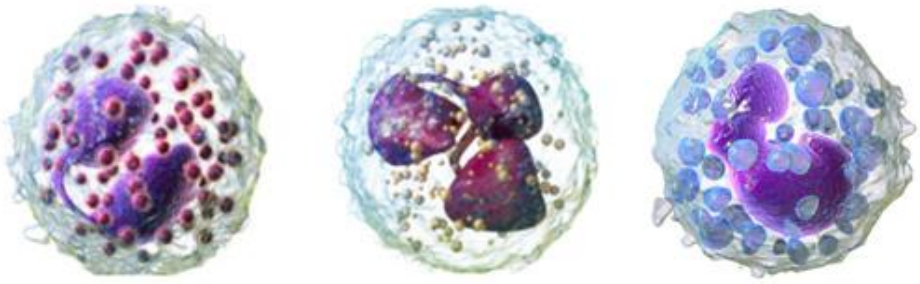

Figura 4.2.- Exemple dels tres tipus de glòbuls blancs granulats.
D'esquerra a dreta: Eosinòfil, Neutròfil i Basòfil.
(Font: [vi])

Per altra banda, el subgrup no granulat està compres per dos tipus: Monòcits i Leucòcits. Aquests últims a la vegada es poden diferenciar en Leucòcits tipus B i Leucòcits tipus T.

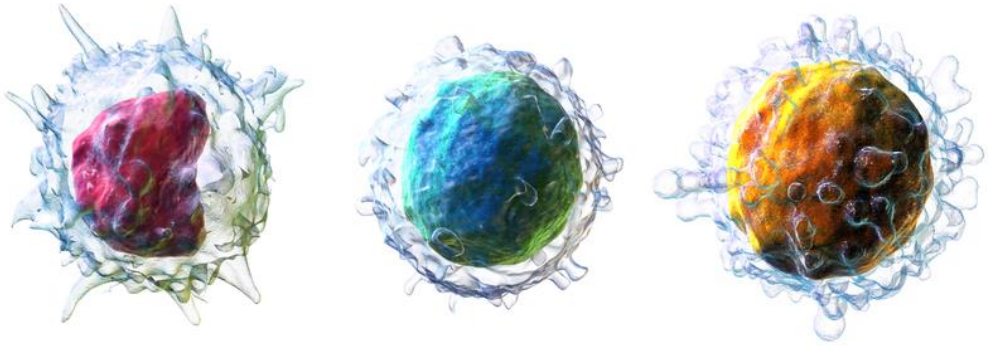

Figura 4.3.- Exemple dels tres tipus de glòbuls blancs no granulats.
D'esquerra a dreta: Monòcit, Leucòcit tipus B i Leucòcit tipus T.
(Font: [v])

Com s'ha comentat anteriorment, la maduració d'un glòbul blanc es pot fer per les dues vies. La via mieloide deriva en un dels tres tipus de glòbuls blancs granulats o en monòcits, mentre que la via limfoide deriva als limfòcits.

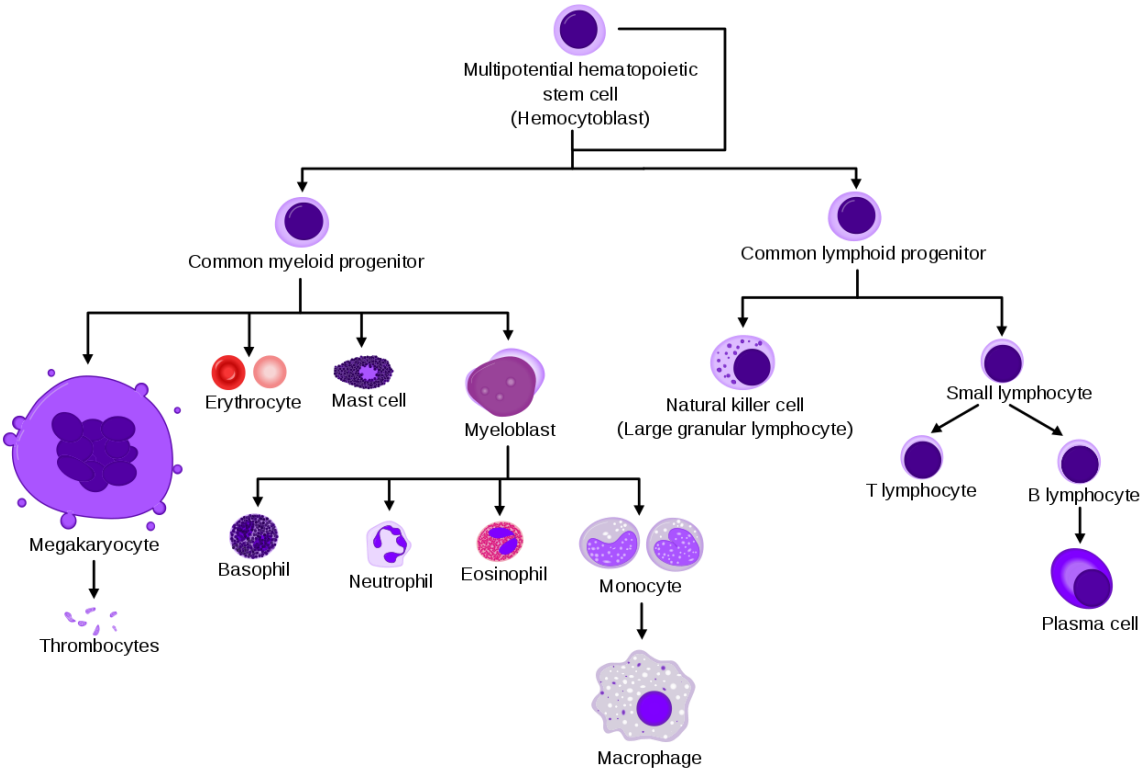

Figura 4.4.- Procés de maduració de les cèl·lules, Hematopoesis.
(Font: [xxiii])

4.2. Plantejament de l'algoritme de classificació

4.2.1. Què classificar, perquè i com

Com ja sabem, una malformació o formació insuficient d'alguna d'aquestes cèl·lules pot arribar a causar problemes de salut. Una forma de comprovar o confirmar la presència d'algun problema pot ser realitzant una anàlisi visual de la composició i morfologia cel·lular de la sang a través d'un frotis. S'utilitzarà aquesta tècnica juntament amb la tecnologia de les xarxes neuronals i visió artificial per a poder realitzar aquesta comprovació visual de manera automàtica, ràpida i eficaç [2].

Per determinar què classificar es va contactar amb la doctora Anna Merino, del laboratori Core de l'Hospital Clínic de Barcelona. La qual ens va entregar un conjunt de dades extens format per imatges de frotis de sang associada a 18 patologies diferents relacionades amb problemes de limfòcits. Tenint així imatges de limfòcits normals, limfòcits anormals, limfòcits reactius i blasts. Per tant, aquest es tractarà d'un model d'aprenentatge supervisat.

No es buscarà aconseguir classificar imatges de frotis amb presència de limfòcits en cada una de les 18 patologies entregades sinó que es buscarà classificar-les en una de les 4 categories anteriors:

- Limfòcits normals
- Limfòcits anormals
- Limfòcits reactius
- Blasts

Aquesta classificació en 4 grups ens permet realitzar un diagnòstic temprà de possibles patologies associades a la presència en sang de limfòcits anormals, com és el cas de leufemas; limfòcits reactius, com és el cas d'infeccions; o blasts, indicadors d'anèmies agudes [2].

4.2.2. Adequació de les dades

Les xarxes neuronals utilitzen un gran nombre de dades d'entrada per entrenar-se. Ara bé, a vegades no es pot simplement entregar les dades directament, sinó que s'han de processar i preparar perquè siguin més còmodes de treballar amb.

En aquest cas, les imatges tenen una grandària inicial de 1200 × 900 píxels i gran part d'ella no aporta informació sobre la cèl·lula. Això s'interpreta com a soroll i una forma de solucionar-ho és retallant les imatges a, per exemple, una nova mida de 900 × 900 píxels.

4.2.3. Desequilibri de dades

Un altre dificultat típica que ens apareix en els problemes de *machine* i *deep learning* és el desequilibri de dades.

El desequilibri de dades és un fenomen perjudicial que ens apareix quan en la nostra base de dades d'entrada no hi ha una distribució equitativa entre les diferents categories a les quals es pot classificar fent que el model de classificació acabi tendint cap al grup majoritari [2], [3]. En aquest cas passa que tenim una base de dades de 9780 imatges en total que no estan repartides a parts iguals en els 4 grups que hem determinat. En el grup de limfòcits anormals tenim 6825 imatges, dues terceres parts del total.

Per arreglar els problemes de desequilibri de dades en xarxes neuronals s'utilitzen diferents tècniques. Entre elles trobem: augmentar el nombre de dades d'entrada de les classes minoritàries, disminuir el nombre de dades de les classes majoritàries o aconseguir generar un algoritme que no es vegi afectat pels problemes de desequilibri de dades [2].

4.2.4. Grups d'entrenament, validació i testeig

En els casos supervisats, que tenim les dades d'entrada i coneixem les seves sortides, se sol seguir un procediment per tal d'entrenar el model de forma més efectiva. Se separen les dades d'entrada en 3 grups diferents i s'utilitzaran les dades de cada grup en diferents etapes de l'aprenentatge. Aquests grups són:

Conjunt d'entrenament: El grup d'entrenament és aquell del qual s'agafen les dades amb les seves classificacions per tal de determinar quines característiques o particularitats relacionen les entrades amb les sortides. En altres paraules, és el conjunt de dades que el model observa com a exemple. A l'hora de separar les dades d'entrada el conjunt d'entrenament resulta el més voluminós dels tres [2].

Conjunt de validació: El conjunt de validació s'utilitza quan el model ja ha observat el conjunt d'entrenament i ja ha pensat les regles que relacionen les entrades amb les sortides. S'utilitza aquest conjunt per a comprovar que les regles generades són encertades. El model agafa les entrades de dades (sense les seves classificacions) i els hi aplica les regles per determinar a quina classe pertanyen, a continuació es contrasten els resultats obtinguts amb les seves classificacions reals de tal manera que, quan el resultat obtingut no és el correcte, s'ajusta el model per tal de corregir els errors del model. D'aquesta manera se l'ajusta, fent-lo més precís [2].

Conjunt de testeig: El grup de testeig segueix la mateixa dinàmica que el grup de validació encara que, per contra, no contrasta els resultats obtinguts amb els reals per ajustar el model. Simplement compara els resultats obtinguts amb els reals de tal forma que es té una estimació del rendiment o qualitat del model [2].

4.3. Estructura d'una xarxa neuronal

Un cop ja sabem què és el que ha de fer el model i hem solucionat problemes de desequilibri de dades i sobre ajust i hem separat el conjunt de dades en els tres blocs necessaris podem començar a treballar el model en si.

4.3.1. Procés de crear un model

Encara que el resultat pugui semblar breu o simple, el procés de crear un model vàlid pot arribar a ser llarg i fatigós. Hi ha moltes maneres diferents de realitzar-ne un i és molt recomanable provar-les totes per tal de trobar la més apta. A més, es pot variar la dinàmica alterant certs paràmetres o utilitzant diferents components, pel que encara tenim més possibilitat.

A l'hora de crear el model de classificació de cèl·lules es va seguir un procés additiu on es va partir del més simple i es van anant afegint propietats o es van provar diverses alternatives. L'ordre que es va seguir és el següent:

1. Creació d'un model des de zero.
   1.1. Sense augment de dades.
   1.2. Amb augment de dades.
2. Creació d'un model a partir d'un model preentrenat
   2.1. Extracció de característiques sense guardar-les
   2.2. Extracció de característiques guardant-les
3. Millora del model amb Ajust fi.

A continuació s'explicarà cada pas amb la seva informació rellevant.

4.3.2. Creació d'un model des de zero

La creació d'un model des de zero consisteix en, com bé indica el nom, partir de res. Això vol dir que nosaltres indicarem l'estructura que tindrà la nostra xarxa i assignarem tots els paràmetres necessaris, així com el classificador a utilitzar [6].

Un cop s'ha determinat el cos, entrenarem l'algoritme amb el conjunt de dades del que disposem. Això es farà amb augment de dades si fos necessari. Aquest augment pot consistir a deformar, reorientar o variar la mida de les imatges que ja tenim per tal de crear imatges noves a partir d'elles.

És al entrenar-lo que aquest comença a generar els regles de determinació amb les que podrà diferenciar des de conceptes trivials com identificar fronteres fins a conceptes més abstractes. Aquestes regles s'aconsegueixen a l'assignar valors a un conjunt de pesos que es guardaran en tota la part convolucional com a característiques [6] ,[ix].

S'haurà creat un model en el moment en què aquest tingui tots els pesos i característiques amb uns valors capaços de diferenciar i percebre patrons i generar uns resultats que, a l'entregar-los, al classificador, també entrenat, ens podrà classificar les entrades de manera òptima. Aquest procés pot ser difícil de dur a terme.

4.3.3. Utilització d'un model preentrenat

Com s'ha comentat en l'apartat anterior, quan un model és entrenat té guardades les característiques capaces d'identificar patrons, de manera més o menys eficient, en la part convolucional.

Una forma de desenvolupar un model és utilitzant les característiques d'un model extern al teu. En altres paraules, en lloc d'entrenar un model des de zero arriscant-nos a obtenir un resultat no gaire precís importarem les característiques d'un model extern obert al públic que ha sigut entrenat amb una gran base d'imatges de manera que és molt eficient. Ara bé, aquestes característiques serveixen com a funció general així que sempre serà recomanable utilitzar diferents models externs per tal de trobar el més eficient pel problema a tractar [2], [ix].

Un cop s'han importat les característiques del model extern, i abans d'entrenar el model, es congelen les capes convolucionals per tal que no variïn les seves característiques. El que es farà és entrenar el classificador pel nostre problema en concret.

4.3.4. Ajust fi

Per tal de millorar el rendiment i l'eficàcia d'un model es pot aplicar la tècnica de l'ajust fi, de l'anglès *fine tunning*. El *fine tunning* consisteix a descongelar la capa o capes convolucionals més pròximes a la sortida de manera que puguin ajustar les característiques en lloc de només deixar ajustar el classificador [2].

4.4. Model final de classificació d'imatges en cèl·lules sanguínies

Al crear el model es van seguir els passos indicats a l'apartat 4.3.1. A continuació es farà un breu resum d'aquest procés i s'entrarà més en detall amb el model preentrenat *Xception*, el que va obtenir els millors resultats.

4.4.1. Resultats del model creat des de zero

El model creat es compon de 4 blocs. Els tres primers són els blocs convolucionals, cadascun format per 2 capes convolucionals seguides duna capa *maxpool* i una *dropout*. L'últim bloc és el de sortida i està format per una capa d'aplanament, una densa, un altre drop-out i finalment el classificador [2].

Taula 4.1- Millors resultats obtinguts en els experiments de models creats des de zero sense i amb augment de dades.
(Font: [2])

Amb augment de dades	Entrenament		Validació		Testeig		Optimitzador utilitzat
	Pèrdua	Precisió	Pèrdua	Precisió	Pèrdua	Precisió	
No	0.17	95.01	1.24	63.02	1.22	63.50	RMSProp
Sí	0.94	66.36	0.72	69.62	0.71	69.75	RMSProp

L'optimitzador s'encarrega d'actualitzar la xarxa en funció als resultats obtinguts en el procés de validació.

Observem que creant una xarxa neuronal des de zero no s'obtenen grans resultats. No obstant això, ens aporta una estimació de quin valor mínim de precisió hauran de tenir els models preentrenats.

4.4.2. Resultats amb models preentrenats

Per a dur a terme els experiments amb models preentrenats es van escollir bastants per tal de poder contrastar al màxim els resultats i poder escollir el millor. Com s'ha explicat abans, es van extreure les característiques d'aquests i se'ls va aplicar el classificador adient per al nostre problema.

Taula 4.2- Millors resultats obtinguts en els experiments de models preentrenats.
(Font: [2])

Model preentrenat	Entrenament		Validació		Testeig		Optimitzador utilitzat
	Pèrdua	Precisió	Pèrdua	Precisió	Pèrdua	Precisió	
VGG 16	0.10	98.30	0.50	81.00	0.57	79.17	RMSProp
VGG 19	0.13	97.72	0.54	79.00	0.60	76.67	RMSProp
ResNet 50	1.17	50.60	1.15	56.00	1.20	45.83	RMSProp
Inception V3	0.24	93.77	0.43	83.83	0.53	79.42	RMSProp
Xception	0.27	91.17	0.41	84.33	0.46	82.50	RMSProp
Inception V4	0.73	71.47	0.85	67.83	1.13	67.53	RMSProp

Aquesta taula fa referència a l'experiment en el qual es van guardar els pesos en el disc de manera que es poguessin reutilitzar. Ara bé, cal comentar que abans que aquest es va realitzar un altre experiment en el qual no es guardaven amb alguns models preentrenats més. Els models que en l'altre experiment van obtenir resultats molt dolents van ser exclosos pel següent experiment.

En general, els resultats són millors que els de l'experiment del model creat de zero, tret d'alguna excepció. De moment, el millor resultat obtingut és el del model *Xception*, amb un 82.5%.

4.4.3. Resultats amb ajust fi

Finalment, es va aplicar la tècnica de l'ajust fi als diferents models preentrenats. En general els resultats van millorar considerablement, encara que només alguns van aconseguir un resultat acceptable.

Taula 4.3- Millors resultats obtinguts en els experiments de models preentrenats amb ajust fi.
(Font: [2])

Model preentrenat	Entrenament		Validació		Testeig		Optimitzador utilitzat
	Pèrdua	Precisió	Pèrdua	Precisió	Pèrdua	Precisió	
VGG 16	0.35	86.77	0.32	88.58	0.34	87.58	SGD
VGG 19	0.29	89.14	0.26	88.83	0.27	89.92	SGD
Inception V3	0.08	97.40	0.13	94.59	0.16	93.75	SGD
Xception	0.13	96.00	0.14	94.51	0.14	94.50	SGD
Inception V4	0.42	90.85	0.36	92.91	0.38	91.25	SGD

Després d'aplicar el *fine tunning*, es comprova que el model *Xception* segueix sent el més efectiu.

Cal comentar que els models VGG 16 i 19 i els models *inception* i *Xception* han sigut tractats diferent degut a la seva estructura. Els models VGG són de caràcter seqüencial. És a dir, els blocs de capes segueixen un ordre un després de l'altre. Per altra banda, els models *inception* i *Xception* no són seqüencials, els blocs poden estar disposats en paral·lel. Això fa que s'hagin utilitzat classificadors seqüencials i no seqüencials [2].

4.4.4. Model Xception

El model *Xception* ha resultat el més efectiu a l'hora de tractar el nostre problema de classificació, pel que farem una breu explicació d'ell.

Creat l'abril de l'any 2017 per Francois Chollet, el mateix creador de la biblioteca Keras, el model *Xception* és una variant dels models *inception*. De fet, el seu nom prové de *Extreme Inception*. Es caracteritza per utilitzar la tècnica *depthwise* i *pointwise*. A grans trets, consisteixen a analitzar la imatge pas a pas. Explicat millor, els models més antics analitzen tota la imatge a la vegada (amplada, alçada i profunditat), el que comporta un pes computacional important. Per altra banda, els models *inception* i per tant el *Xception* també, realitzen un primer processat de la imatge analitzant-la capa per capa [7], [ix].

Per exemple, diguem que tenim una imatge 12 × 12 × 3 píxels que volem arribar a convertir en un tensor 8 × 8 × 256 píxels.

Si fem servir una convolució normal, aplicaríem a la imatge d'entrada un bloc de, per exemple, 5 × 5 × 3. Que analitzarà part per part la imatge i ens entregarà de resultat una capa 8 × 8 × 1. Aleshores, si en lloc d'utilitzar un bloc 5 × 5 × 3 en fem servir 256, aconseguirem les 256 capes 8 × 8.

Per altra banda, si fem servir *depthwise* i *pointwise* primer analitzaríem la imatge amb tres capes 5 × 5 × 1 que cada una ens entregaria una resposta 8 × 8 × 1. De manera que tindríem un bloc 8 × 8 × 3, això és la part *depthwise*. Després, aplicarem a la capa 8 × 8 × 3 un tensor 1 × 1 × 3 que analitzarà cada part del bloc per entregar-nos de resposta una capa 8 × 8 × 1. Finalment, d'igual manera que amb la convolució normal, si apliquem 256 tensors 1 × 1 × 3, obtindrem les 256 capes 8 × 8 que volem [viii].

Vist així, estem fent el mateix en ambdues formes. No obstant això, si observem el càlcul computacional de cada mètode, veurem els resultats clars.

Convolució normal: (5*5*3)*(8*8)*256 = 1.228.800 càlculs

Depthwise: (5*5*3)*(8*8) = 4.800 càlculs

Pointwise: (1*1*3)*(8*8)*256 = 49.152 càlculs

Depthwise + Pointwise = 53.952 càlculs

Al contrastar els 1.228.800 amb els 53.952 veiem clara la millora d'una forma respecte a l'altre.

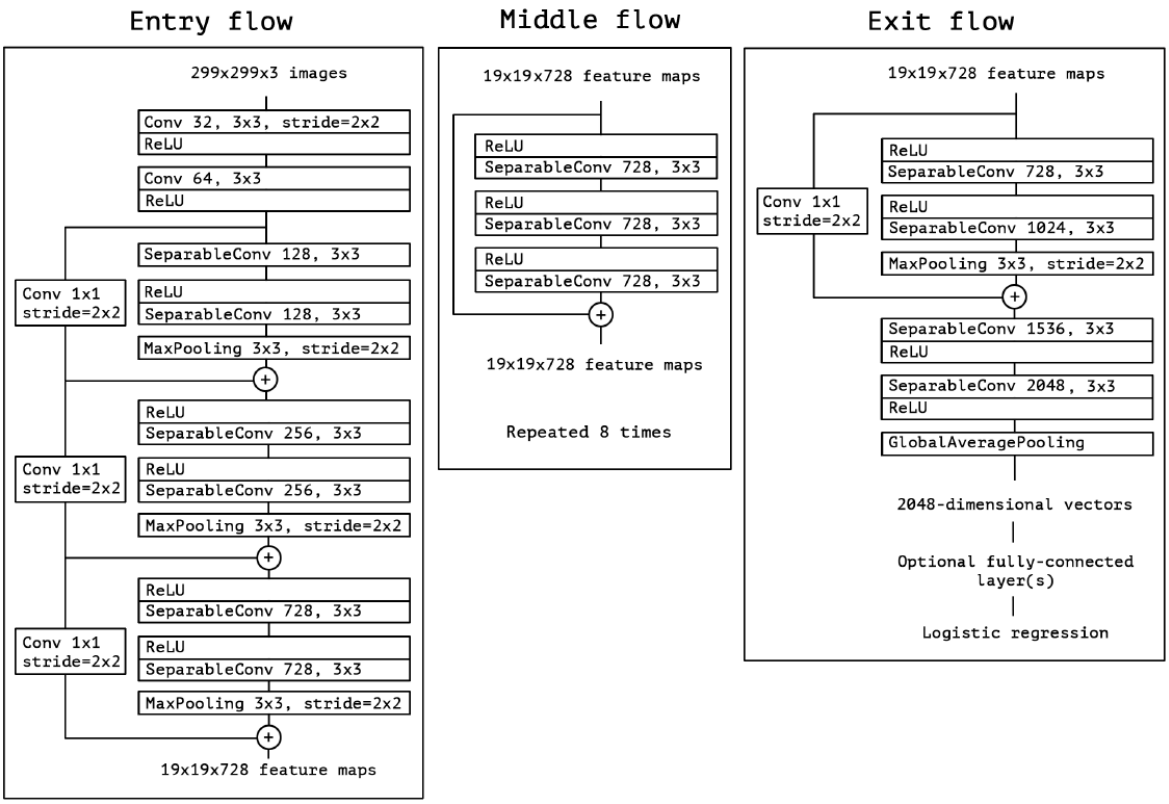

Figura 4.5.- Arquitectura del model preentrenat Xception.

(Font: [xxv])

4.4.5. Conclusions del model

En conclusió el model realitzat per a classificar imatges de cèl·lules sanguínies ha estat satisfactori. S'han aconseguit els objectius de poder classificar correctament les cèl·lules en les 4 classes pertinents amb una precisió de 94.5%, un valor acceptable.

Cal comentar que aquest resultat s'accepta com a bo, ja que aquest model és un ajut per al personal mèdic i no pas un substitutiu d'aquest, que aleshores sí que es demanaria una precisió molt més elevada.

5. Hardware i software del prototip

En aquest capítol s'explica tot el relacionat amb el hardware i el software del model físic en el qual s'implementaran tant el model de classificació com l'aplicació dash que es desenvoluparà més endavant.

5.1. Hardware

Com a entorn d'implementació s'ha escollit una placa de desenvolupament Jetson Tx2, del fabricant Nvidia. Una placa de desenvolupament és un ordinador d'una sola placa on s'hi ha col·locat tot el necessari per simular un ordinador de característiques més reduïdes o potenciant únicament les que ens interessa [x].

S'ha escollit aquesta placa perquè es necessitava una placa amb una unitat de processament gràfic (GPU) prou elevada per a poder corre el model de classificació de manera àgil, sense sobrecarregar-la. A continuació es mostra la placa i s'enllisten les seves característiques.

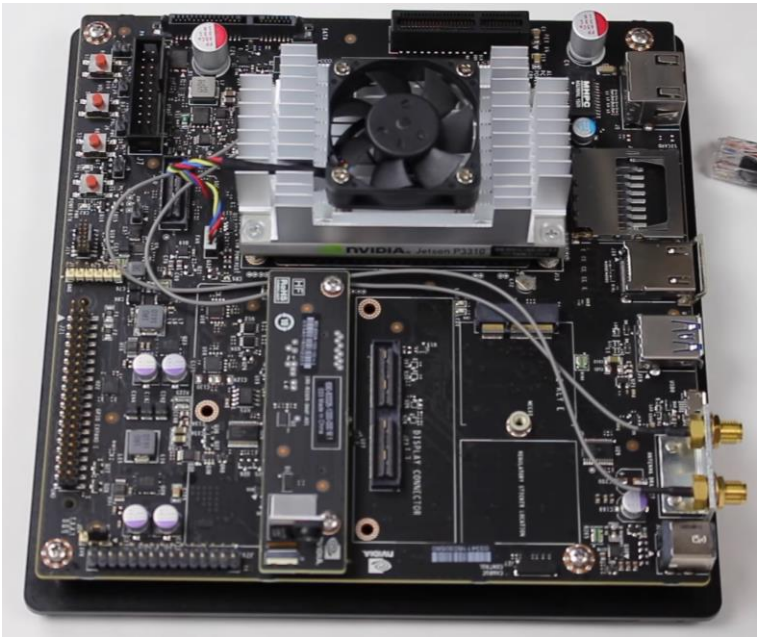

Figura 5.1.- Vista superior de la placa de desenvolupament Jetson TX2.
(Font: Pròpia)

Llista de característiques [x]:

GPU: NVIDIA Pacal™, 256 CUDA Cores 8GB
CPU: HMP Dual Denver 2/2 MB L2 + Quad ARM® A57/2 MB L2
Vídeo: 4K x 2K 60Hz Codificador (HEVC), 4K x 2K 60Hz Decodificador (12-Bit Suport)
Memòria: 8GB 128 bit LPDDR4, 59.7Gb/s
Display: 2x DSI, 2x DP 1.2 / HDMI 2.0 / eDP 1.4
CSI: Fins a 6 càmeres (2 línies) CSI2 D-PHY 1.2 (2.5GBps/Línia)
PCIE: Gen 2 | 1x4 + 1x1 o 2x1 + 1x2
Emmagatzematge: 32GB eMMC, SDIO, SATA
Altres: CAN, UART, SPI, I2C, I2S, GPIOs
USB: USB 3.0 + USB 2.0
Connectivitat: 1 Gigabit ethernet, 802.11ac WLAN, Bluetooth
Mecànic: 50mm x 87mm (Connector Placa-a-Placa de 400 Pins)

En la Figura 5.2, s'observen els connectors externs que es poden fer servir al configurar la placa.

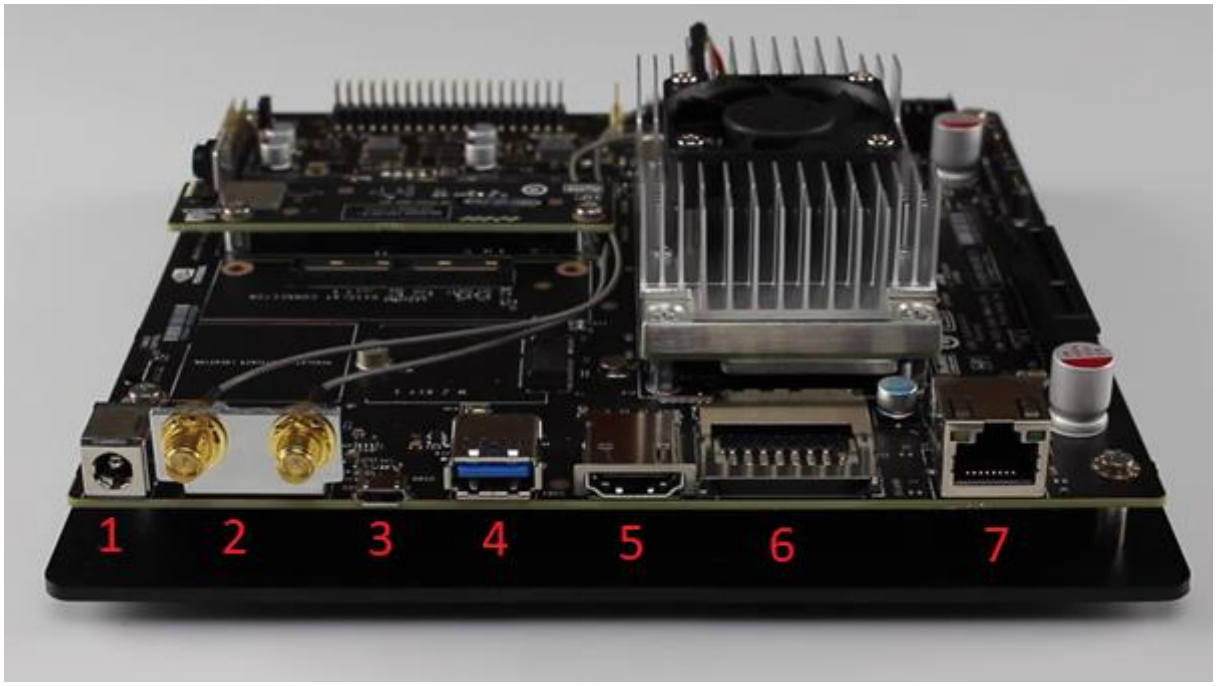

Figura 5.2.- Vista lateral de pla Jetson TX2: 1. Alimentació DC, 2. Antenes Wifi, 3. Micro USB tipus AB, 4. USB tipus A, 5. HMDI, 6. Lector de targetes SD, 7. Ethernet. (Font imatge: Pròpia, Font dels ports: [x])

5.2. Software

La part més important per què la placa funcioni correctament (contant que el hardware és operacional) és la configuració d'aquesta. Recordem que la Jetson Tx2 es comporta com un ordinador, de manera que se la configurarà com a qualsevol ordinador, tret d'alguna diferència.

La placa ve predeterminada amb el sistema operatiu Ubuntu 16.04. No obstant això, per a configurar tot el demès s'ha de fer a través d'un altre ordinador també amb Ubuntu 16.04. Per això ens servirà qualsevol ordinador amb aquest sistema o podem instal·lar-lo en un altre equip fent una partició de disc. Aquest últim ha estat el nostre cas. La connexió entre l'equip, que serà l'amfitrió (de l'anglès Host) i la Jetson Tx2, que serà l'objectiu (de l'anglès *Target*) es fa amb el port micro USB [xv]. Connectats amfitrió i objectiu, es procedeix a instal·lar el JetPack 3.2 als dos equips, trobarem aquest paquet a la pàgina oficial de Nvidia [xvii]. Aquest paquet ens permet el funcionament de la placa.

Un cop s'ha realitzat la instal·lació del JetPak seguint les instruccions d'instal·lació típiques cal instal·lar Python i les llibreries necessàries per a poder corre el model neuronal i l'aplicació Dash. Per tant, són necessàries les llibreries de Dash, Tensorflow i Keras [xv].

Python: És el llenguatge amb el qual s'ha programat el model de classificació. Es pot descarregar des de la pàgina oficial de Python [xvi], nosaltres instal·larem la versió Python 3.6.6.

Dash: És un entorn de treball de Plotly amb el qual es poden crear aplicacions web de caràcter local. Per instal·lar-lo només cal utilitzar l'ordre del quadre de codi 1 al quadre de comandaments (d'ubuntu) [i]:

```
Pip install Dash
```

Quadre de codi 1.- Instal·lació de l'entorn de treball Dash a través de la consola.

Tensorflow: És una llibreria desenvolupada per Google de codi obert per a aprenentatge automàtic. S'instal·larà la versió 1.9 amb l'ordre del quadre de codi 2 [xviii]:

```
pip3 install --extra-index-
url=https://developer.download.nvidia.com/compute/redist/jp33 tensorflow-gpu
```

Quadre de codi 2.- Instal·lació de la biblioteca Tensorflow 1.9 a través de la consola.

Keras: És una interfície de programació d'aplicacions d'alt nivell per programar models d'aprenentatge automàtic de codi obert que pot ser utilitzada juntament amb Tensorflow. Instal·larem la versió 2.2 amb les ordres del quadre de codi 3 [xix]:

```
Keras Install
echo "Install dependencies"
sudo apt-get -y install libhdf5-serial-dev hdf5-tools
for python3
sudo -H pip3 install keras
```

Quadre de codi 3.- Instal·lació de la biblioteca Keras 2.2 a través de la consola.

6. Disseny i desenvolupament de l'aplicació Dash

6.1. Introducció al Dash

Dash és un entorn de desenvolupament d'aplicacions web produït l'any 2015 per Plotly, una empresa d'informàtica enfocada en l'anàlisi de dades i la visualització d'aquestes. Dash esdevé l'aposta de Plotly per desenvolupar aplicacions web per a la visualització de dades creant una pàgina web de caràcter HTML però en llenguatge Python, un dels llenguatges de programació més utilitzats en l'actualitat [xi].

Fent ús d'ell, és possible crear aplicacions web de caràcter local capaces de visualitzar dades i d'interactuar amb l'usuari de l'aplicació de tal manera que es pot recrear l'experiència d'utilitzar un programa complet.

El codi d'una aplicació Dash s'estructura en dos blocs principals: el bloc del *Layout* i el bloc dels *callbacks* i funcions. En la part del *layout* bàsicament s'especifica quins components tindrà la nostra aplicació i de quina manera estaran disposades [i]. Tot el layout s'emmarca dins d'una divisió, com es veu en el quadre de codi 1:

```
app.layout = html.Div([
        #Aqui va el layout
])
```

Quadre de codi 4.- Divisió principal del layout.

El component html.Div() ens genera un espai o bloc on situarem els components. Per exemple, si fem dues divisions seguides on cada una està formada únicament per un text qualsevol obtindrem el resultat A de la Figura 6.1. En canvi, si només emprem una divisió composta per dos textos qualsevol, obtindrem el resultat B [i].

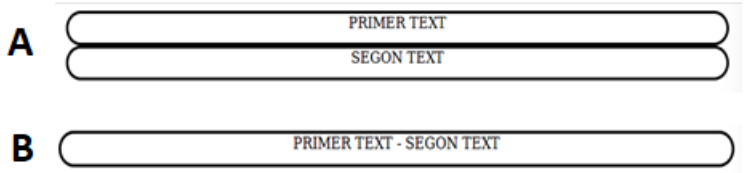

Figura 6.1.- Exemples de disposicions de divisions. A: Dues divisions seguides. B: Una divisió amb dues parts. (Font: Pròpia)

Seguidament se situen totes les funcions que s'activaran quan interactuem amb els components del *layout*. Aquestes s'escriuen com es veu en el quadre de codi 5:

```
@app.callback(Output(sortides),
              [Input(triggers)])

def nom_Del_def (entrades):
```

Quadre de codi 5.- Estructura dels callback, emprant funcions.

6.2. Plantejament de l'aplicació

Anteriorment s'ha creat un model capaç de classificar imatges de frotis de sang en quatre classes diferents en funció a les cèl·lules presents en aquestes, aquest model va enfocat a agilitzar i alleugerar el treball que ha de realitzar personal mèdic. Ara bé, aquest no necessàriament ha de ser capaç de fer anar el model fent ús de la consola del sistema. És per això que es va decidir crear un pont que connectes el potencial del model amb aquest personal.

Aleshores, es va decidir crear una aplicació amb el nou entorn Dash que brindés als metges la capacitat d'utilitzar tot el potencial del model de manera molt senzilla i intuïtiva.

6.2.1. Interfície

La interfície és la cara de l'aplicació, és el que veu qui la utilitza. Per tant, ha de ser clara i agradable a la vista, no pot està sobrecarregada ni tampoc desordenada. Ha d'estar presentada de tal manera que un cop utilitzada l'aplicació una vegada l'usuari sigui capaç d'utilitzar-la repetides vegades sense necessitat de tornar a entendre com funciona. Ha de seguir una lògica molt intuïtiva.

Amb aquests objectius clars, es van plantejar els punts o condicions que la interfície de la nostra aplicació hauria de complir:

- Ha d'haver-hi una zona per importar les imatges a classificar. Aquesta zona ha de ser fàcil de trobar pel que la situarem al principi de tot.
- Ha de ser capaç de mostrar-nos les imatges en diferents seccions en funció a la classificació rebuda. Es buscarà crear algun tipus de separació visual.
- Ha de calcular i mostrar amb algun tipus de gràfic com s'ha distribuït la classificació. Tant en valors com en percentatges.

Seguint aquests objectius ens disposem a realitzar un esquema de l'estructura que seguirà l'aplicació. Aquest esquema ens servirà de base a l'hora de col·locar les prestacions.

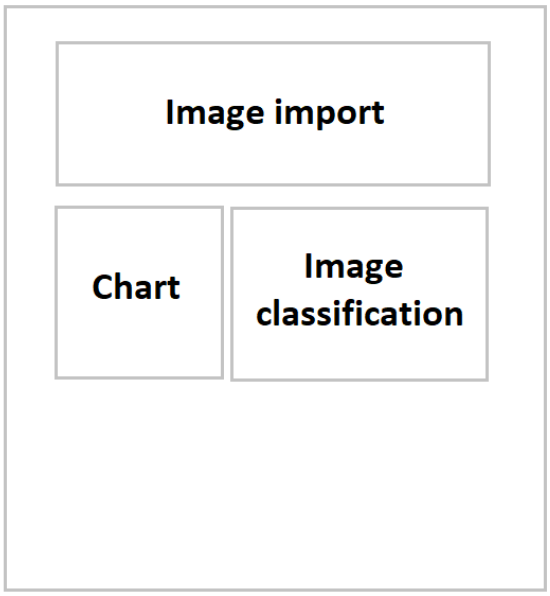

Figura 6.2.- Esquema base de l'aplicació.
(Font: Pròpia)

6.2.2. Eines per a desenvolupar l'aplicació

Dash és un entorn de treball de caràcter *open source*. Això vol dir que el codi darrere d'aquest està obert al públic. És a dir, qualsevol persona pot accedir a la seva base de dades i utilitzar-la lliurement.

A la pàgina web oficial de plotly trobem el codi necessari per a cada una de les funcions que pot fer Dash [xii].

En aquesta base de dades identifiquem ràpidament tres mòduls que ens interessaran:

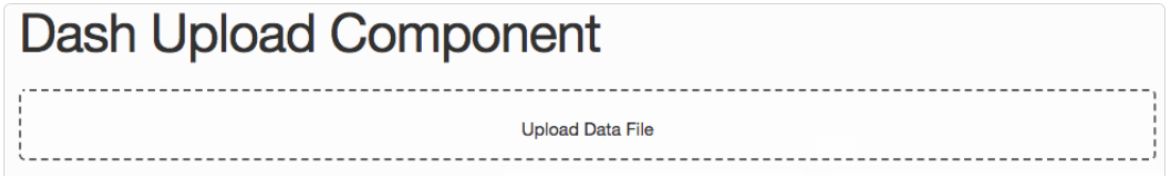

Figura 6.3.- Dash - Importar imatges.

(Font: [xii])

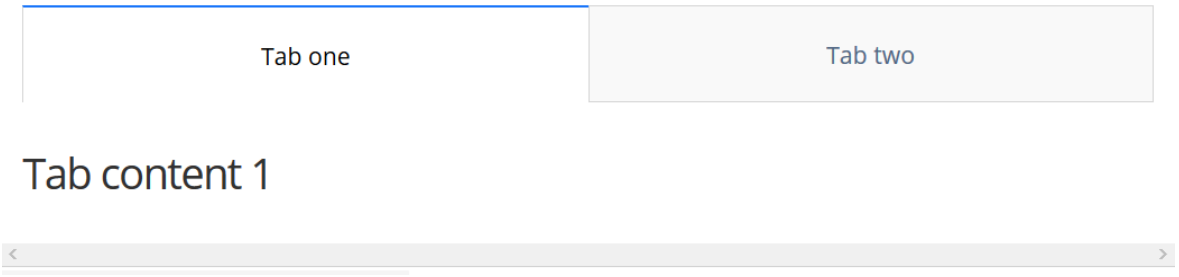

Figura 6.4.- Dash – Pestanyes per a visualització d'informació en funció a quina es prem.

(Font: [xii])

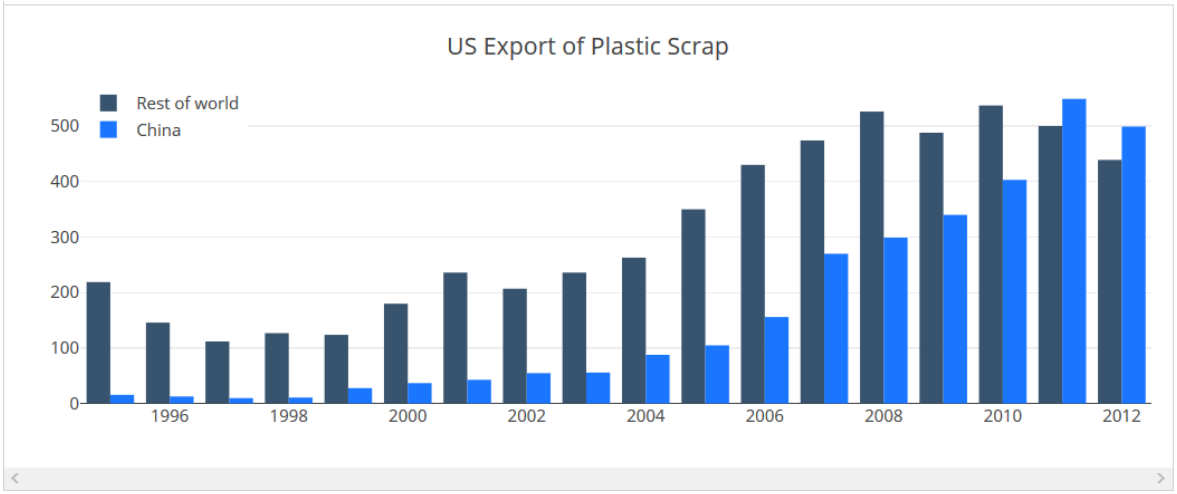

Figura 6.5.- Dash – Gràfiques.

(Font: [xii])

Aquesta base de dades és molt útil, ja que et mostra exemples de funcions que pot realitzar el Dash (vegeu Figures 6.2, 6.3 i 6.4) amb els que pots interactuar per veure que poden fer i com funcionen. Juntament amb els exemples hi ha el seu codi, d'aquesta manera es té a disposició un exemple amb com s'ha fet. Amb això, ràpidament es pot relacionar cada part del codi amb el què fa. A més, al final de cada exemple hi ha un enllaç que si es prem, et porta a una nova pestanya on trobem diferents formes de realitzar l'exemple en concret. Per exemple, si es prem l'enllaç de les pestanyes, ens mostrarà diferents codis que faran que les pestanyes tinguin una estètica diferent o ens ensenyaran a variar la seva disposició.

6.3. Desenvolupament de l'aplicació

6.3.1. Descripció de l'aplicació

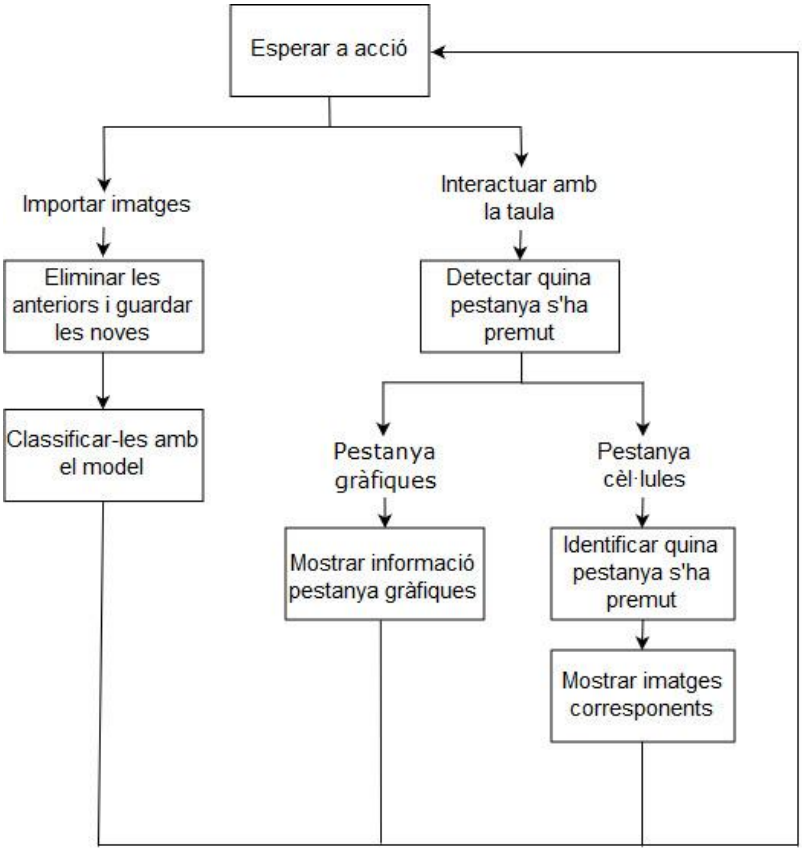

Figura 6.6.- Esquema del funcionament de l'aplicació.
(Font: Pròpia)

En la Figura 6.6 s'observa el diagrama de flux que seguirà l'aplicació. Quan l'aplicació s'hagi obert i tot el *layout* s'hagi carregat, s'esperarà a que l'usuari faci una de les dues accions possibles. Realment, la primera vegada només podrà realitzar una acció, que és importar imatges. Quan hagi seleccionat les imatges a importar, es guardaran en un directori i a continuació es procedirà a classificar-les. Un cop classificades es generarà una taula amb diverses pestanyes, una per mostrar les gràfiques que mostraran els resultats i les altres per mostrar les imatges classificades a cada tipus. Això es mostrarà en funció a quina pestanya s'hagi premut. Després d'això, l'usuari podrà escollir si seguir observant els resultats de la classificació actual o importar noves imatges per a ser classificades. Per a dur a terme aquestes accions es requeriran més ordres i procediments que veurem en els següents subapartats.

6.3.2. Llibreries necessàries

Per a desenvolupar l'aplicació va ser necessari instal·lar una sèrie de llibreries i components, en el sistema, que s'enllisten a continuació:

Taula 6.1- Llistat de les biblioteques necessàries per a treballar amb Dash còmodament.
(Font: [i])

chardet==3.0.4	numpy==1.14.2
click==6.7	pandas==0.22.0
Cython==0.28.2	pandas-datareader==0.6.0
dash==0.21.0	plotly==2.5.1
dash-core-components==0.22.1	python-dateutil==2.7.2
dash-html-components==0.10.0	pytz==2018.4
dash-renderer==0.12.1	requests==2.18.4
decorator==4.3.0	urllib3==1.22
nbformat==4.4.0	Werkzeug==0.14.1

Aquesta instal·lació s'ha realitzat amb un arxiu tipus .txt en el que hi ha aquesta mateixa llista de components i seguidament entrant en la consola l'ordre del quadre de codi 6:

```
pip install -r requirements.txt
```

Quadre de codi 6.- Instal·lació de les biblioteques necessàries a través de la consola.

Abans de realitzar aquesta ordre s'ha d'anar fins a la ubicació on es troba l'arxiu .txt amb l'ordre del quadre de codi 7:

```
cd (ubicació)
```

Quadre de codi 7.- Ordre per a desplaçar-se fins a la ubicació desitjada.

6.3.3. Ordres prèvies

Abans de començar a dissenyar la interfície i l'esquelet de l'aplicació cal realitzar una sèrie de tasques. Entre aquestes trobem importar les llibreries (quadre de codi 8), crear una carpeta per guardar les imatges i crear el servidor flask.

```
8   import base64
9   import os
10  # import sh
11  from flask import Flask
12
13  import dash
14  # import dash_auth # Use only as a secure option
15  from dash.dependencies import Input, Output, State
16  import dash_core_components as dcc
17  import dash_html_components as html
18  import time
19
20   import plotly.graph_objs as go
21
22  import numpy as np
23  import matplotlib.pyplot as plt
24  from collections import Counter
```

Quadre de codi 8.- Importació de les llibreries necessàries.

El primer que farem a l'aplicació serà crear una carpeta on guardar les imatges a classificar (Línies 29-37 de l'annex). Fem aquest pas perquè el servidor treballa de manera més efectiva si guardem les imatges que ens interessa classificar en una carpeta a part, sobretot per contar el nombre d'imatges i el nombre de classes.

A continuació ens disposarem a crear el servidor flask per l'aplicació. Les ordres per crear servidors flask per a aplicacions Dash es poden veure en el quadre de codi 9.

```
42  server = Flask(__name__)
43  app = dash.Dash(__name__, server=server)
```

Quadre de codi 9.- Creació del servidor flask i l'aplicació.

6.3.4. Disseny de la interfície

L'inici de la interfície és la primera vista que tindrem d'aquesta abans de realitzar cap acció amb l'aplicació. En altres paraules, la presentació. Aleshores, per a la pàgina inicial es van cercar a internet interfícies ja existents de manera que les puguem utilitzar fent una importació d'aquesta en el mateix codi (Línies 127-142). Això es fa just després de la divisió mare de l'aplicació.

Aquestes ordres busquen en la xarxa aquestes adreces, on hi ha la informació necessària. D'aquesta manera aconseguim que en la pàgina inicial de l'aplicació hi hagi un rètol on posar el nom de l'aplicació, que s'anomenarà *Cellsimatic Classification App*, i una imatge de la icona de Plotly Dash. Tot això ens ho encasella dins d'un quadre on col·locarem el resta de components. Per situar el rètol i la imatge farem servir les ordres del quadre de codi 10.

```
58      # Banner
59      html.Div([
60          html.H2(
61              'Cellsimatic Classification App',
62              id='title'
63          ),
64          html.Img(
65              src="https://s3-us-west-1.amazonaws.com/plotly-tutorials/logo/new-
    branding/dash-logo-by-plotly-stripe-inverted.png"
66          )
67      ],
68          className="banner"
69      ),
```

Quadre de codi 10.- Divisió del rètol i imatge.

Fixem-nos en el fet que s'ha realitzat una sola divisió dividida en dos blocs diferents, com s'ha comentat en l'apartat 6.1. Aquesta mateixa estructura és representada en la forma B de la Figura 6.1. Seguidament disposarem l'importador d'imatges fent servir l'estructura de la pàgina Dash core components (quadre de codi 11) [xii]:

```
71      #  Image Uploader
72      html.Div(className="container",
73          children=[

90              #Upload Images
91              dcc.Upload(
92              id='upload-image',
93              children=html.Div([
94                  'Drag and Drop or ',
95                  html.A('Select Files'),
96              ]),
97              style={
98                  'width': '90%',
99                  'height': '60px',
100                 'lineHeight': '60px',
101                 'borderWidth': '1px',
102                 'borderStyle': 'dashed',
103                 'borderRadius': '25px',
    . . .
```

```
104             'textAlign': 'center',
105             'marginBottom':'10px',
106             'marginLeft':'50px',
107             #'margin' : 'auto',
108         },
109         # Allow multiple files to be uploaded
110         multiple=True,
111         accept='image/*'
112     ),
```

Quadre de codi 11.- Divisió de l'importador d'imatges amb el seu estil.

Ajuntant les ordres del css, del ròtol i títol i del quadre on importar imatges obtenim una primera vista de l'aplicació (Figura 6.6).

Figura 6.7.- Interfície inicial de l'aplicació Dash.
(Font: Pròpia)

Pel que fa a la disposició de les gràfiques i les imatges, la idea principal era aconseguir la interfície plantejada en l'apartat 6.2.1 (Figura 6.2). No obstant això, es va observar que situant els gràfics a l'esquerra es perdia molt d'espai a l'hora de mostrar les imatges classificades. Aquest problema es va solucionar en utilitzar una de les pestanyes de la taula per mostrar les gràfiques en lloc de fer-ho fora d'aquesta. Per implementar això es va requerir molt de temps i moltes proves, ja que no es trobava la forma que aquesta idea funcionés.

Finalment, es va aconseguir fer-ho quan es va descobrir que no necessàriament els components han d'estar especificats en la part del *layout* sinó que es poden importar d'igual manera que s'importaria el resultat d'una funció. D'aquesta manera es poden mostrar nous components en la pantalla de l'aplicació en funció al que es premi. De l'altra manera es partia de què l'aplicació tindria certs components des d'un principi i aquest variarien. Això només és possible si executem l'ordre del quadre de codi 12 abans del *layout*:

```
51  app.config['suppress_callback_exceptions']=True
```

Quadre de codi 12.- Ordre que permet importar layout a través dels callback.

Al quadre de codi 13 podem veure la forma d'importar *layout* amb un callback.

```
114     html.Div(id='output-image-upload'),
115     html.Div(id='tabs-content'),
```

Quadre de codi 13.- Ordres amb les que importar layout. Estrictament aquestes importen informació, pot ser layout o pot ser informació d'altre tipus.

Aquestes dues línies el que fan és generar dues divisions en les quals es mostrarà el que hi hagi lligat a les id que han sigut introduïdes dins seu. Més endavant es veurà que es lliga a aquestes.

6.3.5. Callbacks i Funcions

Com s'ha comentat anteriorment, un callback és una resposta a una acció dins de l'aplicació. Utilitzant aquesta propietat s'ha introduït la generació de la taula dins d'un callback, en concret dins del lligat a importar imatges. Quan s'importen imatges, entre altres coses, s'etiqueten cada una de les imatges d'entrada segons la classificació rebuda i a continuació es crea el component taula amb una pestanya per a cada classificació d'aquestes imatges i una pestanya base on s'hi situaran les gràfiques [i]. Això es fa com es veu en el quadre de codi 14:

```
237     # Automatic Tabs infered from the LABELS
238     children = html.Div([
239             dcc.Tabs( id="tabs", value='tab-0', children=[
240                 dcc.Tab(label='Chart', value='tab-0')] +
241                 [ dcc.Tab(label=l, value=l) for l in LABELS ] )
242             ])
243
244     return children
```

Quadre de codi 14.- Creació de la taula capaç de crear una pestanya per a cada una de les classes detectades a més d'una pestanya base per a les gràfiques.

En la Figura 6.7 s'observa un exemple en el qual el classificador ha detectat dos tipus de cèl·lules (blast i limfòcits atípics) i, per tant, ha generat una taula amb una pestanya per cada una d'aquestes juntament amb la pestanya base de les gràfiques.

Figura 6.8.- Interfície amb pestanyes de l'aplicació Dash.
(Font: Pròpia)

L'esquelet de l'aplicació està format per 2 callbacks i diverses funcions. Els dos s'activen amb importar imatges i en prémer les diverses pestanyes de la taula. Els callbacks es comporten de manera similar a les funcions en el sentit de què tenen valors de sortida i d'entrada, l'únic que les entrades i sortides van lligades a les ID dels components.

Al callback lligat a l'import d'imatges se li han assignat diverses tasques a realitzar, tasques que formen part del mateix import d'imatges i altres que ens interessa que passin al mateix moment, com la creació de la taula abans comentada. A continuació es presenta part del codi amb una explicació d'aquest.

La primera tasca és eliminar les imatges de l'anterior classificació que estan a la carpeta creada al principi i guardar les noves imatges importades a aquesta mateixa. Això ho fem amb un senzill condicional que, si detecta arxius dins la carpeta, executa la funció d'esborrar arxius (quadre de codi 15).

```
202    # Delete the previous files
203    if os.listdir(UPLOAD_DIRECTORY):
204        removefiles()
```

Quadre de codi 15.- Ordre per detectar si hi ha imatges al directori especificat i, si n'hi hagués, esborrar-les.

El codi de removefiles() el trobem a les línies 353-363 de l'annex.

Les imatges es guarden de manera similar, s'utilitza el mateix condicional per assegurar-se que no hi ha imatges a la carpeta (quadre de codi 16). Si no ens asseguréssim, podrien quedar imatges de l'anterior classificació que no s'han esborrat correctament i es barrejarien amb les noves, contaminant la classificació.

```
206    # Save all the images
207    if list_of_contents is not None:
208        for c, n in zip(list_of_contents, list_of_names):
209            save_file(n, c)
```

Quadre de codi 16.- Guardar les imatges importades a la carpeta especificada.

El codi de la funció save_file(n, c) el trobem a les línies 346-351 de l'annex.

Un cop guardades les noves imatges, cridarem el model de predicció perquè les classifiqui i seguidament es crearà una llista amb les diferents classificacions obtingudes (quadre de codi 17):

```
221     # Keras classifier
222     PREDICTIONS_DICT = prediction(UPLOAD_DIRECTORY)
223
224     print("El tiempo para clasificar cada imagen es de {} s".format(time.time()-start))
225
226
227     # Make the prediction dictionary with files and predictions
228     # PREDICTIONS_DICT = dict(zip(files, predictions))
229     print(PREDICTIONS_DICT)
230
231     # ================================================
232
233     # Unique labels
234     LABELS = list(set(PREDICTIONS_DICT.values()))
```

Quadre de codi 17.- Crida del model de predicció perquè classifiqui les imatges importades.

Finalment, després de la classificació, trobem la creació de la taula, comentada anteriorment.

El segon callback actua quan canviem de pestanyes. Aleshores, la funció que farà és assegurar-se de què es mostren les imatges a les pestanyes pertinents i les gràfiques. Primer, fent ús d'un condicional, realitzarem unes accions o altres en funció de si es tracta de la pestanya de les gràfiques o les altres. Començarem per la de les gràfiques.

El primer que es farà serà adquirir les dades necessàries per a realitzar les gràfiques. Aquesta tasca és senzilla, tan sols s'ha de mirar la llista de prediccions i contar tant les diferents classificacions com la quantitat d'imatges a cada una d'aquestes (quadre de codi 18).

```
263     # Count the frecuencies of each predicted label
264     labels, values = count_images(PREDICTIONS_DICT)

339 #Count amount of Images
340 def count_images(PD):
341     d = Counter(PD.values())
342     labels = list(d.keys())
343     values = list(d.values())
344     return labels, values
```

Quadre de codi 18.- Ordres per contar el nombre d'imatges de cada tipus i el nombre de classificacions diferents.

Per recomanació de la doctora Anna Merino s'afegirà un missatge que indiqui la classe majoritària (línies 266-274). Aquest missatge el farem servir unes línies més tard.

La següent divisió, mostrada en el quadre de codi 19, està formada per diverses divisions dins d'ella, tot aquest conjunt és el que s'exportarà com a *layout* per a les gràfiques. En la part superior se situa el missatge seguidament per les dues gràfiques en paral·lel sota seu.

```
277     out = html.Div([
278         html.H3(mensaje),
279         html.Div([
280         html.Div([
281             dcc.Graph(
282                 figure={
283                     'data':[
284                         go.Bar(x=labels, y=values,
285                             textfont=dict(size=20))
286                     ],
287                 'layout':go.Layout(
288                   title="Bar plot of predictions"
289                     )
290                 }
291           )],
292             className="six columns",
293              ),
294           html.Div([
295             dcc.Graph(
296                 figure={
297                     'data':[
298                         go.Pie(labels=labels, values=values,
299                             textfont=dict(size=20))
300                     ],
301                 'layout':go.Layout(
302                   title="Pie chart of predictions"
303                     )
304                 }
305           )],
306             className="six columns",
307         ),
308     ], className="row")
309     ])
310
311     return out
```

Quadre de codi 19.- Layout de les gràfiques. S'ha utilitzat el *className="six columns* perquè es vegin dues gràfiques en paral·lel.

Afegint aquesta estructura a la qual ja teníem anteriorment ens generarà un *layout* tal com el de la Figura 6.8.

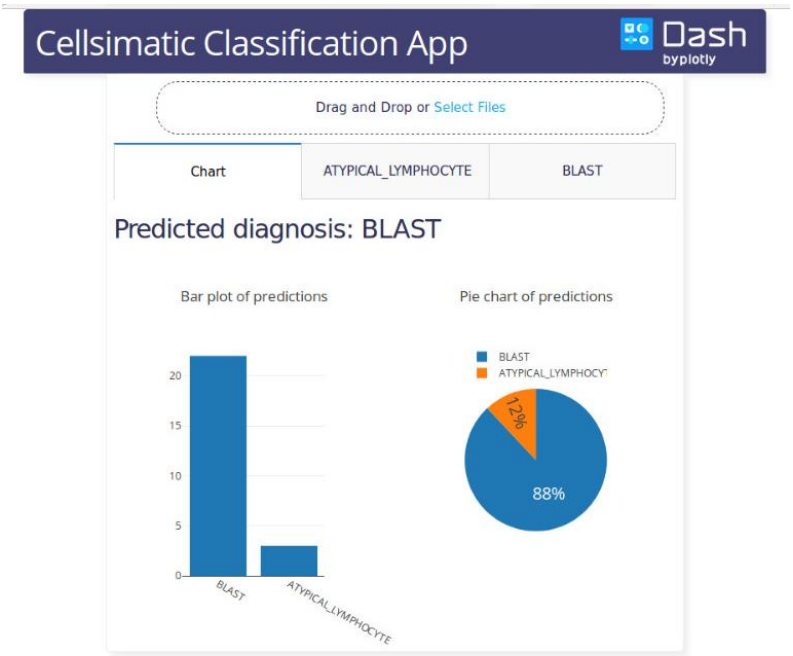

Figura 6.9.- Interfície amb gràfiques de l'aplicació Dash.
(Font: *Pròpia*)

En cas que no estiguem a la pestanya de les gràfiques, o en altres paraules, estiguem a les pestanyes de les imatges classificades, realitzarem una acció diferent, mostrada en el quadre de codi 20.

```
313    else:
314        for l in LABELS:
315            if tab == l:
316                return html.Div( children = show_test(tab) )
```

Quadre de codi 20.- Detectar quina pestanya s'ha pres i cridar la funció show_test entregant-li aquesta informació.

Aquest bloc detecta a quina classe representa la pestanya que s'ha premut i envia aquesta informació a la funció *show_test*, que busca en la galeria d'imatges classificades aquelles que comparteixen classe amb la pestanya seleccionada i les envia a la funció *parse_contents*, que crearà el *layout* adient per a aquestes. Finalment afegeix cada imatge, amb el seu *layout* corresponent, a la divisió de la pestanya seleccionada (quadre de codi 21).

```
370 def show_test(clase):
371     children = []
372     for name,label in PREDICTIONS_DICT.items():
373         if label == clase:
374             # out.append(html.H4(name))
375             contents = encode_image(os.path.join(UPLOAD_DIRECTORY, name))
376             children.append(parse_contents(contents, name, label))
377     return children

152     def parse_contents(contents, filename, prediction):
153     layout = html.Div([
154         html.Div(className='gallery',
155                  children=[
156                     html.A(
157                         target='_blank',
158                         href=contents,
159                         # HTML images accept base64 encoded strings in the same format
160                         # that is supplied by the upload
161                         children=html.Img(src=contents, width=360, height=363,
162                                           style={'width': '100%',
163                                                  'height': 'auto'})
164                     ),
165                     html.Div(filename,
166                              style={'padding': 5,
167                                     'textAlign': 'center',
168                                     'font-size': '0.8em'}
169                              ),

179                  ],
180                  style={
181                     'margin': 5,
182                     'border': '1px solid #ccc',
183                     'float': 'left',
184                     'width': 150,
185                     'height': 190}
186                  )
187     ])
188
189     return layout
```

Quadre de codi 21.- Bloc encarregat de detectar quina pestanya s'ha premut i generar el *layout* amb totes les imatges corresponents per a la pestanya en concret.

Amb la implementació d'aquest bloc i el de les gràfiques aconseguim que l'aplicació prengui la forma final (vegeu Figures 6.10 i 6.11).

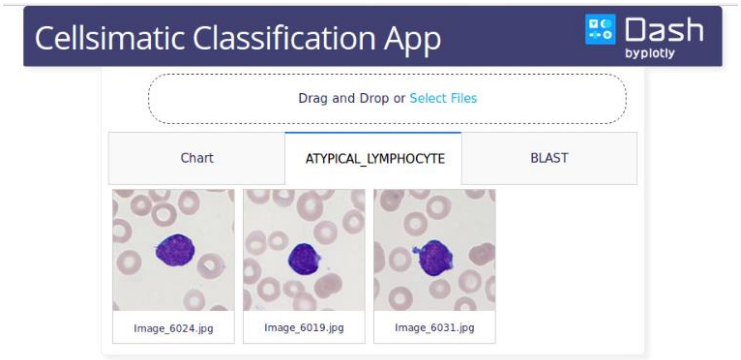

Figura 6.10.- Pestanya amb les classificacions de limfòcit anormal.
(Font: Pròpia)

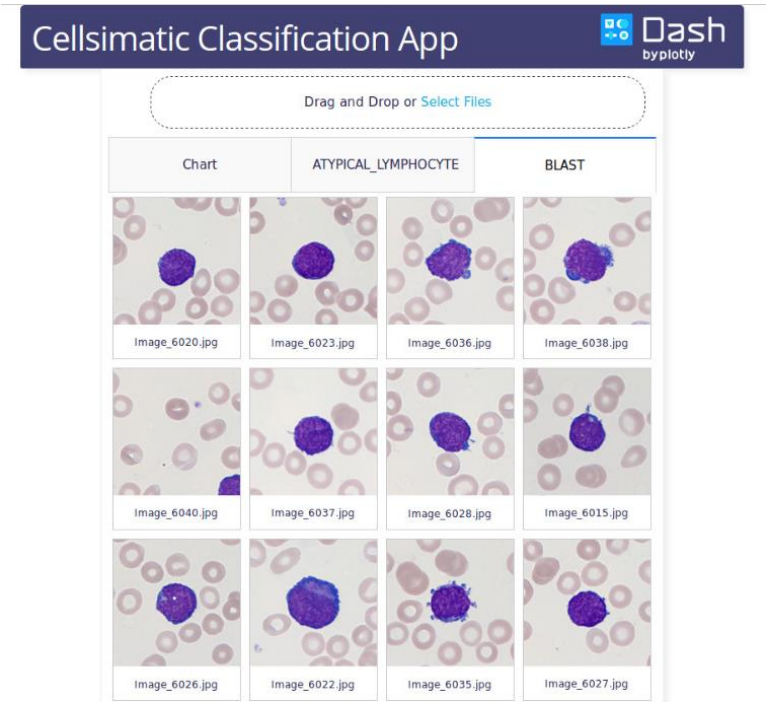

Figura 6.11.- Pestanya amb les classificacions de Blasts.
(Font: Pròpia)

Cal destacar la funció *enconde_image* (línies 366-368), que s'encarrega de codificar les imatges en format base64. Aquesta codificació és necessària per a l'aplicació Dash. En cas contrari, no podria treballar amb les imatges.

6.4. Resultats de l'aplicació

Amb tot això aconseguim una aplicació fàcil d'utilitzar i ràpida a l'hora de classificar que presenta les següents propietats:

- ✓ Permet la classificació de múltiples imatges a la vegada.
- ✓ Permet visualitzar quines imatges han sigut classificades a cada una de les classificacions.
- ✓ Aporta informació visual en forma de gràfica de barres i un diagrama de sectors sobre les classificacions.
- ✓ Ens informa a través d'un missatge quin ha estat el grup majoritari.
- ✓ Permet tornar a classificar un altre grup d'imatges sense necessitat de reiniciar l'aplicació.

L'últim pas és integrar l'aplicació i el model a la placa Jetson Tx2. Això es fa de manera senzilla amb un *pendrive* mateix. S'exporten a aquesta l'aplicació Dash en format python, els arxius importants del model (.h5, .json, classes) i algunes imatges per a poder realitzar proves finals de caràcter demostratiu.

7. Anàlisi de l'impacte ambiental

Aquest treball no genera cap residu o impacte en el medi ambient més enllà dels impactes indirectes que comporten el procés de fabricació de la placa de desenvolupament i l'energia necessària per al seu funcionament. No obstant això, el correcte funcionament de la placa aporta una reducció de temps considerable que sestaria personal mèdic realitzant les classificacions pel que aquest impacte es pot considerar no important.

Un altre impacte que es podria considerar és el de fabricació del microscopi electrònic encarregat de prendre les imatges cel·lulars, ja que sense ell el treball no seria possible.

8. Anàlisi Econòmica

A l'hora de realitzar aquest treball han sigut necessaris molts recursos, ja que es basa en recerca i investigació i la realització d'una aplicació de programa i codi gratuït. No obstant això, sí que ha sigut necessari la compra d'una placa de desenvolupament. Altres productes es poden considerar menyspreables, ja que són habituals en entorns hospitalaris com pantalles d'ordinador i ratolins.

8.1. Material

En aquest subapartat es consideraran tots els costos materials sense els quals no hauria sigut possible aquest treball. No serà contat l'ordinador portàtil personal perquè no ha sigut comprat explícitament per a la realització d'aquest.

Taula 8.1- Llistat de productes necessaris per a la realització del treball amb els seus preus. (Font: [2])

Producte	Preu (€)
Microscopi Olympus DX43 [2]	6450
Objectiu OPTIKA C-P8 [2]	936
Nvidia Jetson TX2 [xxviii]	671.18
Total (IVA inclòs)	8057.18

8.2. Software

Tot el software utilitzat és de caràcter gratuït i pot ser trobat en l'últim apartat de la bibliografia, el qual enllista tot el software utilitzat i on trobar-lo.

8.3. Formació

La gran majoria de formació ha provingut de la lectura d'estudis oberts al públic i diverses pàgines web per a contrastar la informació. Tanmateix, sí que es va realitzar un curs en línia de pago.

Taula 8.2- Llistat de la formació no gratuïta realitzada amb el seu preu.
(Font: [2])

Formació	Preu (€)
Udemy - *Interactive Python Dashboards with Plotly and Dash* [xxix]	10
Total (IVA inclòs)	10

8.4. Costos d'enginyeria

Els costos d'enginyeria consisteixen en la suma del valor del temps invertit en el treball d'un enginyer electrònic industrial i ambdós director i co-director. El treball té una durada aproximada de 600 hores, que és el resultat de multiplicar els 24 crèdits acadèmics per la durada de 25 hores per crèdit.

Taula 8.3- Pressupost d'enginyeria.
(Font: [2])

Individu	Temps (h)	Import hora (€)	Total (€)
Enginyer electrònic industrial i automàtic junior	600	10	6000
Director del treball	40	20	800
Co-director del treball	10	25	250
IVA 21%			1480.5
Total (IVA inclòs)			8530.5

9. Conclusions

9.1. Conclusions

Es considera que el resultat del treball ha estat més que satisfactori. S'ha creat un prototip funcional d'un dispositiu per al reconeixement automàtic d'imatges de cèl·lules sanguínies portàtil capaç de complementar-se amb personal mèdic per tal d'aconseguir realitzar operacions de manera molt més ràpida i eficaç, amb una precisió de 94.5%. Cosa que quan parlem de diagnòstic de patologies en pacients, el temps premia.

En aquest procés s'han adquirit coneixements extensos sobre el *machine learning* i el *deep learning,* així com els tipus de problemes possibles i com resoldre'ls, la idea de neurona artificial i les xarxes neuronals artificials, l'estructura i la metodologia d'aquestes últimes i com poden arribar a aprendre a classificar grans bases de dades. S'ha observat el potencial de les xarxes neuronals convolucionals en el camp del reconeixement d'imatges i el fet d'utilitzar models preentrenats, com el model Xception, per a maximitzar el rendiment dels models així com altres tècniques que serveixen per acabar de perfilar-los, com l'ajust fi.

S'ha entès la base biològica que ens permet poder arribar a classificar diferent tipus de cèl·lules, en els seus grups corresponents, en funció a la morfologia d'aquestes, arribant a diferenciar cèl·lules immadures de tres tipus de limfòcits diferents. S'ha entès l'interès a poder realitzar aquesta classificació, descobrint les diferents patologies que pot indicar la presència, en grans números, d'alguna d'aquestes cèl·lules en sang. Juntament amb això, s'ha estudiat un model de classificació entrenat, que actualment està sent utilitzat a l'hospital clínic de Barcelona, per a realitzar aquesta tasca.

S'han après les bases de codificació en llenguatge Python per a la creació de xarxes neuronals i s'ha endinsat molt en el territori de l'entorn de treball Dash. Sent capaços de desenvolupar una aplicació web amb característiques complexes per a millorar els processos mèdics de diagnòstic de patologies malignes, permetent a aquest importar imatges que, un cop classificades internament, seran disposades a la pantalla en diferents pestanyes amb una pestanya extra que indica els resultats obtinguts de la classificació.

Finalment, s'ha descobert el món de les plaques de desenvolupament i el seu potencial en l'àmbit de la computació de models convolucionals, emprant equips d'una mida molt compacte i òptims, com és el cas de la placa Jetson Tx2 del fabricant Nvidia. Capaç de fer corre el model convolucional de classificació amb una velocitat de càlcul molt elevada pel preu d'aquesta.

9.2. Possibles millores

El què és interessant de la tecnologia és que un producte mai estarà acabat al complet, sempre hi haurà lloc a millores. Aquest cas no és una excepció, pot ser millorat per totes bandes:

Per la part del model es podria aconseguir precisions més elevades provant diferents models preentrenats, entrenant-los diferent, aconseguint més dades i més equilibrades, entre altres.

A part, el model utilitzat és capaç de classificar imatges de cèl·lules associades a 18 patologies diferents en 4 grans grups. Una possible millora seria aconseguir un model prou sofisticat per a poder classificar-les en els 18 grups diferents.

Per la part de la placa de desenvolupament també hi podria haver millores. Utilitzant plaques més potents obtindríem millors resultats, encara que es considera que no compensa la petita millora que obtindríem amb l'increment de preu. La Jetson TX2 és una molt bona placa en relació qualitat/preu.

Per últim, l'aplicació podria ser millorada de moltes maneres, ja que aquesta és una versió molt nata d'aplicació funcional. Es podrien afegir moltes característiques com:

- Una interfície més clara o més bonica.
- Poder moure imatges d'una pestanya a un altre per corregir els possibles errors de classificació.
- Donar l'opció a escollir de què noves pujades d'imatges eliminin les anteriors o s'afegeixin a aquestes.
- Permetre canviar l'idioma de l'aplicació.

Bibliografia

Referències de la literatura

[1] Alférez Baquero, E. S. (2015). **Methodology for automatic classification of atypical lymphoid cells from peripheral blood cell images**. [Accés: Gener 2019]

[2] Villarraso Jiménez, R. (2018). **Reconocimiento automático de células malignas en sangre periférica a partir de imágenes digitales y utilizando redes neuronales convolucionales**. [Accés: Gener 2019]

[3] Chollet, Francois. "**Deep Learning with Python**." (2017). [Accés: Gener 2019]

[4] Kotsiantis, Sotiris B., I. Zaharakis, and P. Pintelas. "**Supervised machine learning: A review of classification techniques**." Emerging artificial intelligence applications in computer engineering 160 (2007): 3-24. [Accés: Febrer 2019]

[5] Alpaydin, Ethem. **Introduction to machine learning**, MIT press, 2009. [Accés: Febrer 2019]

[6] Zurada, Jacek M. **Introduction to artificial neural systems**. Vol. 8. St. Paul: West publishing company, 1992. [Accés: Febrer 2019]

[7] Chollet, François. "**Xception: Deep learning with depthwise separable convolutions**." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017. [Accés: Març 2019]

Referències en la xarxa

[i] Curs online d' Udemy - *Interactive Python Dashboards with Plotly and Dash*: https://www.udemy.com/interactive-python-dashboards-with-plotly-and-dash/learn/v4/overview [Accés: Gener 2019]

[ii] Diferents usos del Machine leargning en la actualitat. https://www.forbes.com/sites/bernardmarr/2016/09/30/what-are-the-top-10-use-cases-for-machine-learning-and-ai/ [Accés: Febrer 2019]

[iii] Xarxes neuronal artificials: http://www.theprojectspot.com/tutorial-post/introduction-to-artificial-neural-networks-part-1/7 [Accés: Febrer 2019]

[iv] Estructura de capes de les xarxes neuronals artificials: https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/ [Accés: Març 2019]

[v] Capes neuronals convolucionals: https://medium.co m/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148 [Accés: Març 2019]

[vi] Bases biològiques: http://leucocitos.org/ [Accés: Març 2019]

[vii] Explicació del model Xception: https://towardsdatascience.com/review-xception-with-depthwise-separable-convolution-better-than-inception-v3-image-dc967dd42568 [Accés: Abril 2019]

[viii] Explicació del model Xception: https://www.kdnuggets.com/2017/08/intuitive-guide-deep-network-architectures.html/2 [Accés: Abril 2019]

[ix] Explicació de les xarxes convolucionals, pàgina oficial de la universitat Stanford: http://cs231n.github.io/convolutional-networks/ [Accés: Març 2019]

[x] Vista general de la Jetson Tx2: https://www.jetsonhacks.com/2017/03/14/nvidia-jetson-tx2-development-kit/ [Accés: Febrer 2019]

[xi] Introducció al Dash de Plotly: https://dash.plot.ly [Accés: Febrer 2019]

[xii] Pàgina on trobem el codi necessari per a dur a terme l'aplicació Dash: https://dash.plot.ly/dash-core-components [Accés: Febrer 2019]

Software i Hardware

[xiii] Pàgina web oficial d'Ubutnu, d'on es pot descarregar el sistema operatiu Ubuntu 16.04: http://releases.ubuntu.com/16.04/ [Accés: Febrer 2019]

[xiv] Pàgina web oficial d'Atom, d'on es pot descarregar el programa de desenvolupament Atom: https://atom.io/ [Accés: Gener 2019]

[xv] Configuració de la placa de desenvolupament Jetson TX2: https://tm3-ghost-io.cdn.ampproject.org/v/s/tm3.ghost.io/2018/07/06/setting-up-the-nvidia-jetson-tx2/amp/?usqp=mq331AQECAFYAQ%3D%3D&amp_js_v=0.1#referrer=https%3A%2F%2Fww [Accés: Març 2019]

[xvi] Pàgina on descarregar Python 3.6.6: https://www.python.org/downloads/ [Accés: Febrer 2019]

[xvii] Pàgina on descarregar el JetPack 3.2.1: https://developer.nvidia.com/embedded/downloads/archive [Accés: Febrer 2019]

[xviii] Instal·lació de Tensorflow 1.9: https://devtalk.nvidia.com/default/topic/1038957/jetson-tx2/tensorflow-for-jetson-tx2-/ [Accés: Febrer 2019]

[xix] Instal·lació de Keras 2.2: https://www.hackster.io/wilson-wang/jetson-tx2-tensorflow-opencv-keras-install-b74e40 [Accés: Febrer 2019]

[xx] Preu de la Jetson Tx2: https://store.nvidia.com/store?Action=DisplayPage&Env=BASE&Locale=es_ES&SiteID=nvidia&id=QuickBuyCartPage [Accés: Abril 2019]

[xxi] Preu del curs online: https://www.udemy.com/courses/search/?q=interactive%20python%20dashboards%20with%20plotly%20and%20dash&src=sac&kw=interactive%20Python%20Dash&p=1&price=price-paid [Accés: Abril 2019]

Bibliografia d'imatges

[xxii] Imatge de les cèl·lules de la sang: https://en.wikipedia.org/wiki/Blood#/media/File:Red_White_Blood_cells.jpg [Accés: Març 2019]

[xxiii] Imatge de model completament connectat: *https://es.wikipedia.org/wiki/Archivo:Colored_neural_network.svg* [Accés: Març 2019]

[xxiv] Imatge del esquema de la hematopoesis: https://es.wikipedia.org/wiki/Hematopoyesis#/media/File:Hematopoiesis_(human)_diagram_es.svg [Accés: Març 2019]

[xxv] Arquitectura del model preentrenat Xception: http://openaccess.thecvf.com/content_cvpr_2017/papers/Chollet_Xception_Deep_Learning_CVPR_2017_paper.pdf [Accés: Abril 2019]

Annex

A1. Codi complet de l'aplicació Dash

A continuació es presenta el codi sencer de l'aplicació realitzada. S'han mantingut, en forma de comentari, procediments anteriors per a poder visualitzar una mica més com ha estat el desenvolupament d'aquest.

```
"""Script to generate a classification of multiple images and show the images and
their classification as a gallery. The current method uses Dash with python
 (and flask) and KERAS for the classification
 To put in production do:
 gunicorn image_upload_gallery_classify:server -b :7070 -t 1000
"""

import base64
import os
# import sh
from flask import Flask

import dash
# import dash_auth # Use only as a secure option
from dash.dependencies import Input, Output, State
import dash_core_components as dcc
import dash_html_components as html
import time

 import plotly.graph_objs as go

import numpy as np
import matplotlib.pyplot as plt
from collections import Counter


## USERNAME_PASSWORD = [['marti', 'codalab']] # Use only as a secure option

# Detecting and/or making the current image path
CURRENT_FOLDER = os.path.dirname(os.path.realpath(__file__))
UPLOAD_DIRECTORY = os.path.join(CURRENT_FOLDER, 'app_uploaded_files')
# PREDICTIONS_DICT = dict()


```

```
35 #Creating image folders
36 if not os.path.exists(UPLOAD_DIRECTORY):
37     os.makedirs(UPLOAD_DIRECTORY)
38
39
40 # Normally, Dash creates its own Flask server internally. By creating our own,
41 # we can create a route for downloading files directly:
42 server = Flask(__name__)
43 app = dash.Dash(__name__, server=server)
44
45 # Use only as a secure option
46 ## auth = dash_auth.BasicAuth(app, USERNAME_PASSWORD)
47
48 # If you are assigning callbacks to components that are generated by other
49 # Callbacks (and therefore not in the initial layout), then you can suppress
50 # this exception by setting
51 app.config['suppress_callback_exceptions']=True
52
53 #_____________________________LAYOUT_____________________________
54
55 # MAIN LAYOUT
56 app.layout = html.Div([
57
58     # Banner
59     html.Div([
60         html.H2(
61             'Cellsimatic Classification App',
62             id='title'
63         ),
64         html.Img(
65             src="https://s3-us-west-1.amazonaws.com/plotly-tutorials/logo/new-
   branding/dash-logo-by-plotly-stripe-inverted.png"
66         )
67     ],
68         className="banner"
69     ),
70
71     #  Image Uploader
72     html.Div(className="container",
73         children=[
74             # html.Div([
75             #     #Clear Imagaes
76             #     dcc.ConfirmDialogProvider(
77             #         children=html.Button(
78             #             'Clear gallery'
79             #         ),
80             #         id='clearbutton',
81             #         message='"The current gallery will be removed,
```

```
82            #                   are you sure you want to proceed?'"
83            #     ),
84            #     html.Div(id='clear_output')
85            # ],
86            #     style={'float':'right',
87            #           'margin':'auto',
88            #           }
89            # ),
90            #Upload Images
91            dcc.Upload(
92            id='upload-image',
93            children=html.Div([
94               'Drag and Drop or ',
95               html.A('Select Files'),
96            ]),
97            style={
98               'width': '90%',
99               'height': '60px',
100              'lineHeight': '60px',
101              'borderWidth': '1px',
102              'borderStyle': 'dashed',
103              'borderRadius': '25px',
104              'textAlign': 'center',
105              'marginBottom':'10px',
106              'marginLeft':'50px',
107              #'margin' : 'auto',
108           },
109           # Allow multiple files to be uploaded
110           multiple=True,
111           accept='image/*'
112       ),
113
114       html.Div(id='output-image-upload'),
115       html.Div(id='tabs-content'),
116
117       ],
118           style={#'float':'right',
119                 'float':'middle',
120                 'width':'70%',
121                 },
122       ),
123
124
125 ])
126
127 #app.css.append_css({'external_url': 'https://codepen.io/chriddyp/pen/bWLwgP.css'})
128
```

```
external_css = [
    # Normalize the CSS
    "https://cdnjs.cloudflare.com/ajax/libs/normalize/7.0.0/normalize.min.css",
    # Fonts
    "https://fonts.googleapis.com/css?family=Open+Sans|Roboto"
    "https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css",
    # For production
    "https://cdn.rawgit.com/xhlulu/0acba79000a3fd1e6f552ed82edb8a64/raw/dash_template.css",
    # Custom CSS
    "https://cdn.rawgit.com/xhlulu/dash-image-processing/1d2ec55e/custom_styles.css",
]

for css in external_css:
    app.css.append_css({"external_url": css})

'''_____________________________End of Layout_____________________________________'''
'''_____________________________________________________________________________'''


'''_____________________________Image layout_____________________________________'''

## ------------------Build the gallery----------------------

def parse_contents(contents, filename, prediction):
    layout = html.Div([
        html.Div(className='gallery',
                 children=[
                    html.A(
                        target='_blank',
                        href=contents,
                        # HTML images accept base64 encoded strings in the same format
                        # that is supplied by the upload
                        children=html.Img(src=contents, width=360, height=363,
                                          style={'width': '100%',
                                                 'height': 'auto'})
                    ),
                    html.Div(filename,
                             style={'padding': 5,
                                    'textAlign': 'center',
                                    'font-size': '0.8em'}
                             ),
                    # html.Div([
                    #     html.Button(
                    #         'A',
                    #         id='img_A',
                    #         style={
```

```
175                 #            'borderRadius':'20px',
176                 #            'heigt':'100px',
177                 #     }),
178                 # ]),
179             ],
180             style={
181                 'margin': 5,
182                 'border': '1px solid #ccc',
183                 'float': 'left',
184                 'width': 150,
185                 'height': 190}
186             )
187     ])
188 
189     return layout
190 
191 
192 '''_____________________Image Uploader_____________________'''
193 
194 @app.callback(Output('output-image-upload', 'children'),
195               [Input('upload-image', 'contents')],
196               [State('upload-image', 'filename')])
197 
198 def update_output(list_of_contents, list_of_names):
199 
200     global  PREDICTIONS_DICT, LABELS
201 
202     # Delete the previous files
203     if os.listdir(UPLOAD_DIRECTORY):
204         removefiles()
205 
206     # Save all the images
207     if list_of_contents is not None:
208         for c, n in zip(list_of_contents, list_of_names):
209             save_file(n, c)
210 
211       # ==============Classification process ====================
212       ## HAY QUE CREAR LA FUNCION DE CLASIFICACION prediction_function
      CON ENTRADA LA RUTA DEL DIRECTORIO Y SALIDA LOS NOMBRES DE
      LOS ARCHIVOS Y LAS PREDICCIONES
213         #files, predictions = prediction_function(UPLOAD_DIRECTORY)
214 
215         start = time.time()
216 
217         # Dummy classification for running tests
218         # predictions = ["CELL" for t in list_of_names]
219         #files = list_of_names
```

```
220
221         # Keras classifier
222         PREDICTIONS_DICT = prediction(UPLOAD_DIRECTORY)
223
224         print("El tiempo para clasificar cada imagen es de {} s".format(time.time()-start))
225
226
227         # Make the prediction dictionary with files and predictions
228         # PREDICTIONS_DICT = dict(zip(files, predictions))
229         print(PREDICTIONS_DICT)
230
231         # ========================================================
232
233         # Unique labels
234         LABELS = list(set(PREDICTIONS_DICT.values()))
235
236
237         # Automatic Tabs infered from the LABELS
238         children = html.Div([
239                 dcc.Tabs( id="tabs", value='tab-0', children=[
240                     dcc.Tab(label='Chart', value='tab-0')] +
241                     [ dcc.Tab(label=l, value=l) for l in LABELS ] )
242                 ])
243
244         return children
245
246
247     # if list_of_contents is not None:
248     #     children = [parse_contents(c, n, predictions_dict[n])
249     #                 for c, n in zip(list_of_contents, list_of_names)]
250     #     return children
251
252
253
254
255 '''_____________________Tabs_____________________'''
256
257 @app.callback(Output('tabs-content', 'children'),
258               [Input('tabs', 'value')])
259
260 def render_content(tab):
261     if tab == 'tab-0':
262
263         # Count the frecuencies of each predicted label
264         labels, values = count_images(PREDICTIONS_DICT)
265
266         # Find the major class
267
```

```
268         valuesp = np.array(values)/sum(values) # calculate proportions
269         major = np.array(labels)[valuesp > 0.5].tolist() # Majoritary class
270 
271         if major:
272             mensaje = "Predicted diagnosis: " + major[0]
273         else:
274             mensaje = "There is not a predicted diagnosis"
275 
276 
277         out = html.Div([
278             html.H3(mensaje),
279             html.Div([
280             html.Div([
281                 dcc.Graph(
282                     figure={
283                         'data':[
284                             go.Bar(x=labels, y=values,
285                                    textfont=dict(size=20))
286                         ],
287                     'layout':go.Layout(
288                       title="Bar plot of predictions"
289                         )
290                     }
291                 )],
292                     className="six columns",
293                  ),
294               html.Div([
295                     dcc.Graph(
296                         figure={
297                             'data':[
298                                 go.Pie(labels=labels, values=values,
299                                        textfont=dict(size=20))
300                             ],
301                         'layout':go.Layout(
302                           title="Pie chart of predictions"
303                             )
304                         }
305                     )],
306                     className="six columns",
307                 ),
308         ], className="row")
309         ])
310 
311         return out
312 
313     else:
314         for l in LABELS:
```

```
315             if tab == l:
316                 return html.Div( children = show_test(tab) )
317 
318 # Forma antigua, lo que se hizo fue ahorrar codigo mediante un FOR y de paso volver
    la estructura dinamica
319  #elif tab == 'CELL':
320         #return html.Div(
321             ## html.H3('Here will be displayed the Lymphocyte images'),
322             #children = show_test(tab))
323     #elif tab == 'tab-2':
324         #return html.Div([
325             #html.H3('Here will be displayed the Variant Lymphocyte images')
326         #])
327     #elif tab == 'tab-3':
328         #return html.Div([
329             #html.H3('Here will be displayed the Atypical Lymphocyte images')
330         #])
331     #elif tab == 'tab-4':
332         #return html.Div([
333             #html.H3('Here will be displayed the Blast images')
334         #])
335 
336 
337 '''_____________________XXX_____________________'''
338 
339 #Count amount of Images
340 def count_images(PD):
341     d = Counter(PD.values())
342     labels = list(d.keys())
343     values = list(d.values())
344     return labels, values
345 
346  #save function
347  def save_file(name, content):
348     #Decode and store a file uploaded with Plotly Dash.
349     data = content.encode('utf8').split(b';base64,')[1]
350     with open(os.path.join(UPLOAD_DIRECTORY, name), 'wb') as fp:
351         fp.write(base64.decodebytes(data))
352 
353 #remove function
354 def removefiles():
355     """Remove all files inside the UPLOAD_DIRECTORY"""
356     for the_file in os.listdir(UPLOAD_DIRECTORY):
357         file_path = os.path.join(UPLOAD_DIRECTORY, the_file)
358         try:
359             if os.path.isfile(file_path):
360                 os.unlink(file_path)
361                 # elif os.path.isdir(file_path): shutil.rmtree(file_path)
```

```
362         except Exception as e:
363             print(e)
364 
365 
366 def encode_image(image_file):
367     encoded = base64.b64encode(open(image_file, 'rb').read())
368     return '{}'.format(encoded.decode())
369 
370 def show_test(clase):
371     children = []
372     for name,label in PREDICTIONS_DICT.items():
373         if label == clase:
374             # out.append(html.H4(name))
375             contents = encode_image(os.path.join(UPLOAD_DIRECTORY, name))
376             children.append(parse_contents(contents, name, label))
377     return children
378 
379 
380 if __name__ == '__main__':
381 from production import prediction
382 app.run_server(debug=True)
```