Universitat de Girona
Escola Politècnica Superior

Treball final de grau

Estudi: Grau en Enginyeria Informàtica

Títol: Desenvolupament d'algorismes que permetin fer el diagnòstic de CoVid-19 a partir d'imatges RX

Document: Memòria

Alumne: Eduard Lloret Carbonell

Tutor: Dr. Rober Martí Marly / Dr. Xavier Lladó Bardera
Departament: Arquitectura i Tecnologia de Computadors
Àrea: Arquitectura i Tecnologia de Computadors

Convocatòria(mes/any): Setembre 2021

Índex

Índex

Índex de Figures

Índex de Taules

1 Introducció

El **CoVid-19** és una malaltia infecciosa vírica ocasionada per una família vírica anomenada "Coronaviridae". El nom es deu a la similitud del virus, sota un microscopi, amb la corona solar[1]. El nom CoVid-19 prové de Corona, Virus i 2019 que amb tres s'acoten a CoVi-19 i per que per a fàcil pronunciació es modifica a CoVid-19.
Actualment hi ha diverses soques[2] d'aquest virus, d'entre les quals en podem destacar tres de majoritàries d'acord al Centers for Disease Control and Prevention[5][3]:

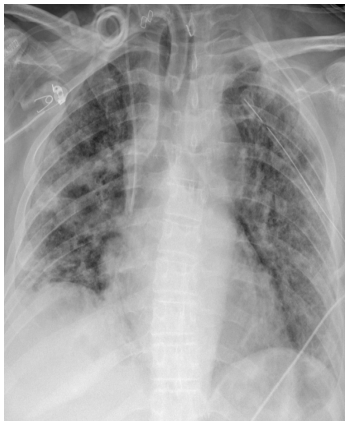
- **Variant Alfa** Originària del Regne Unit i és un 50% més contagiosa que la original.
- **Variant Beta** Originària de Sud-àfrica i més contagiosa que la original i relativament més resistent a la immunitat al virus original.
- **Variant Gamma** Originària del Brasil és també més transmissible i amb una resistència moderada a la immunitat.
- **Variant Delta** Originària de l'Índia és un 60% més contagiosa que la Alfa i té una resistència feble a la immunitat.

El CoVid-19 és i ha sigut una de les majors malalties víriques de les últimes dècades. Tot i que no té una de les mortalitats més altes, és de les més contagioses en diferència. El que fa que sigui molt important poder detectar-la el més ràpid possible per a poder aïllar correctament el brot.

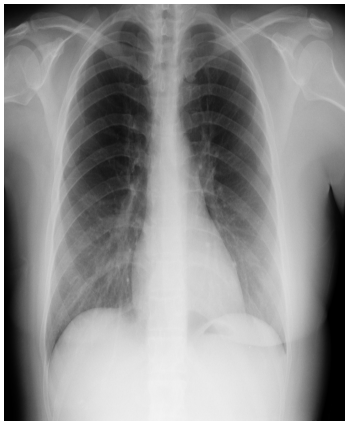
En aquest punt entren les **radiografies de tòrax**[4] les quals són un recurs ràpid i molt eficaç que serveix per a analitzar la situació interna d'un pacient. També és un recurs quasi il·limitat i molt barat donada l'estructura d'un hospital actual en els països més desenvolupats. Gràcies a aquests avantatges, s'han convertit en el recurs més utilitzats tant en el diagnòstic del CoVid-19 com en l'exploració de l'evolució del pacient quan ingressa a l'hospital.

[1] Part més externa de Sol, l'estrella que conforma el sistema solar
[2] Variants generades a partir de mutacions sobre un virus principal.
[3] CDC: Centers for Disease Control and Prevention.
[4] Una radiografia de tòrax és una tècnica que usa una variant de rajos Ultraviolats anomenats X per a crear imatges bidimensionals de la part toràcica d'un pacient

En les següents Figures 1.1 es mostren tres imatges de diferents exemples que ens podem trobar quan tractem amb radiografies de tòrax.



(a) Pulmons amb pneumonia CoVid-19



(b) Pulmons sense patologies

(c) Pulmons amb pneumonia no-CoVid-19

Figure 1.1: Comparació entre els canvis de pulmons segons la patologia

Els **radiòlegs**[5] són els encarregats d'analitzar les imatges radiogràfiques. El problema és que per convertir-se en un radiòleg experimentat es necessiten molts anys d'experiència, ja que és necessari poder distingir correctament les diferents patologies dins d'una imatge radiogràfica. Molts cops, l'anàlisi d'una imatge radiogràfica requereix un temps considerable que ni els radiòlegs més especialitzats poden evitar.

En aquest punt és quan entren els sistemes d'intel·ligència artificial, aplicats a l'anàlisis de radiografies, en joc. Aquests sistemes no només poden ajudar a obtenir resultats en un període de temps molt més breu sinó que també, poden ajudar als radiòlegs a obtenir informació que és molt difícil veure a ull nu. D'aquesta manera augmentant la capacitat d'aconseguir un anàlisi considerablement més ràpid.

Per aquesta raó, en aquest TFG s'intentarà analitzar, estudiar i desenvolupar un programa informàtic, que intenti predir si el pacient està afectat pel virus CoVid-19. I finalment un altre programa que intenti predir com es desenvoluparà la malaltia dins el pacient utilitzant imatges de Raig-X de tòrax . Per a poder d'aquesta manera ajudar a la detecció d'aquest virus i predir com es desenvoluparà dins del pacient.

Per a poder dur a terme aquesta tasca, utilitzarem tècniques de **"Deep Learning"** un subgrup dins dels algorismes de l'Intel·ligència Artificial[6].

[5]Metges especialitzats en el diagnosis d'imatges radiogràfiques, d'ultrasons i qualsevol altre imatge imatge mèdica.

[6]L'Intel·ligència Artificial és l' intel·ligència expressada per màquines. I aquesta es dona quan aquestes

El "Deep Learning" o aprenentatge profund són un conjunt de tècniques o algorismes que aconsegueixen dur a terme tasques molt diverses i necessiten un gran poder computacional per a poder representar el món com un conjunt de conceptes jeràrquics, on cada concepte està definit per conceptes més simples, i representacions més abstractes processades utilitzant-ne de menys abstractes[18].

En els últims anys aquestes tecnologies basades en "Deep Learning" s'han anat incorporant en el sector mèdic amb bons resultats. Sobretot durant aquest últim any i mig, des de l'aparició de la pandèmia, ha sigut quan el nombre d'aquestes tecnologies implementades en el sector mèdic s'ha disparat. El quals han ofert als doctors una nova manera més ràpida i amb una major precisió d'analitzar les imatges.

1.1 Motivacions

Des del meu ingrés a la universitat sempre he tingut els ulls posats en la informàtica basada en el sector mèdic. Per aquesta raó vaig cursar l'itinerari de Computadors on hi ha l'assignatura de visió per computador i la d'anàlisis i processament d'imatges.

Quan vaig voler seleccionar un treball vaig anar a demanar opinió als professors encarregats del departament de visió mèdica de la UdG. Els quals em van oferir la proposta d'aquest treball, el qual s'adheria tant al meu àmbit de recerca, amb col·laboracions amb l'hospital Dr. Josep Trueta, i al meu interès personal.

1.2 Propòsit i objectius del Projecte

El propòsit d'aquest projecte, **és fer un programa que donada una imatge radiològica dels pulmons d'una persona es pugui, a partir de la informació donada, detectar amb un percentatge important si el pacient té o no CoVid-19. I també que es pugui intentar predir si aquesta persona empitjorarà i en quin grau ho farà**.
Per tant els Objectius a seguir seran els següents:

1. Aprendre els conceptes sobre el "Deep Learning" i com s'implementa.
2. Analitzar les dades, pertanyents a la base de dades **BIMCV-COVID19**, constituida per un aproximat de 45000 casos, de "Medical Imaging Databank of the Valencia Region" [3]. I posteriorment fer triatge i modificacions sobre elles.
3. L'anàlisi d'un programa que donades imatges radiològiques d'un pacient es pugui detectar si té o no CoVid-19.

màquines poden evaluar el seu entorn i augmentar les possibilitats d'èxit en un objectiu o treball

4. Crear un nou programa que pugui a partir de la mateixa imatge donar-nos informació de com evolucionarà el pacient.
5. Avaluació exhaustiva dels resultats amb les dades analitzades i quantificació numèrica.
6. Elaboració de la memòria del projecte explicant els estudis fets i els resultats obtinguts.

2 Metodologia

La **metodologia** usada en aquest treball, al ser un treball de recerca, s'ha basat més en una metodologia de desenvolupament que en una metodologia de programació pura. Per tant com a base s'ha seguit un mètode de desenvolupament iteratiu basat en hipòtesis, objectius i tasques incloent l'avaluació quantitativa.
Cal destacar que en aquesta secció no s'explicarà quins programaris s'han usat ni com és el seu funcionament, tota aquesta informació estarà dins de la **Secció 5(Estudis i decisions)**.

2.1 Preparació d'un entorn de treball

El primer i més important per a poder dur a terme aquest treball ha sigut buscar un bon entorn de treball.
Primerament, s'ha d'elegir a on dur-lo a terme, en aquest cas en un servidor hem pensat que seria la millor opció, tant per privadesa de dades com per les necessitats de còmput que requereix aquest projecte.
Seguidament, s'havia de triar quins programaris o llibreries s'han de fer servir per a dur a terme el projecte. Aquest pas és molt rellevant, perquè diferents llibreries o programaris, et donen certs avantatges però també et donen certs inconvenients.
El tercer pas és fer una avaluació de les llibreries i programari usat, per d'aquesta manera, trobar incompatibilitats o manca de llibreries que en un principi no es trobaven necessàries.
Finalment l'últim pas en la preparació és aprendre a usar totes les funcions de les llibreries que s'utilitzaran per a dur a terme el projecte.

2.2 Creació del projecte

El primer pas per dur a terme aquest projecte és tenir en ment quina arquitectura es vol utilitzar i per a què. Un cop sabem quina arquitectura s'utilitzarà, ara és el moment de tractar les dades per a poder utilitzar-les posteriorment com a inputs de l'algorisme. Seguidament s'ha testejar l'algorisme amb diferents paràmetres, per a intentar millorar el resultat o intentar buscar un nou enfoc o solució. D'aquesta forma intentar millorar l'encert de l'algorisme desenvolupat.
Finalment repetirem el tractament de dades i el canvi de paràmetres reiteradament fins que trobem un resultat raonable o no trobem més solucions al problema.

3 Planificació

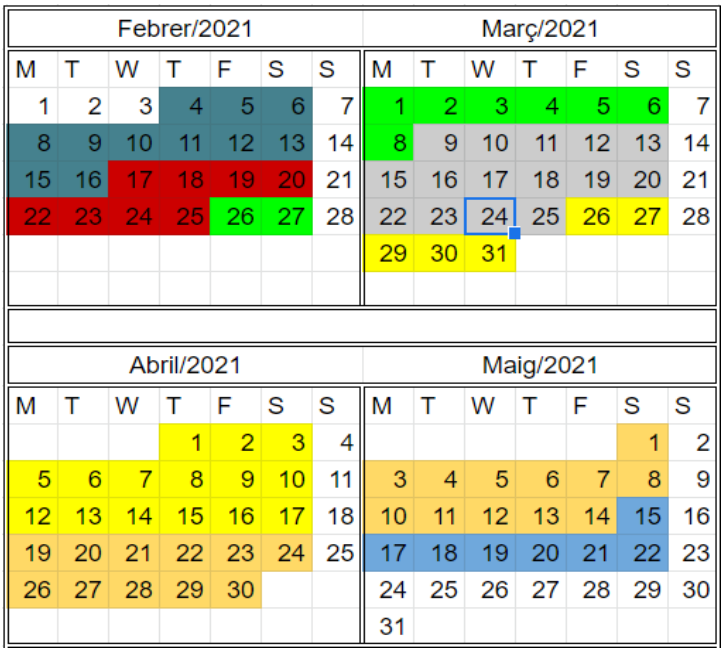
En aquest apartat es mostrarà com s'ha planificat el temps del treball realitzat. Primerament mostraré gràficament la distribució dels dies utilitzats i seguidament donaré una breu explicació de cada tasca i quins objectius s'hi esperen.

3.1 Distribució dels dies

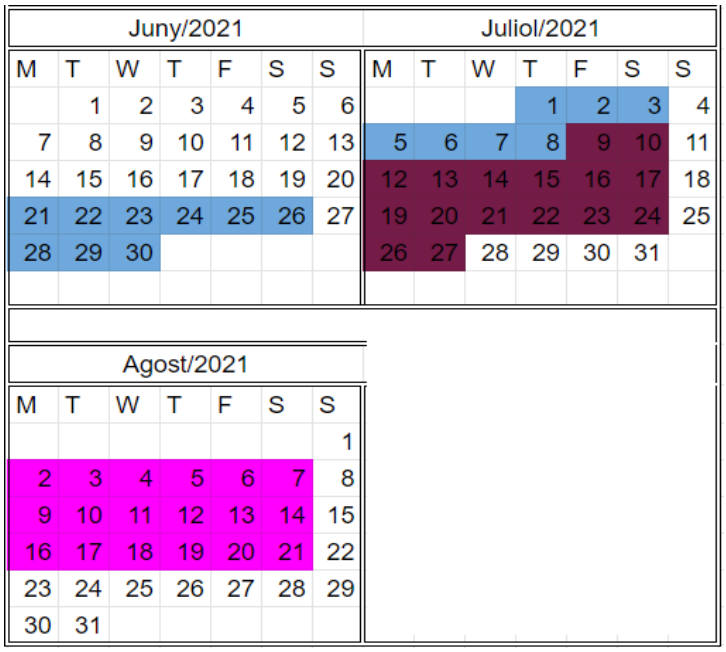
En aquest apartat mostraré la distribució del treball durant els mesos treballats:

Tasca	Color
Posta a punt del programari i diverses proves	
Aprenentatge Deep Learning amb FastAI 2	
Aprenentatge PyTorch	
Tractament de les Dades	
Implementació mètode Binary CoVid/no-CoVid	
Implementació mètode multiclasse	
Implementació mètode predictiu	
Valoració de resultats	
Redacció de la memòria	

Table 1: Tasques fetes durant el projecte



(a) Planificació Mesos 2-5 CoVid-19



(b) Planificació mesos 6-8

Figure 3.2: Planificació Projecte

3.2 Descripció de les tasques

En aquest punt descriuré cadascuna de les tasques i quins objectiu s'espera obtenir. També especificaré el temps que s'ha necessitat per a dur a terme cada tasca.

3.2 Descripció de les tasques

3.2.1 Posta a punt del programari i diverses proves

Temps necessitat: 2 setmanes
Durant aquesta tasca el que s'ha fet ha sigut buscar quins llenguatges, llibreries i programari es necessiten per a dur a terme el projecte. I posteriorment instal·lar-ho en l'"enviroment" dins del servidor usat.

3.2.2 Aprenentatge Deep Learning amb FastAI 2

Temps necessitat: 1.5 setamanes
Durant aquesta tasca s'ha aprés a com utilitzar els recursos dins la llibreria de FastAI i com crear de manera simple models, entrar les dades als models i veure com progressa l'aprenentatge. També gràcies als meus tutors, s'han tingut exemples ja implementats de diverses **xarxes neuronals** [7] els quals s'han usat per a aprendre com és la estructura de implementació d'aquests algorismes.

3.2.3 Aprenentatge amb PyTorch

Temps necessitat: 1.5 setmanes
Un cop apreses les bases de les Xarxes neuronals amb FastAI[13], s'ha volgut utilitzar llibreries una mica a més baix nivell per a dur a terme aquest projecte. Per aquesta raó, durant aquest lapse de temps, s'ha aprés PyTorch que juntament amb FastAI depenen del llenguatge Python. També s'ha aprés a com modificar els inputs, les imatges entrants, per així intentar millorar en certs casos l'aprenentatge i a més a més fer **Fine tuning** [8] del model amb el que es treballa. També com a punt menys important s'ha aprés a com funcionen a baix nivell les xarxes neuronals.

3.2.4 Tractament de Bases de dades

Temps necessitat: 2 setmanes
I finalment abans de començar a entrenar i perfeccionar l'entrenament de les diverses xarxes, s'han de tractar les dades d'entrada. Per tant durant aquestes dues setmanes el que s'ha fet ha sigut triar dades modificar-les, separar-les en grups i triar quines **etiquetes**[9] són rellevants i quines no. Cal destacar que aquest procés també s'ha dut a terme en part abans de la implementació del segon i tercer mètode.

[7] Tècnica que pertany al grup d'algorismes del "Deep Learning" i intenta simular un model simplificat d'un cervell humà[11].
[8] Modificar els paràmetres d'una xarxa que poden millorar o empitjorar tant l'acert com el temps d'entrenament
[9] Atributs booleans que representen diferents aspectes de la imatge

3.2.5 Implementació mètode Binary CoVid/no-CoVid

Temps necessitat: 3 setmanes
En aquest pas és quan ja es comença l'entrenament de les xarxes neuronals, el que es fa és provar diversos tipus de xarxes neuronals i fer un "Fine Tuning" per a trobar quins són els millors paràmetres per a determinar si en una imatge s'hi detecten o no indicis de CoVid-19. I finalment obtenir uns resultats que en passos posteriors tractaré.

3.2.6 Implementació mètode multiclasse

Temps necessitat: 4 setmanes
Aquesta pas es basa en poder detectar amb Xarxes neuronals quines patologies dins d'un subgrup donat (pneumònia,atelèctasis,..) pateix el pacient. I en aprendre a com implementar xarxes neuronals amb diverses etiquetes d'entrada i de sortida.

3.2.7 Implementació mètode predictiu

Temps necessitat: 4 setmanes
Finalment l'últim pas de l'entrenament es basa en aprendre a implementar un nou tipus de Xarxa neuronal que està conformada per diverses xarxes neuronals més petites per a poder recrear imatges noves a partir d'una imatge donada. I seguidament entrenar-la per a poder predir com respondrà el pacient a la malaltia.

3.2.8 Valoració de resultats

Temps necessitat: 3 setmanes
En aquest pas es valoraran els resultats obtinguts de les xarxes neuronals que s'han entrenat i s'intentarà donar conclusions sobre els resultats trobats.

3.2.9 Redacció de la memòria

Temps necessitat: 3 setmanes
Per acabar, es fa la redacció de la memòria a partir dels resultats que anteriorment s'han anat generant i les tasques que s'han dut a terme.

4 Marc de treball i conceptes previs

En aquesta secció s'intentarà introduir tots els temes que han sigut necessàri aprendre durant la realització d'aquest projecte. Seguint un ordre similar a l'ordre que segueix el treball.

4.1 Que és la IA o Intel·ligència artificial

El 2004 es va escriure un article que responia moltes de les preguntes relacionades amb la Intel·ligència artificial entre elles i la primera "What is artificial intelligence?" , el qual en Català és el títol d'aquest apartat. La definició que ens dona és: **«És la ciència i la enginyeria de fer màquines intel·ligents, més concretament programes intel·ligents. És similar a la tasca d'entendre l'intel·ligencia humana utilitzant ordinadors, només que no estem limitats a mètodes biològicament observables»**[19].
El que ens ve a dir és que la intel·ligència artificials no és res més que la creació d'algorismes informàtics que simulen la intel·ligència humana, però amb l'avantatge de no estar limitats per el propi funcionament del cervell humà.
Al no estar limitats per res més que la pròpia capacitat d'invenció humana i la capacitat de còmput actual, han anat apareguen noves tècniques cada vegada més complexes basades en el concepte d'Intel·ligència Artificial. I tal com podem veure en el gràfic següent les patents relacionades en IA s'han disparat en els últims anys, vegeu la Figura 4.3.

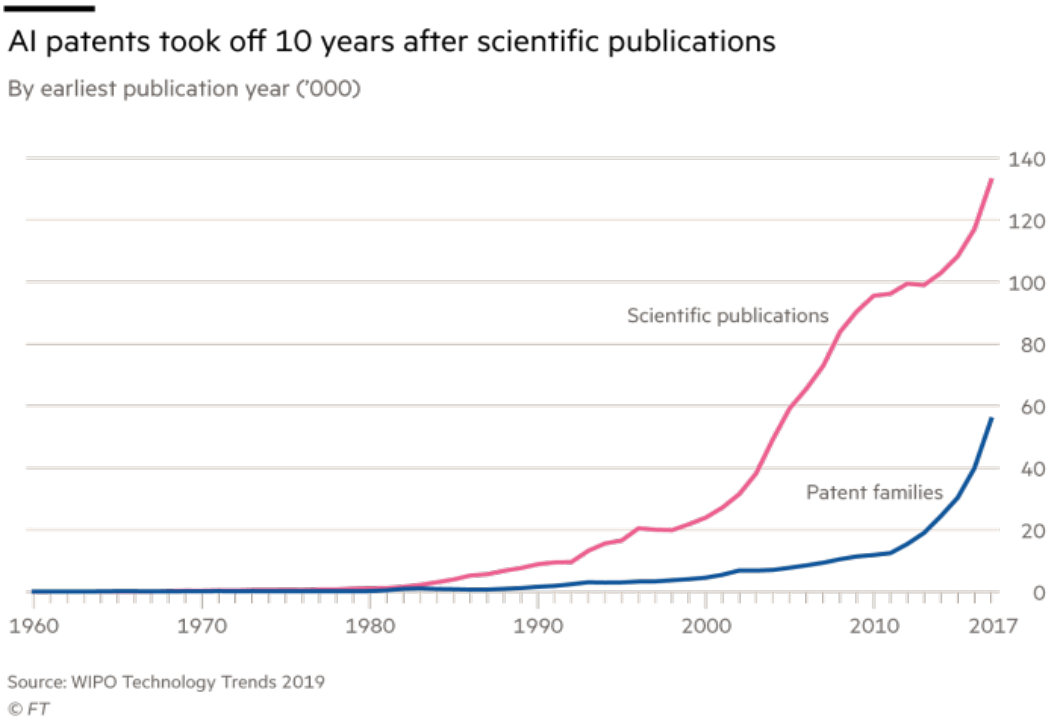


Figure 4.3: Patents de tecnologies basades en AI segons la WIPO(Organització Mundial de la Propietat Intel·lectual) [23]

4.2 Branques de l'Intel·ligència artificial

Dins del conjunt d'algorismes que conformen l'IA, en podem extreure 7 grans branques, tal com es mostren a la Figura 4.4. On cadascuna té el seu propòsit en específic **tot i que molts cops els algorismes no estan només definits dins d'una sola d'aquestes branca.**

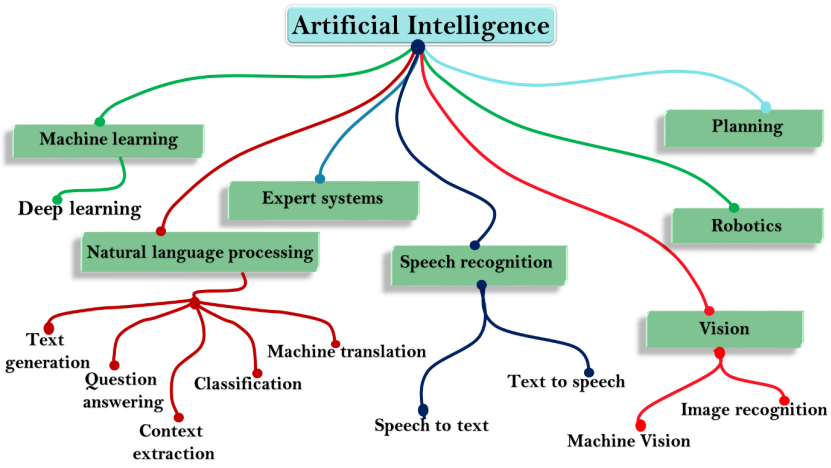


Figure 4.4: Les 7 grans branques de la IA

1. **L'Aprenentatge automàtic(Machine Learning)**:És el conjunt d'algorismes que per si mateixos poden millorar el seu rendiment.

2. **Visió Artificial (Vision)**: És el conjunt algorismes que produeixen informació numèrica o simbòlica a partir del processament i l'anàlisis d'imatges.

3. **Sistemes Experts (Expert Systems)**: Són els algorismes que intenten emular el raonament humà en una àrea concreta.

4. **Reconeixement de veu (Speech Recognition)**: Són els algorismes que intenten desxifrar la parla humana.

5. **Robòtica (Robotics)**: Són els algorismes que intenten emular diferents característiques humanes utilitzant un robot com a medi d'interacció.

6. **Planificació (Planning)** : Són els algorismes que busquen solució a problemes logístics.

7. **Processament del llenguatge natural (Natural language processing)**: Són els algorismes que intenten respondre a problemes humans amb respostes humanes.

En aquest treball només veurem part de **visió** i part d'**Aprenentatge automàtic**, totes les altres branques no són necessàries per al desenvolupament d'aquest projecte per tant seran obviades.

4.3 Visió Artificial

La **Visió Artificial** és el conjunt d'algorismes que, intenten imitant la visió humana, adquirir, processar, analitzar i comprendre les imatges del món real amb la fi de produir una informació numèrica o simbòlica que pugui ser usada per la màquina.
Les tècniques i algorismes dins de l'Aprenentatge automàtic que treballen sobre imatges, necessiten sempre una part de Visió Artificial per a poder extreure característiques de la imatge que posteriorment, la màquina, podrà utilitzar per a detectar patrons.
Un bon exemple és el **SIFT (Scale invariant feature transform)**, un algorisme que extreu característiques invariants a l'escala d'una imatge, per identificar si aquesta imatge està dins d'una altre[2].

4.4 Aprenentatge automàtic

L'**Aprenentatge automàtic** es refereix al conjunt d'algorismes que són capaços d'aprendre i adaptar-se sense la necessitat de seguir unes instruccions predefinides, buscant o inferint patrons en les dades que rep com a input.
Aquest subgrup d'algorismes intenten imitar el comportament d'aprenentatge humà el qual es basa en 3 simples passos: **Observar, aprendre i utilitzar el coneixement adquirit**. D'aquesta manera poder solucionar problemes que en termes informàtics poden ser molt complexos com els **NP-Hard**[10], el reconeixement de patrons o l'automatització de mètodes científics.
Un bon exemple és el **TR10**, una IA que valida els efectes mèdics de noves molècules que poden ser usades en el món mèdic. Utilitza una base de dades existent d'altres molècules ja provades per a inferir noves molècules que puguin tenir efectes beneficiosos pel cos humà[24].
Tot algorisme o tècnica dins de l'Aprenentatge automàtic necessita d'un etapa d'aprenentatge per a poder aprendre a detectar els patrons que es troben dins al informació. Se'n poden veure dos tipus principals d'aprenentatges: l'aprenentatge supervisat i l'aprenentatge no supervisat.

4.4.1 Aprenentatge no supervisat

L'**Aprenentatge no supervisat** és el conjunt d'algorismes que aprenen a trobar patrons en dades no etiquetades. O sigui que l'algorisme durant l'aprenentatge no sap en quin grup està classificat cada element i per si mateix fent inferència ha de trobar patrons en les dades donades

[10]Són un conjunt de problemes que tot problema NP pot ser reduït a ells, on NP són els problemes que es poden solucionar en temps polinòmic amb una màquina no determinista

i classificar-les.
L'objectiu d'aquest algorisme és trobar si existeixen relacions entre les dades donades i seguidament poder-es agrupar o classificar segons a quin subgrup pertanyen (p.e. Tenim un grup de fruites i volem trobar com agrupar-les). Vegeu vegeu la Figura 4.5

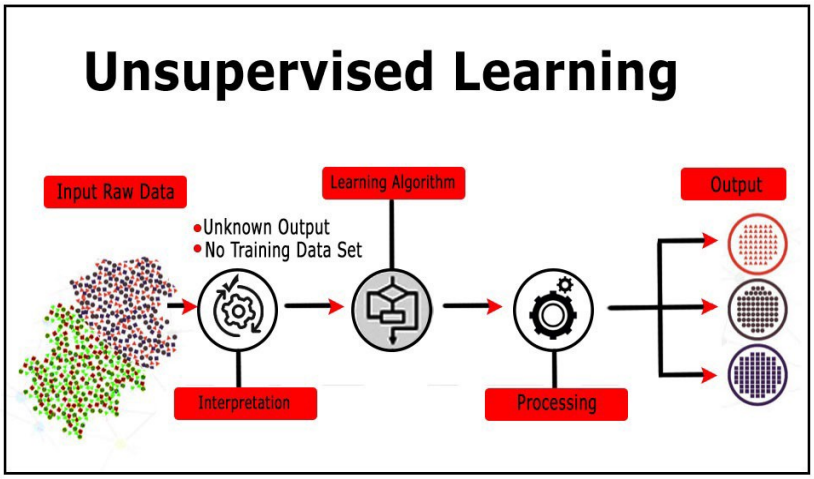


Figure 4.5: Estructura d'un algorisme basat en aprenentatge no supervisat [16]

L'algorisme **K-Means** és un exemple d'algorisme d'aprenentatge no supervisat que està situat dins el subgrup "Clustering" o en Català "Agrupar". El funcionament és simple, es transformen les dades en punts a l'espai on cada atribut és un valor axial, seguidament s'indica quants grups es volen generar i ell sol trobarà la solució quasi-òptima de com agrupar-los.

4.4.2 Aprenentatge supervisat

L'**Aprenentatge supervisat** és un grup d'algorismes que aprenen utilitzant un conjunt de dades etiquetades a classificar dades o a predir resultats[12].
Contràriament a l'aprenentatge no supervisat que totes les dades s'entren al mateix moment, en l'aprenentatge supervisat les dades són introduïdes en paquets i l'algorisme va ajustant els seus paràmetres interns fins que el seu entrenament ha sigut satisfactori. Podríem descriure l'aprenentatge no supervisat com la lògica humana sense coneixements previs i l'aprenentatge supervisat l'estudi i posteriorment l'aplicació dels coneixements estudiats.
Tal com un humà l'aprenentatge supervisat necessita un estudi previ abans de dur a terme les tasques desitjades per tant podem separar-lo en dos fases: Entrenament i Aplicació. Vegeu la Figura 4.6 per veure l'estructura.

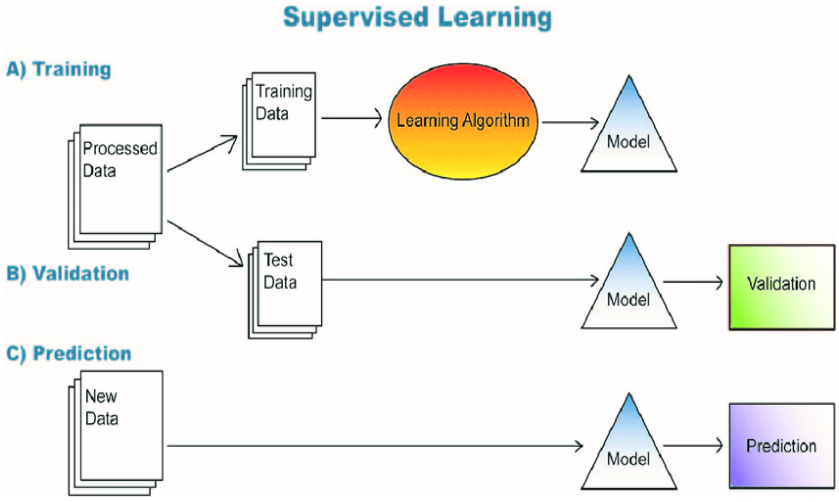


Figure 4.6: Estructura d'un algorisme basat en aprenentatge supervisat

EL **Naïve Bayes** és un exemple d'algorisme d'aprenentatge supervisat on s'utilitzen els principis d'independència condicional. L'algorisme busca a partir dels paràmetres de la distribució normal trobats durant l'entrenament, la probabilitat que un input pertanyi a un dels tipus coneguts per l'algorisme i finalment el que té més probabilitat és el seleccionat.

4.5 Deep Learning

El **Deep Learning** és un subgrup dins de l'Aprenentatge automàtic, en la majoria de casos d'aprenentatge supervisat. Es destaca per no necessitar un humà que extregui o filtri les característiques importants. L'algorisme per si sol extreu, fa un filtratge de les dades i posteriorment les estructura sens ela necessitat d'interacció externa.
Aquests algorismes poden rebre com a input dades no estructurades com blocs de text o imatges i per si sols poden determinar quines són les característiques principals i quines no son rellevants[10].
Un exemple seria etiquetar imatges d'animals segons quin animal hi aparegui. Durant el procés d'entrenament aquest algorismes buscarien patrons o característiques a la imatge (la forma de les orelles, quants ulls té,...) sense la necessitat d'un humà que les ressalti. Mentrestant en els altres algorismes dins de l'Aprenentatge automàtic era quasi sempre necessari un humà que estructura les dades de manera que l'algorisme les pogués entendre.

Gràcies al **Deep Learning** ja no estem limitats a petits grups de dades que han de ser prèviament processats per humans amb anterioritat. Ara podem de manera fàcil es pot evocar grans Bases de dades com a input del nostre model sense problema. D'aquesta manera es poden fer models molt més complexos i amb un ventall molt més grans de casos.
Tot i això té o tenia dos grans problemes que amb l'avenç de la tecnologia han anat disminuint:

1. **Necessita grans quantitats de dades**, amb l'aparició del que s'anomena **Big Data**[11]

[11]Grans bases de dades que s'an anat publicant i ara s'emmagatzemen en grans servidors

aquest problema ha anat desapareixent.

2. **Necessita un "gran" poder de còmput**, aquest punt és molt depenent de la mida del nostre model i el nivell de perfecció que volguéssim arribar. Però en l'actualitat amb un hardware relativament econòmic podríem aconseguir uns resultats decents.

4.6 Xarxes neuronals

Les **Xarxes neuronals** són algorismes informàtics que intenten simular el funcionament de les neurones del cervell dels animals. On a partir d'un estímul principal s'envien senyals elèctrics que passen a través de les neurones i amb uns pesos s'estimula una sortida concreta.

4.6.1 Parts d'una Xarxa neuronal

Les Xarxes neuronals es basen en una col·lecció de nodes que s'anomenen neurones artificials i una interconnexió entre aquest nodes que simula les **dendrites**[12] de les neurones.
Oposadament al cervell humà les xarxes neuronals utilitzen un nombre normalment en **coma flotant**[13] per a transmetre la senyals. Cada connexió de les xarxes també li correspon un pes, també en FP32, que es modifica no lineament segons l'entrenament.

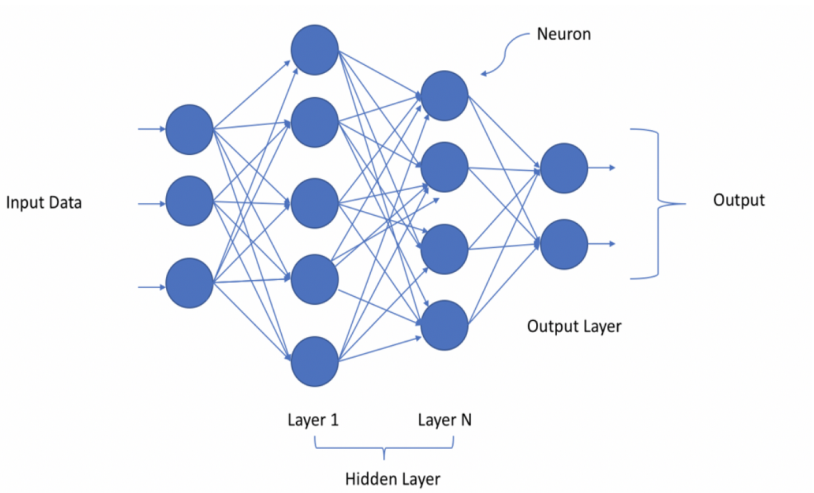


Figure 4.7: Arquitectura d'una Xarxa neuronal

Tal com podem apreciar a la Figura 4.7 tenim 2 components que conformen la Xarxes neuronals que estan dividits en 3 capes ("Layers").

Components

[12]Són les parts de les neurones encarregades de rebre les senyals electriques
[13]Forma de notació científica usada en els ordinadors per a representar nombres reals amb molta precisió de manera eficient i compacta

- Neurona: Són els cercles pintats de blau en l'imatge, intenten representar una neurona humana.
- Connexions: Són les connexions entre les neurones, les connexions són 1xN on N és el nombre de neurones del següent "Layer". El camí es decidirà segons el valor d'entrada multiplicat per el pes que tenen aquestes connexions. Cal ressaltar que mai en cap cas una connexió d'una neurona no anirà a cap altre neurona que no sigui a les de la següent capa.

Capes

- Capa d'entrada ("Input Layer"): És el conjunt de neurones que reben l'input d'entrada del model.
- Capes intermèdies o amagades ("Hidden Layer"): Són el conjunt de capes que estan situades entre la Capa d'entrada i la Capa de sortida i serveixen per augmentar la precisió del model.
- Capa de sortida ("Output Layer"): És el conjunt de neurones que donen el valor de sortida del model, si normalitzem els valors de sortida de cadascuna de les neurones en aquesta capa, obtindrem en quina probabilitat l'entrada pertany a cadascuna de les classes de sortida.

4.6.2 Com aprèn la Xarxa neuronal?

La xarxa per aprendre utilitza un mètode anomenat Propagació cap endarrera (Backpropagation) , aquest mètode a grans trets es basa en 4+"N" iteracions on N=(nº capes de la xarxa-1).

1. Executar el model per un conjunt d'inputs.
2. Calcular l'**error quadràtic**[14] entre les etiquetes i les sortides del model.
3. Fer N iteracions, on a cada iteració es calculen les derivades parcials entre un parell conseqüent de capes (N amb N-1, N-1 amb N-2,...,1 amb 0) respecte l'error. Finalment multipliquem el gradient obtingut per un valor que decideix el programador anomenat "Learning Rate".
4. Utilitzem els valors obtinguts en el pas anterior incrementant els pesos per el valor obtingut respectivament.
5. Tornem a executar el programa si volem millorar el model.

Tal com es pot veure és un mètode on el "Learning rate" indica com de ràpid entrenarà la xarxa per tant aquest paràmetre és important que sigui adequat.

[14]Funció d'error que té com a resultat la diferencia al quadrat entre el valor correcte i l'obtingut

4.6.3 Learning Rate

Tal com s'ha descrit el **"Learning Rate"** representa la velocitat amb la qual la xarxa aprèn, per tant el primer que ens vindria en ment seria establir-lo al màxim. Però si observem la Figura 4.8 veurem que si l'incrementem massa la funció divergirà .

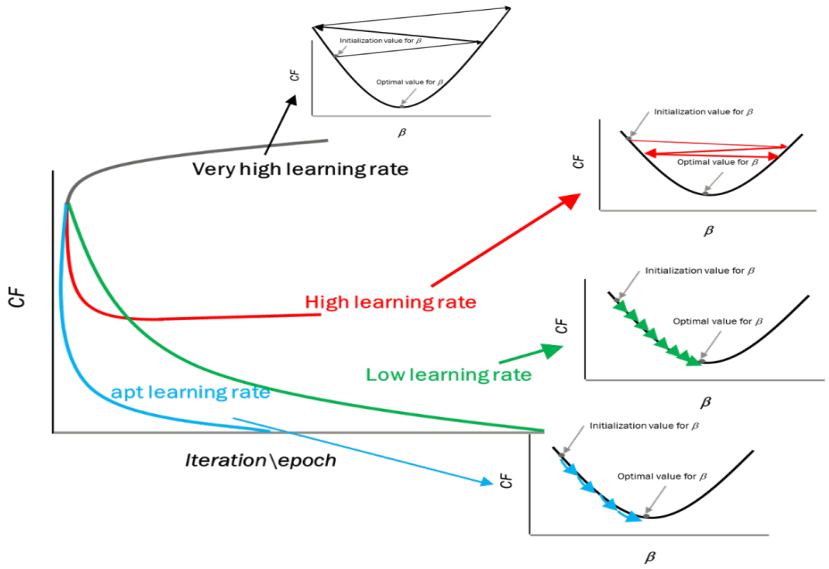


Figure 4.8: L'efecte dels diversos valors del Learning Rate en l'aprenentatge del model [14]

Tal com podem apreciar hi ha 4 casos possibles però en grans termes es pot resumir en que si el "Learning Rate" és massa alt causarem la funció d'actualització dels pesos sortir-se de rang o en els millors dels casos quedar-se en un punt.
Per altre banda si és massa baix es necessitaran un nombre molt més alt d'iteracions d'aquesta manera necessitant molt més temps d'entrenament. Per tant cal aconseguir trobar el Learning Rate adequat per tal que no oscil·li l'error però el nombre d'iteracions sigui adequat.

4.6.4 Xarxes neuronals convolucionals

Finalment un cop introduïdes totes les bases presentaré un dels dos algorismes d'Aprenentatge automàtic que s'utilitzaran per a dur a terme aquest treball les **CNN(Convolutional Neural Networks)** o en Català **Xarxes neuronals convolucionals**. Són la fusió entre dos grans branques de la IA [4.2], la visió artificial i l'Aprenentatge automàtic.
Les CNN són una variant de les Xarxes neuronals que està formada per dos majors parts: Les capes convolucionals i les capes totalment connectades.

Capes Convolucionals

Són les capes de la Xarxa Neuronal encarregades de obtenir les característiques i patrons dins de la imatge utilitzant diferents filtres i funcions. Aquest conjunt de capes en termes generals rep com a entrada una imatge de 3 dimensions la qual se li aplica l'operació de **Convolució** amb una **Màscara** de mida (NxN). Seguidament se li aplica una funció **ReLu** i finalment una funció de **Pooling**.

Kernel o Màscara:

És una matriu (NxN) conformada per uns valors que serviran com els pesos de l'algorisme.

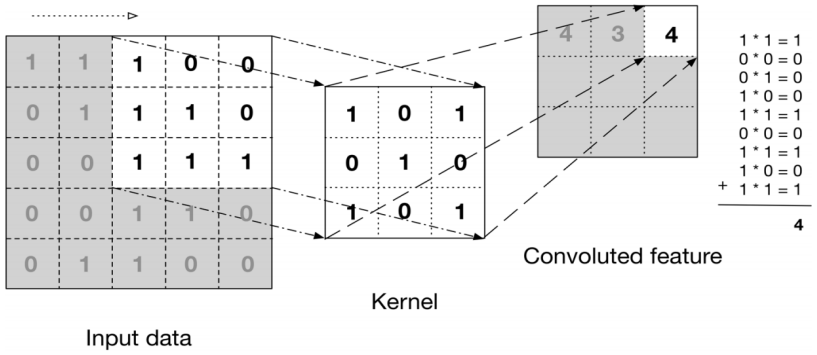


Figure 4.9: Aplicació de la operació de convolució amb un kernel o màscara a una matriu [28]

Tal com podem veure a la Figura 4.9 la màscara és una matriu sempre de mida inferior a la de la matriu a la qual s'aplica la convolució.

Què és una convolució:

La convolució està definida pel símbol "*" i representa el producte de dos funcions dins el domini freqüencial. No és necessàri saber exactament que és el domini freqüencial però en poques paraules és el rati de canvi dels píxels en una imatge o sigui els **gradients**[15] dels píxels que conformen l'imatge. El problema que representa el domini freqüencial és que és molt costós la conversió d'una imatge a domini freqüencial i fer la transformació inversa per tornar a obtenir una imatge en **RGB**[16]. Degut això van aparèixer les convolucions, processos matricials, que simplifiquen molt el cost computacional i obtenen el mateix resultat que aplicar un producte entre funcions en l'espai freqüencial.

[15] Com de gran és el canvi entre el píxels veïns

[16] Representació d'una imatge amb els tres colors primaris Vermell, Verd i Blau

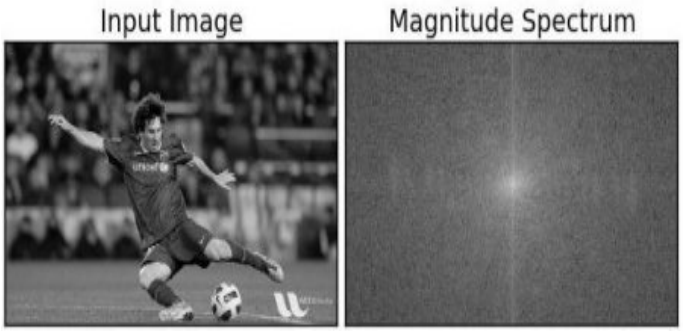


Figure 4.10: Imatge en espai RGB a l'esquerra i la mateixa imatge en espai freqüencial a la dreta[21]

Tal com podem veure a la Figura 4.10 una imatge en l'espai RGB no s'assembla en res en una imatge en l'espai freqüencial. Una imatge en l'espai RGB ens mostra una imatge similar a les que captem els humans a la vida real, oposadament una imatge en l'espai freqüencial ens mostra la distribució dels gradients. Els punts blancs que estan situats en la imatge de la dreta representen els gradients que trobem. Com més allunyats estan aquests punts del centre vol dir que el gradient és més gran i com més el centre de la imatge vol dir que le gradient és mes petit. Gràcies a aquesta relació podem fer uns filtres molt especials que veurem a la Figura 4.11 :

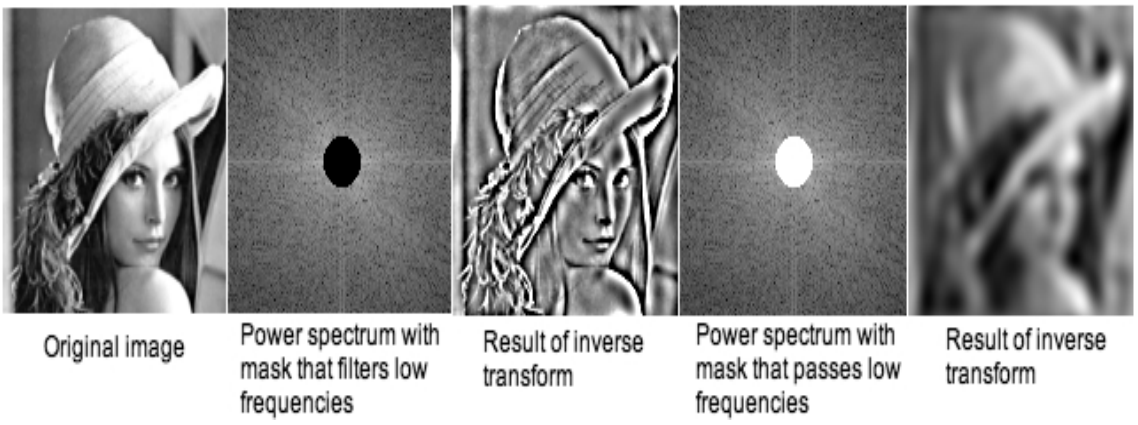


Figure 4.11: Com afecten les operacions en l'espai freqüencial en una imatge[21]

Tal com podem veure el primer filtre exclou els píxels de baixa freqüència per tant ressalta els contorns de les figures dins de la imatge, contràriament la segona exclou els píxels de alt gradient per tant esborra els marges creant una imatge més difuminada.

Operació de convolució:
L'operació de convolució és molt simple, es basa en lliscar una Màscara 4.6.4 per tots els píxels de l'imatge i fer el producte matricial entre la màscara i la porció de l'imatge la qual està situada just a sota. En el cas que la imatge sigui de més de 2 dimensions es sumaran els resultats

d'aquesta aplicació per cadascun dels nivells de profunditat que tingui. I finalment és crearà com a resultat una matriu nova dels valors obtinguts.

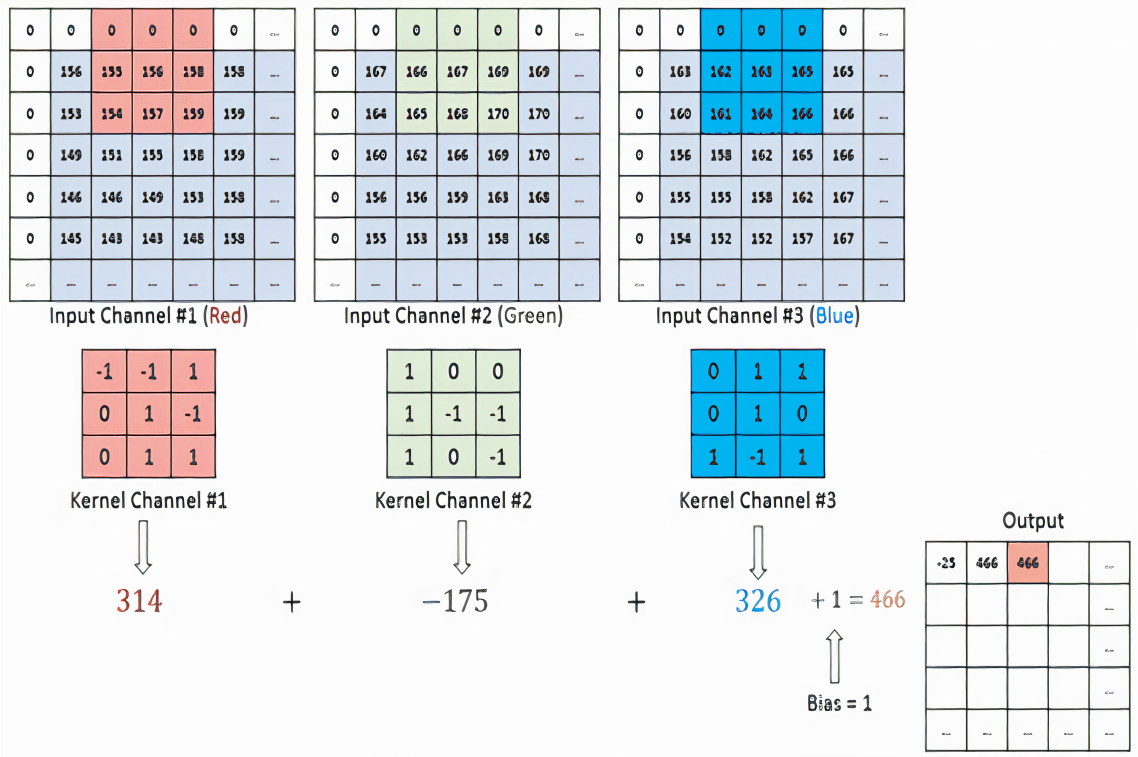


Figure 4.12: Operació de convolució en una imatge 3D[25]

Un bon exemple és el que veiem en la figura 4.12, fixem-nos en la primera matriu que representa el color Vermell d'una imatge. Actualment ja estem situats en el tercer píxel, el que fem és posar la màscara en el píxel tres i multiplicar cadascun dels valors directament sota la matriu per el valor que té la màscara en aquella posició. Per tant: (-1*0),(-1*0),(1*0),(0*155),(1*156),... I seguidament sumar tots els valors obtinguts. Finalment fem el mateix per les matrius que representen els altres colors de la imatge i sumem tots els valors obtinguts. El valor obtingut és col·locat en una nova matriu a la posició en la qual equival a la posició on la màscara està situada sobre la matriu de la imatge.
Finalment si fem que la xarxa tracti els valors de la màscara com a pesos de la xarxa crearem un algorisme que per si sol aprendrà a buscar quin és el millor filtre a aplicar per extreure la major quantitat de característiques rellevants dins d'una imatge.
Cal destacar que utilitzar una màscara major implica un increment quasi exponencial de la complexitat de còmput.

"Pooling" o agrupació de píxels:

La capa de "Pooling" està generalment situada després d'una capa convolucional, s'utilitza per a disminuir el cost computacional de la xarxa, normalment en un factor de 2 per cada dimensió,

afectant tant al ample com al llarg de la imatge però no a la profunditat. A part d'una reducció considerable en el cost computacional, també serveix per donar una nova perspectiva similar a com si nosaltres ens allunyéssim de la imatge, a més serveix per a prevenir certs problemes que poden aparèixer al entrenar amb imatges molt similars en resolució i dimensió.
Hi ha diversos tipus de pooling el més usat i que dona millor resultats és el "max-pooling" però també podem trobar l'"Average Pooling. El "max-pooling" es basa en aplicar al funció **"Max"**[17] en els grups de matrius NxN que conformen la imatge on N és el factor de reducció.

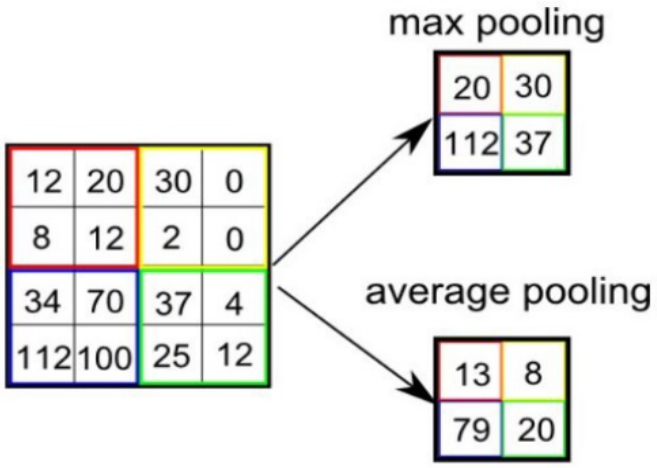


Figure 4.13: Operació de pooling en una matriu 4x4[25]

En la Figura 4.13 podem veure els dos diferents "poolings" citats anteriorment, donada una matriu de 2D amb forma 4x4, el que fem és dividir-la submatrius de dimensions 2x2 i aplicar-hi la funció seleccionada a cadascuna de manera que obtenim una matriu molt més petita.

Abstracció derivada del "pooling":
A ull nu els humans fem una abstracció dels objectes com una unitat. Però si ens apropem amb un microscopi, el nivell d'abstracció disminueix i veiem els conceptes més simples que componen l'objecte. Les xarxes neuronals, oposadament comencen amb una imatge molt detallada per tant amb molt baixa abstracció i conforme l'imatge passa per les diverses "capes" la resolució disminueix. D'aquesta manera donant un nou punt de vista més abstracte a l'algorisme sobre la imatge entrada. Vegeu la Figura 4.14

Tal com podem apreciar a la imatge anterior, l'algorisme analitza una informació diferent depenent de la profunditat. Com podem veure l'entrada de la xarxa és una imatge d'una cara humana.
En la primer capa("Hidden Layer 1") l'algorisme analitza els contorns, en la segona capa com podem veure analitza les faccions (ulls,nas,boca,..) i finalment en la tercera s'analitza la cara

[17]Funció que obté el nombre més gran dins un grup de nombres donats

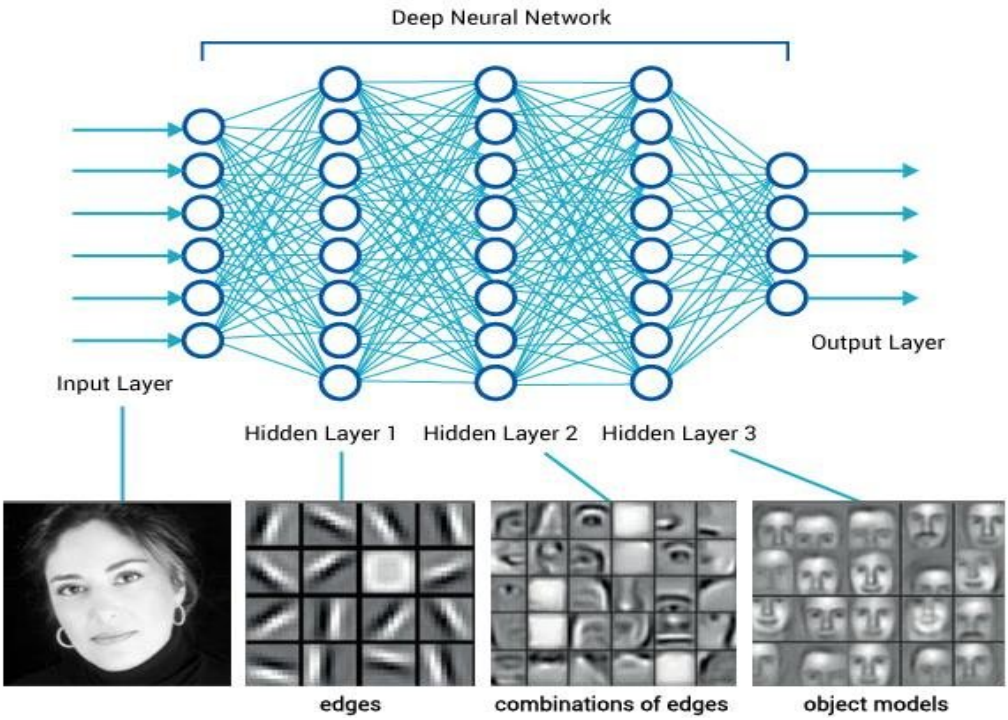


Figure 4.14: Abstracció durant l'entrenament d'una xarxa neuronal [20]

com un objecte. D'aquesta manera l'algorisme aprèn a interpretar diversos nivells d'abstracció obtenint d'aquesta manera un millor resultat.

Funció d'activació ReLu:
La funció **ReLU** o funció de rectificació, és una funció d'activació que bàsicament elimina tot resultat negatiu que pugui ser generat per la convolució. L'aplicació d'aquesta funció abans d'una convolució implica que la xarxa no tractarà valors negatius, fent que la "Propagació cap endarrera" sigui més simple d'executar i també s'ha demostrat empíricament[1] que és millor que la utilització d'altres funcions d'activació. Tot i que cal destacar, que en cap cas, un programador que crei la seva pròpia xarxa neuronal, està limitat a aquesta; n'hi ha moltes i algunes són millors en certs models. En el cas d'aquest treball s'utilitzarà la funció "ReLU". Seguidament a la Figura 4.15 podem veure quina forma té la funció "ReLU":

Capes totalment connectades
Les **capes totalment connectades** són un conjunt de capes connectades 1xN, on N és el nombre de nodes de la següent capa, situades al final de la xarxa després de l'últim "pooling" quan només tenim un vector en comptes d'una matriu. L'objectiu principal d'aquestes capes és classificar el vector resultat, també anomenat vector de característiques[18], entre les diverses classes que hi poden haver. Finalment obtinguent un vector de longitud igual al nombre de

[18]És el vector resultant de les característiques que les capes convolucionals han extret

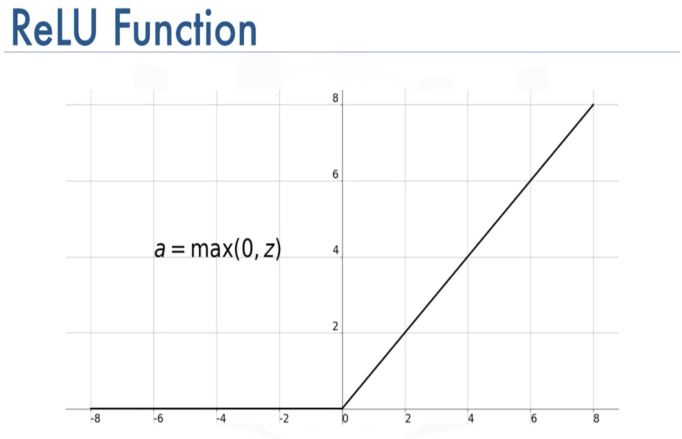


Figure 4.15: Funció ReLU

classes de sortida que té el model el qual ens donara la probabilitat de que l'entrada pertanyi a cadascuna de les diverses classes.

També hi ha un altre node a cada capa anomenat **"bias"** el qual és un altre paràmetre que ajuda a la xarxa i no està connectat a las capes prèvies. De manera fàcil seria com si tinguéssim una recta dins l'axis XY que té que separar correctament les diverses imatges, el "bias" ens ajuda que aquesta recta comenci per el punt de l'eix Y que tingui l'altura més adequada per a fer-ho. Seguidament a la Figura 4.16 en podem veure un exemple:

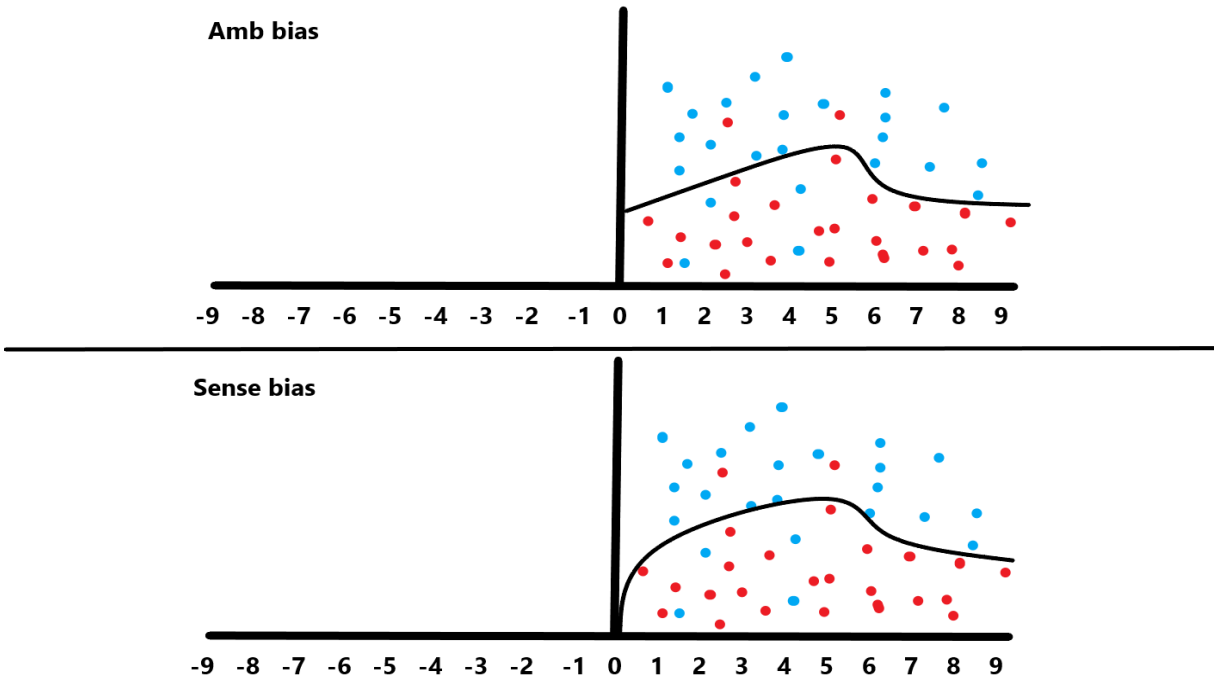


Figure 4.16: Efecte del "bias"

4.6 Xarxes neuronals

I finalment un cop incorporat el "bias" les capes totalment interconnectades seran similars amb un nombre de capes variables a la Figura 4.17 que podem veure a continuació:

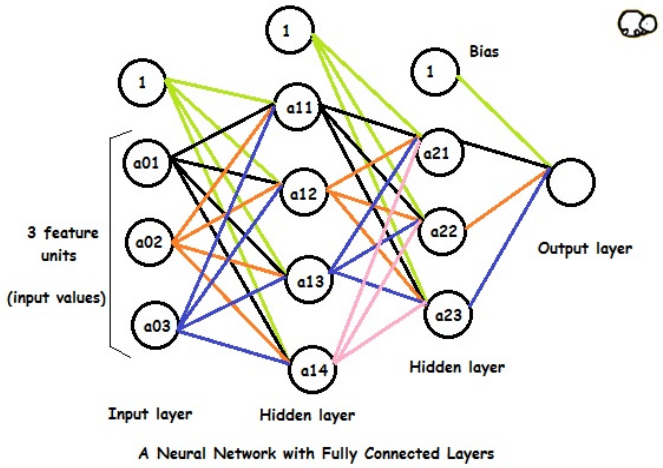


Figure 4.17: Representació Capes totalment interconnectades[22]

I per acabar en al Figura 4.18 podem veure un esquema d'una xarxa la Xarxa neuronal convolucional genèrica amb totes les seves parts.

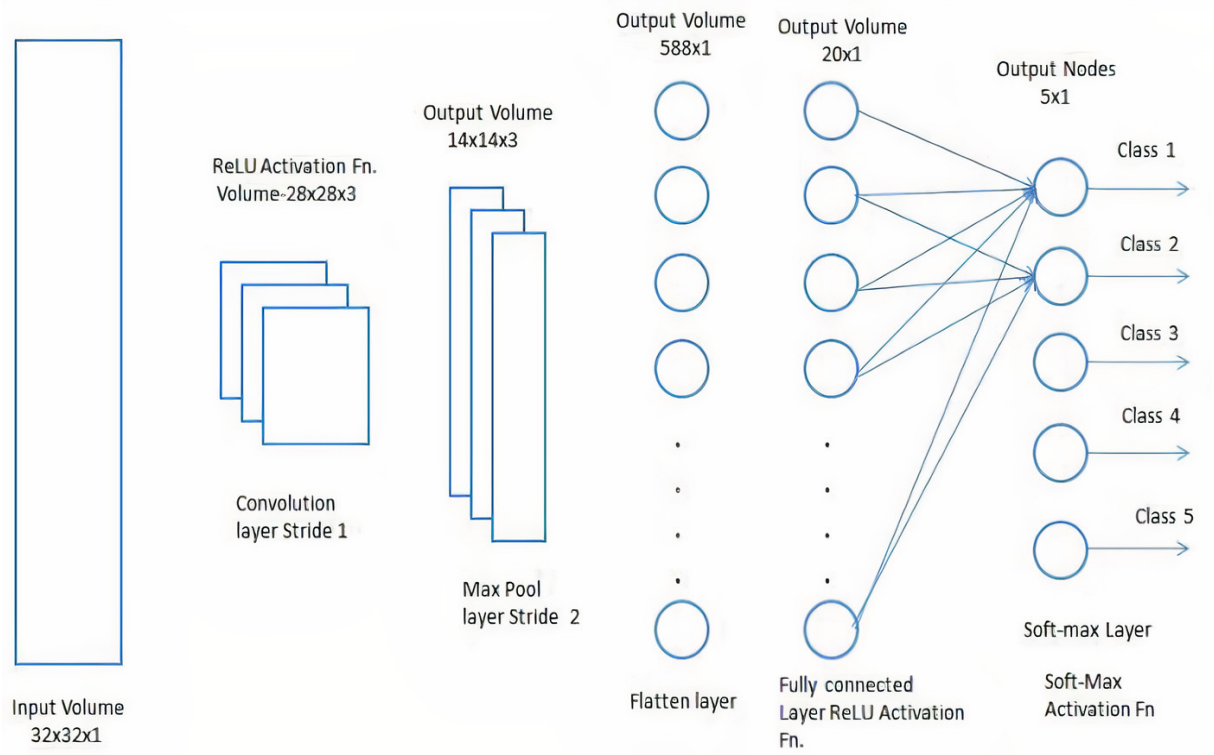


Figure 4.18: Esquema d'una Xarxa neuronal convolucional[25]

4.7 Xarxes neuronals convolucionals prèviament entrenades

En aquest projecte no es crearà una Xarxa neuronal des de zero sinó que utilitzarem xarxes neuronal ja creades i prèviament entrenades per a modelar el nostre problema. Aquestes xarxes s'anomenen "Pretrained"[19]. Per abreviar el nom d'aquest tipus de xarxes durant l'explicació usaré l'abreviatura "XCPE".
Les XCPE són xarxes testejades que tenen un rati ja trobat d'encert, s'utilitzen per a saltar-se el procés de creació de les pròpies xarxes i com que s'han provat sabem que són fiables i ens donaran molts millors resultats que les xarxes que poguéssim en molts casos fer nosaltres mateixos. També cal destacar que al estar prèviament entrenades i el fet de tenir ja uns paràmetres inicials la velocitat d'entrenament s'incrementa respecte una xarxa que no ha sigut prèviament entrenada.
D'entre elles en podem trobar varies:

- Interception-v(2,3)
- ResNet-(18,34,...)

[19]En Català prèviament entrenades

4.7 Xarxes neuronals convolucionals prèviament entrenades

- VGG-19

Cal destacar que per demostració empírica, s'ha demostrat que en l'anàlisis d'imatges amb xarxes neuronals convolucionals el nombre de paràmetres de cada capa no és tant rellevant com el nombre de capes. Per això la tendència en els últims anys ha sigut de fer XCPE cada cop més profundes i amb menys densitat de paràmetres per capa, d'aquesta manera fent-les més fàcils d'entrenar i menys pesades però amb millor rati d'encert.
Hi ha moltes XCPE diferents però en el cas d'aquest projecte només s'utilitzarà la "ResNet" amb totes les seves variants. La "ResNet" té l'avantatge de tenir moltes variants amb un nombre de capes diferents el quals ens dona la capacitat d'escollir si es necessàri i a més és una de les XCPE amb millor resultat d'encert. Per tant és l'elecció que s'ha cregut més òptima per aquest treball.

4.7.1 ResNet

La ResNet va ser una de les primeres xarxes en resoldre el problema de la desaparició del gradient durant la propagació cap endarrera("Vansihing Gradient") 4.6.2, és un problema que apareix quant el nombre de capes és tant gran que el gradient encarregat d'entrenar, es propaga tants cops que al final tendeix a 0. La solució és més simple del que es pensava, només és necessàri la creació de ponts cada N capes per a que el gradient no hagi de fer tants de salts i acabi tornant-se quasi imperceptible.

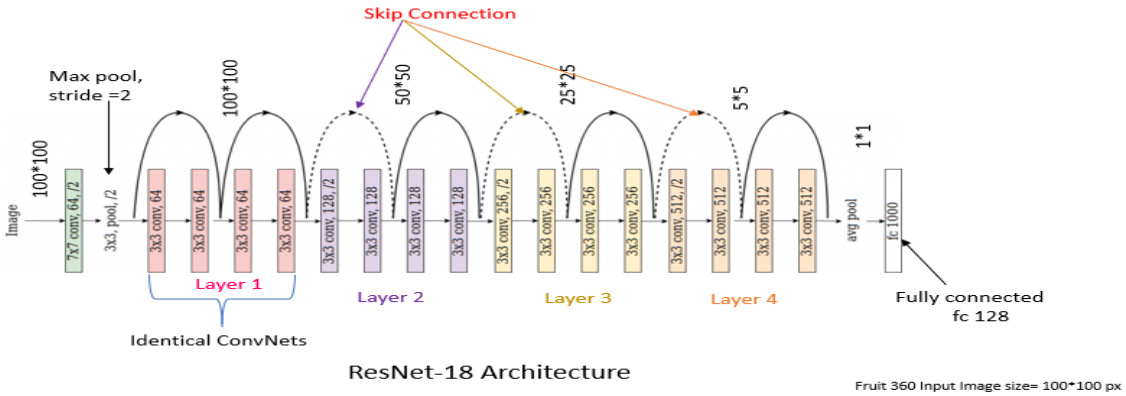


Figure 4.19: Estructura de la ResNet-18[27]

Tal com es pot veure a la Figura 4.19 cada 2 capes hi ha un pont que uneix el següent grup de 2 capes, el qual s'anomena "Skip Connection",serveix per a minimitzar el **problema de desaparició de gradients** descrit en anterioritat. També es pot apreciar que aquesta "ResNet" en particular està conformada per 18 capes. Tot i ser composta per només 18 capes l'estructura es manté entre totes les altres xarxes de la família "ResNet". També podem veure

que l'estructura està dividida en paquets de 4 capes els quals són anomenats "Layers" dins la imatge , al final de cada paquet és quan s'aplica el "pooling". Per tant les matrius dins de cada paquet mantenen les dimensions amb la qual han entrat.
Finalment es pot destacar que a l'inici tenim una capa la qual utilitza un "Pooling" de 3 i al final la capa totalment connectada.

4.8 Xarxes convolucionals completes

Les **Xarxes convolucionals completes** són el conjunt de xarxes neuronals que s'utilitzen per a recrear una nova imatge a partir d'una imatge base. Els seus principals usos són: la segmentació, la correcció de sorolls dins les imatges, la reconstrucció d'imatges,etc. Estan formades per dos parts: l'extracció de característiques(xarxa convolucional normal) i l'operació mirall.
L'**extracció de característiques** són les capes convolucionals que veiem en qualsevol xarxa neuronal convolucional i com a resultat donen un vector de característiques.
Per altre banda l'**operació mirall** utilitza les característiques obtingudes per la primera part per reconstruir una nova imatge segons l'entrenament rebut.
Seguidament a la Figura 4.20 podem veure com està estructurada una Xarxa convolucional completa:

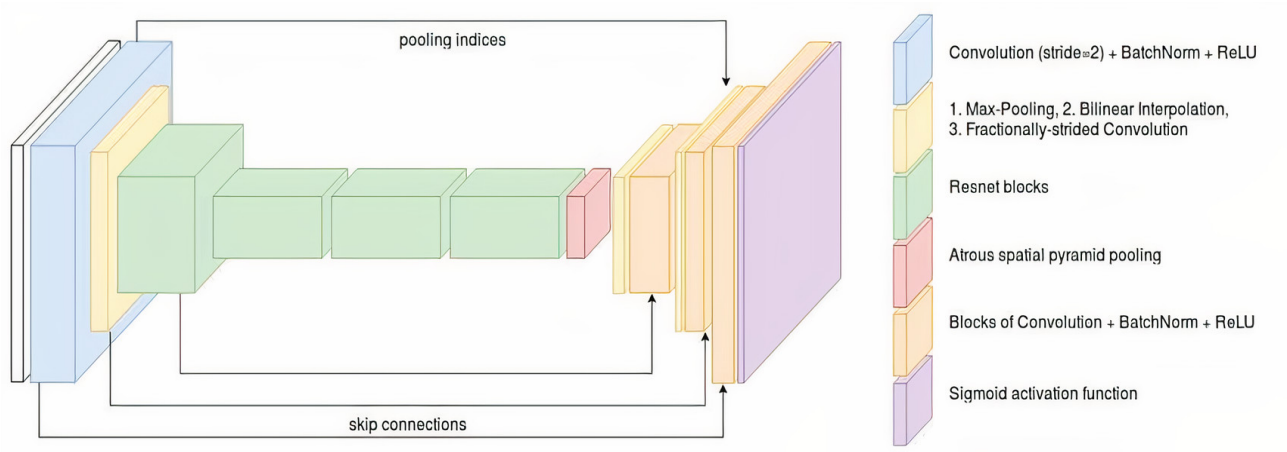


Figure 4.20: Estructura d'una xarxa convolucional completa basada en blocs ResNet[26]

Tal com podem veure en la Figura 4.20, la xarxa té una forma irregular on els extrems són de major mida que el centre. Aquest canvi de mida està relacionat en amb les dos parts que conformen la xarxa.
Durant la primera etapa, l'etapa que redueix la mida de la imatge, s'apliquen les convolucions vistes amb anterioritat que són les encarregades d'extreure les característiques de la imatge entrada. Al final de cada convolució, s'aplica un a funció "max pooling", explicades en la Secció 4.6.4, per disminuir el nombre de píxels.
Seguidament hi ha una secció, la secció situada en el centre, que està formada per un seguit de

convolucions sense "max pooling". Que té la única funció d'extreure més característiques de la imatge.
Finalment tenim la última part conformada per **Interpolació Bilineal**, **Convolucions transposades**, Convolucions, **Atrous Spatial Pyramid Pooling (ASPP)** i una funció d'activació de tipus Sigmoide. Aquesta part té la funció de recrear una nova imatge a partir de la imatge d'entrada.
Per a poder tornar a generar una imatge a partir de les característiques extretes durant la primera i segona part del model s'utilitza **l'"upsampling"**. L'"upsampling" no és més que una paraula que vol dir "incrementar la mida". Per tant per a poder dur a terme aquesta tasca s'utilitzen diversos mètodes. Els quals seguidament es presentaran:

4.8.1 Interpolació Bilineal

L'**Interpolació Bilineal** és una funció que incrementa la la mida d'una matriu interpolant els valors de manera que només els extrems retenen els seus valors. Els píxels restants es generen a partir de la ponderació de la distància entre els píxels del extrems i el píxel que es vol generar. Vegeu l'exemple 2 següent:

Tenim com a matriu inicial la matriu i li fem un "padding" 2x2 :

$$\begin{bmatrix} 4 & 10 \\ 6 & 8 \end{bmatrix} \Longrightarrow \begin{matrix} (x_0, y_0) \\ \\ \\ (x_0, y_1) \end{matrix} \begin{bmatrix} 4 & 0 & 0 & 10 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 6 & 0 & 0 & 8 \end{bmatrix} \begin{matrix} (x_1, y_0) \\ \\ \\ (x_1, y_1) \end{matrix}$$

Seguidament per cada valor creat en la matriu apliquem les fórmules següents:
Com exemple s'aplicarà en la posició (P_1, P_2)=(2,2) de la matriu

$$I(X) = \frac{x_1 - P_1}{x_0 - x_1} * (x_0, y_0) + \frac{P_1 - x_0}{x_0 - x_1} * (x_1, y_0) = 6$$

$$I(4 - 10) = \frac{4-2}{4-1} * 4 + \frac{2-1}{4-1} * 10 = 6$$

$$I(6 - 8) = \frac{4-2}{4-1} * 6 + \frac{2-1}{4-1} * 8 = 6.666$$

$$I(X) = \frac{y_1 - P_2}{y_0 - y_1} * (x_0, y_1) + \frac{P_2 - y_0}{(} x_1, y_1) * 10 = 6$$

$$I(Y) = \frac{4-2}{4-1} * 6 + \frac{2-1}{4-1} * 6.6666 = 6.222$$

Obtenint el següent resultat:

$$\begin{bmatrix} 4 & 10 \\ 6 & 8 \end{bmatrix} \Longrightarrow \begin{bmatrix} 4 & 6 & 8 & 10 \\ 4.666 & 6.222 & 7.777 & 9.33 \\ 5.333 & 6.444 & 7.555 & 8.66 \\ 6 & 6.666 & 7.333 & 8 \end{bmatrix}$$

Table 2: Aplicació de la Interpolació Bilineal

4.8.2 Convolucions transposades

Les **Convolucions transposades**, són convolucions aplicades sobre una matriu que se l'hi ha aplicat un "padding"[20] i un "stride"[21].
Seguidament el que es fa és fer una convolució sobre la matriu que se li ha aplicat el "padding" i l'"stride" amb una màscara de la mida desitjada. Obtenint així una nova matriu normalment amb una mida superior a la inicial.
Vegeu la Figura 4.21:

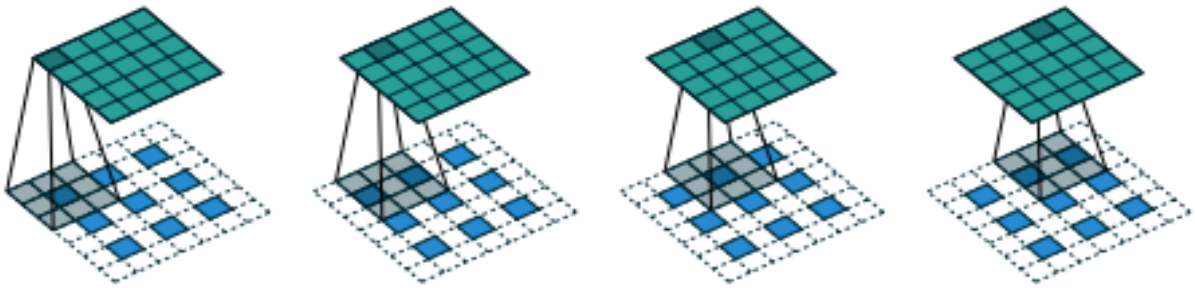


Figure 4.21: Convolució transposada amb padding=(1,1), stride=2 i màscara 3x3.[6]

4.8.3 Atrous Spatial Pyramid Pooling (ASPP)

L'**Atrous Spatial Pyramid Pooling (ASPP)**, és basa en l'utilització de màscares normalment de dimensions 3x3, que apliquen una convolució sobre la matriu. Seguidament els resultats obtinguts es separen verticalment i horitzontalment per r-1 valors 0 aquest procés es repeteix amb diversos valors de r. I finalment es concatenen tots els valors. Obtenint una forma similar a una piràmide.
Vegeu un exemple en la Figura 4.22:

[20] Inclusió de valors 0 al voltant de la matriu, està representat com una tupla de dos valors (x,y).
[21] Separació generada entre els valors de la matriu. Per exemple si el "padding" és 2 hi haurà una separació d'una casella amb valor 0 entre els components de la matriu

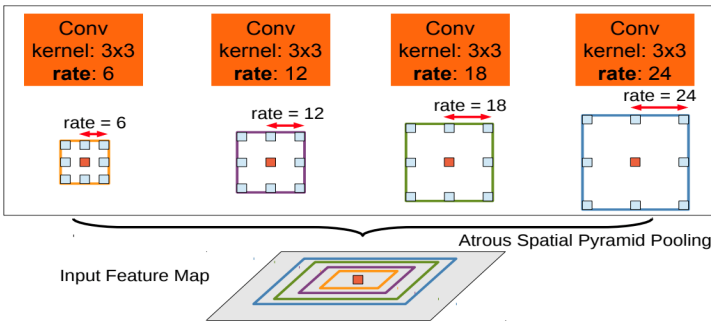


Figure 4.22: Atrous Spatial Pyramid Pooling.[17]

4.8.4 Connexions de salt (Skip connections)

Finalment tenim les Connexions de salt que es basen en copiar les matriu de característiques del costat esquerra, ajustant-les de manera que coincideixin en mida de les matrius del mateix nivell del costat dret. I finalment enganxar-les a la pila de matrius del mateix nivell que tenim en el costat dret. Aquesta funció fa un treball similar als salts de la xarxa ResNet, ajudant a traspassar el gradient cap endarrera més fàcilment. I ajudant a la xarxa a recordar la imatge d'entrada.

4.9 CycleGAN o Xarxes adversàries de cicle-consistent

La **xarxa GAN** és una xarxa neuronal amb l'objectiu de fer una traducció imatge a imatge. O sigui donada una entrada de "Tipus A" transforma-la en una imatge de Tipus B. Per exemple: Transformar una fotos de gats en una foto de gossos, donada una imatge radiològica del tòrax d'un pacient de CoVid-19 generar una imatge que mostri l'evolució de la malaltia en el tòrax,etc.

4.9.1 Estructura d'una CycleGAN

La **CycleGAN** és una unió de xarxes neuronals amb diferents propòsits, normalment i en la gran majoria dels casos estan formades per 4 xarxes, 2 convolucionals i 2 Xarxes convolucionals completes. Seguidament podem veure en la Figura 4.23 un exemple de l'arquitectura d'una xarxa CycleGAN:

En la figura anterior podem distingir que tenim dos parts, la superior que transforma els cavalls en zebres i el torna a transformar en cavall i la inferior que transforma zebres en cavalls i el torna a transformar en zebra.
També podem veure que aquesta CycleGAN està composta per 4 subxarxes les xarxes GAN A i B(Requadres Blaus) que fan el mateix treball però en sentit oposat i finalment dos xarxes de convolució binaries(només diuen si o no), una que diu si la imatge és o no del grup de l'etiqueta d'entrada i una altre que diu si la imatge és de l'etiqueta de sortida.

4.9 CycleGAN o Xarxes adversàries de cicle-consistent

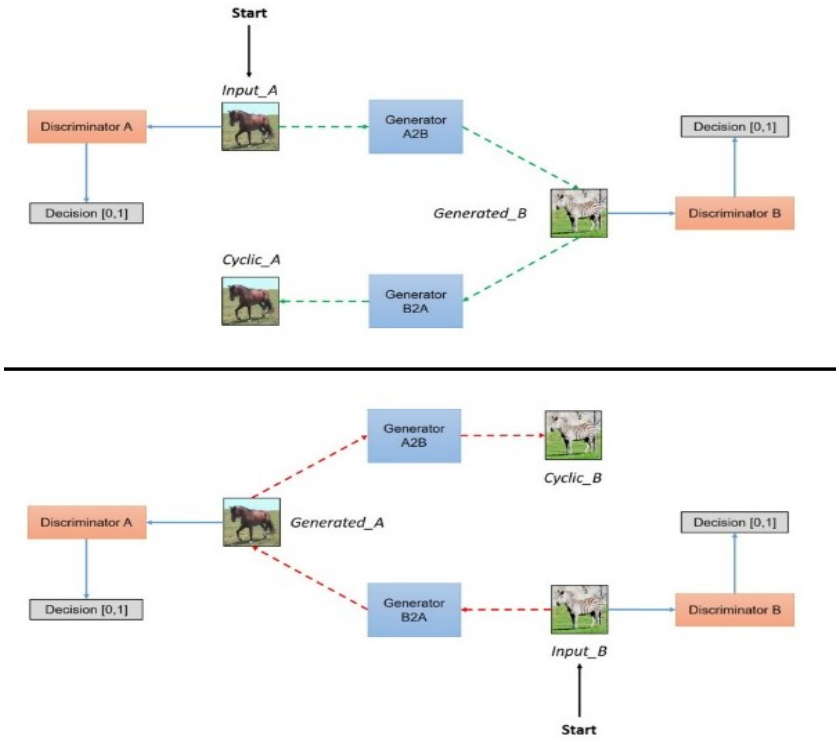


Figure 4.23: Estructura d'una xarxa CycleGAN[29]

4.9.2 Ús de la CycleGAN

Imaginem que tenim una successió d'imatges en el temps, ara imaginem que el temps és una transformació. Si la CycleGAN és capaç de fer una transformació entre dos tipus diferents d'imatges, també hauria de ser capaç de trobar els patrons dels pulmons que tendeixen a empitjorar i els que tendeixen a posar-se sans i aplicar una transformació que pugi crear una imatge predictiva.
El problema principal és la necessitat de tenir un seguit d'imatges ordenades en el temps amb les quals es pot entrenar el model. D'aquesta manera s'ha buscat aquesta correlació entre les imatges pertanyent en el conjunt de dades BIMCV**??**.

4.10 Avaluació de resultats

En aquest treball per a validar els resultats obtinguts, s'utilitzaran les **matrius de confusió**. És la forma més simple i visual de veure i entendre els resultats obtinguts. Si observem la Taula 3 en podem veure un exemple:

		Real	
		Positiu	Negatiu
Trobat	Positiu	VP	FN
	Negatiu	FP	VN

Table 3: Exemple de matriu de confusió

Les columnes de la Taula anterior representen els valors reals que tenen les imatges mentrestant les línies representen els valors trobats per la xarxa.
Tenim 4 valors dins la matriu: VP(Verdaders Positius),FP(Falsos Positius),FN(Falsos Negatius) i VN(Verdaders Negatius). **Els Verdaders positius i negatius** són la quantitat d'imatges que la xarxa neuronal ha encertat. Mentrestant els **falsos positius** són les imatges que la xarxa hauria d'haver donat negatiu i ha donat positiu i els **falsos negatius** són les imatges que la xarxa hauria d'haver donat positiu i ha donat negatiu.

D'aquest matriu en podem extreure dos valors que ens poden ser molt útils que són **l'error i la precisió**.
La precisió és la divisió entre les imatges detectades correctament i el total d'imatges analitzades.

$$Precisio = \frac{VP + VN}{VP + FP + FN + VN} \tag{1}$$

I seguidament **l'error** és simplement la "1-Precisió".

$$Error = 1 - \frac{VP + VN}{VP + FP + FN + VN} \tag{2}$$

També cal destacar que hi ha altres mètriques usades amb les matrius de confusió. Aquestes són:
La **Sensibilitat**, indica la capacitat del nostre model en detectar els casos positius dels negatius, utilitza la fórmula:

$$Sensibilitat = \frac{VP}{VP + FN} \tag{3}$$

I la **especificitat**, indica la capacitat del nostre model de detectar els casos negatius dels positius, utilitza la fórmula:

$$Especificitat = \frac{VN}{VN + FP} \tag{4}$$

4.11 Problemes derivats de l'entrenament de xarxes neuronals

Quan s'entrena una xarxa neuronal amb pesos, poden aparèixer varis problemes derivats d'un entrenament que no s'ha dut a terme de manera correcta.

4.11.1 Overfit

El primer i més comú és **l'Overfitting**, un problema que indica que la xarxa està començant a individualitzar les dades. O sigui donat un conjunt de dades d'entrenament, la xarxa aprèn a separar-les individualment i no com a grup.
Per a poder detectar aquest problema s'utilitzen dos conjunts de dades, les dades d'entrenament i les de validació. I quan es veu que l'error de la xarxa divergeix entre la validació i l'entrenament sabem que **ha començat "l'Overfit"**.

Hi ha diverses raons que causen l'aparició d'aquest problema. La primera és el **sobre entrenament del model**, de manera que de tant entrenar-lo amb un sol conjunt comença a individualitzar les dades, l'altre raó és que les dades que utilitzem en l'entrenament **no són suficientment variades** i aprèn només a diferenciar aquell tipus de dades.

Per **solucionar aquest problema** s'utilitzen diverses tècniques, una és el "Data Augmentation"[22] que ajuda a aprendre nous paràmetres a la xarxa i endarrereix "l'overfit", l'altre és simplement

[22]Són un conjunt de tècniques utilitzades per a crear copies modificades de dades ja existens o la creació de noves dades a partir de les dades existents

parar l'entrenament quant la pèrdua que hi ha durant l'entrenament i la validació és molt diferent.

4.11.2 Underfit

També tenim **l'undefit**, un problema que indica que la xarxa no té suficient capacitat per classificar correctament els inputs que rep.
Es pot detectar aquest problema quan veiem que la classificació dona resultats molt baixos i no millora en el temps, normalment es deu al fet de tenir molts de casos atípics o simplement que el problema és massa complex per a poder detectar-lo correctament.
No hi ha solucions contundents a aquest problema, normalment el que es fa és acotar el problema a un problema més simple per la xarxa o intentar millorar la capacitat de la xarxa. Però en ambdós casos tindrem efectes no desitjats, per exemple en el cas de millorar la xarxa podem fer que el model sigui molt pesat i computacionalment costós a més no és assegurat que millorarà.

5 Estudis i decisions

En la **Secció 2** s'han descrit quins processos s'han dut a terme per seleccionar i prendre decisions sobre l'àmbit de treball. En aquest apartat explicarem a quines conclusions hem arribat, quin programari hem usat i quins requisits hi ha.

5.1 Requisits per a entrenar xarxes neuronals

Al ser un treball de recerca sense creació d'una aplicació, utilitzaré aquest apartat com a explicació dels requisits que s'han de tenir en compte si es vol realitzar un treball utilitzant les Xarxes neuronals.
En aquest apartat mencionarem quins són els **requisits Necessàris**[23] **i Recomanables**[24] que són necessàris per a dur a terme aquest treball.
S'ha de destacar abans que res que tots aquest requisits són d'entrenament, l'execució d'un model no és tant demandant, fins i tot els telèfons mòbils podrien executar-los sense gran problema.

5.1.1 Requisits "Hard"

Les Xarxes Neuronals, són algorismes que requereixen una gran quantitat tant de còmputs matricials com de memòria per a poder-se entrenar. Per tant és quasi un requisit indispensable, tenir en possessió una GPU per a poder entrenar-les.
Seguidament, no és un requisit com a tal del sistema però si indispensable per a entrenar les xarxes neuronals, és tenir una gran base de dades amb una gran quantitat d'exemples per a poder entrenar de manera correcta la xarxa. Hi ha solucions que podríem aplicar en cas de tenir una base de dades molt reduïda amb els mètodes coneguts com a "Data Augmentation"[25] però sempre és necessàri un conjunt important de dades originals per a dur a terme la tasca d'entrenar una xarxa neuronal.

5.1.2 Requisits "Soft"

Utilitzar un servidor amb bastants "nuclis" de processament per a poder dur a terme l'entrenament. L'entrenament treu molta capacitat de còmput sobretot quant es carregen les imatges a la "RAM" de la gràfica (GDDR[26]).
Utilitzar una gràfica amb una memòria considerable recomanable 4 GB com a mínim i una capacitat de còmput decent. La capacitat de còmput no és tant important ja que només varia el

[23] Són els requisits indispensables per a poder dur a terme el treball
[24] Són els requisits recomanables però no totalment obligatoris
[25] Conjunt de algorismes que serveixen per incrementar el nombre de dades que tenim a partir de la modificació de les que tenim en possesió
[26] És la memória dedicada de la gràfica

temps d'entrenament però la memòria es bastant rellevant perquè si és massa petita no podrem entrenar les xarxes.
Utilitzar una GPU de la marca Nvidia. Tot i l'existència d'altres marques de GPUs tant els Drivers com la tecnologia de Tensors que té Nvidia estan a anys de diferència de les altres opcions que hi pugin haver. Per tant és molt recomanable l'ús de GPUs d'Nvidia.

5.2 Maquinari

En aquest treball només requereix d'un ordinador equipat amb una gràfica per a poder executar i provar tots els algorismes que necessitem. Degut al factor de no tenir en propietat un ordinador amb gràfica discreta[27] i que les dades són molt pesades i requereixen una gran quantitat d'espai a disc, els tutors em van oferir la oportunitat de poder usar un servidor dins el seu departament el qual podem veure les seves especificacions a la Taula 4:

Sistema operatiu	Ubuntu 16.04.6
CPU	Intel Xeon E5-2630 v4
GPU	Nvidia GeForce RTX 2080 Ti
GPU GDDR6	11GB
RAM	320 GB

Table 4: Informació sobre el sistema

5.3 Controladors específics

Els **Controladors o drivers** són part de software que actuen com el primer pont entre la maquinaria i el software, i donen la capacitat al software de poden cridar funcions de la maquinària.
En el cas d'aquest projecte només és necessàri l'instal·lació d'uns drivers en específic que són els drivers dedicats de les GPUs d'Nvidia.

5.4 Llenguatge de programació

En aquest punt s'havia de decidir quin llenguatge de programació era el més adequat per a utilitzar. No va ser una tasca molt difícil, el llenguatge amb més llibreries i millor suport per a dur a terme aquest projecte era el **Python** amb diferència.
Tot i que hi ha molts llenguatges que donen suport a la creació de xarxes neuronals, molt

[27] Tarjeta gràfica no pertanyent al processador que té memòria pròpia

pocs tenen un suport i tant bones llibreries com té el Python. També hi ha programes com el **MatLab** que tenen una llibreria integrada amb capacitat d'entrenar xarxes neuronals de manera fàcil, però no tens la llibertat de retocar les xarxes que pots trobar en les llibreries de Python.

Vam elegir la versió **Python 3.6.10**, la raó principal per a no elegir la versió més nova el Python 3.9.x és degut a les incompatibilitats que es van trobar al principi. En un primer moment es va descarregar la versió 3.9.1, però van aparèixer certs errors el qual ens van portar de a la 3.6.10 la qual és la que acabarem fent servir.

5.5 Llibreries

Existeixen moltes llibreries que optimitzen, faciliten o són necessàries durant l'implementació de les xarxes neuronals. Però moltes no cal instal·lar-les ja que són "in-built" o sigui dins altres llibreries que s'utilitzaran. Per tant les mencionarem durant l'introducció de la llibreria que les conté . Dit això seguidament s'introduiran les llibreries utilitzades:

5.5.1 PyTorch

Les llibreries **PyTorch** van ser implementades per l'entrenament de xarxes neuronals i processament d'imatges. Inicialment era una llibreria implementada per al llenguatge C++, però amb la tendència dels últims anys cap al "Python" es va transcriure el codi en llenguatge Python. Actualment PyTorch ja és molt més gran que el seu predecessor escrit en C++.

Per que el PyTorch?
Hi ha tres grans llibreries dins del món de les xarxes neuronals: **Keras, TensorFlow, PyTorch** on cadascuna té els seus avantatges i inconvenients. Si ens fixem en la Figura 5.24 podrem veurem quins són aquest avantatges i inconvenients:

5.5 Llibreries

	Keras	TensorFlow	PyTorch
Level of API	high-level API[1]	Both high & low level APIs	Lower-level API[2]
Speed	Slow	High	High
Architecture	Simple, more readable and concise	Not very easy to use	Complex[3]
Debugging	No need to debug	Difficult to debugging	Good debugging capabilities
Dataset Compatibility	Slow & Small	Fast speed & large	Fast speed & large datasets
Popularity Rank	1	2	3
Uniqueness	Multiple back-end support	Object Detection Functionality	Flexibility & Short Training Duration
Created By	Not a library on its own	Created by Google	Created by Facebook[4]
Ease of use	User-friendly	Incomprehensive API	Integrated with Python language
Computational graphs used	Static graphs	Static graphs	Dynamic computation graphs[5]

Figure 5.24: Avantatges i inconvenients de les diverses llibreries de xarxes neuronals [15]

L'informació està en anglès però amb poques paraules podríem dir que **Keras** és una bona llibreria per crear i testejar de manera que és la més útil per desenvolupar projectes ràpid i saber com funcionen, però perd en flexibilitat i velocitat d'entrenament.
Per altre banda el **TensorFlow** destaca pel fet de ser una llibreria amb una corba d'aprenentatge continu, és molt fàcil d'utilitzar quan volem fer algorismes simples i quan volem retocar-los o optimitzar-los només caldrà aprendre una mica més sobre el que ja hem aprés.
Finalment el **PyTorch** és el més complex dels tres però també el més eficient i més flexible i és més fàcil de "debuggar"[28] i trobar els problemes que podem haver fet durant al programació.
Com podem veure el millor és el **PyTorch** però també és el més difícil d'aprendre per això s'ha de trobar una manera simple de poder aprendre a com usar-lo.

Llibreries "in-built" PyTorch

El PyTorch durant la seva instal·lació automàticament ja instal·la totes les llibreries necessàries per a poder executar totes les funcions que té incorporades. Per tant hi ha dos llibreries que voldria destacar que en condicions normals hauríem d'instal·lar.

La primera és **CUDA** són un conjunt de llibreries d'NVIDIA que serveixen per optimitzar i facilitar als usuaris l'utilització dels diferents components de la gràfica. En aquest cas podem destacar la sub-llibreria dins de CUDA anomenada **cuDNN** que optimitza l'execució de xarxes neuronals activant els Tensor Cores [29].

[28] Procés en el qual s'intenten trobar els error dins de l'algorisme desenvolupat.
[29] Conjunt de nuclis dedicats en l'entrenament de AI els quals faciliten el Propagament cap endarrera

La segona és **ATen** bàsicament és una llibreria que fa operacions amb Tensors [30] i s'obté d'una llibreria ja creada en C++ que s'exporta a Python utilitzant una API de Python.

5.5.2 FastAI

En l'apartat anterior s'ha introduït el PyTorch, una llibreria dedicada en l'entrenament de Xarxes neuronals, però tal com s'ha dit té un gran problema i és la corba d'aprenentatge. Per a poder solucionar aquest problema s'utilitzarà **FastAI** [13], un conjunt de llibreries escrites en Python que tenen com a base el PyTorch i faciliten molt l'ús i l'aprenentatge d'aquest.

El gran avantatge del FastAI resideix en les guies que tenim a la seva web que expliquen pas a pas com és el funcionament d'aquesta llibreria. Però el més important és que al mateix moment que s'aprèn a utilitzar el FastAI mitjançant aquestes guies, també s'aprèn a com interactuar amb el PyTorch. I com més coses volem canviar a baix nivell més aprenem a interactuar amb el PyTorch i un cop finalitzades les guies que podem trobar a la web de FastAI [13] haurem aprés tant l'utilització de les seves llibreries com part de les de PyTorch.

El FastAI s'utilitzarà com a mètode d'aprenentatge, però durant l'implementació **s'utilitzarà PyTorch pur**.

5.6 Jupyter Notebook

El **Jupyter Notebook** és un entorn informàtic que facilita l'estructuració del codi emprat utilitzant diverses tècniques com la simbiosis entre Text i codi,l'utilització del cel·les, la capacitat de mostra gràfics i la millora de la visualització dels textos.
Utilitza el format de dades JSON[31] per a guardar els fitxers, tot i que l'extensió en aquest cas és ".ipynb".
S'ha escollit l'utilització d'aquest entorn per dos raons principals:

- **La facilitat d'incorporar Entorns virtuals**[32] el qual ens ajuda a tenir el nostre grup de llibreries separades del servidor i així seleccionar quines versions són les més òptimes per el nostre treball sense haver de modificar les llibreries que utilitzen els altres usuaris dins del servidor.
- **La modularitat i l'aïllament que donen les cel·les**, ens ofereix la capacitat de tornar a córrer fragments de codi sense haver de tornar a córrer tot el programa amb al seva

[30] Són contenidors que serveixen per guardar dades en diferents dimensions. ex:(si tenim 1 dimensió tindrem un vector, si tenim 2 dimensions tindrem una matriu,...)
[31] És un format de dades de notació literal que facilita molt la creació de programes de lectura per a aquest format.
[32] Són entorns que tenen les seves pròpies llibreries i aplicacions, aïllades de les que es tenen instal·lades en el Sistema Operatiu.

utilitats estalviant molt de temps. A més si tenim moltes xarxes per entrenar no hem de crear un programa per cadascuna només caldrà que executem les cel·les que necessitem.

5.7 Pandas

Pandas és una biblioteca de software escrita com a extensió del NumPy que serveix per modificar i gestionar conjunts de dades.
En aquest projecte es treballarà sobre grans porcions de dades per tant s'ha cregut òptim que utilitzar una llibreria per ajudar a llegir, modificar i gestionar aquestes dades milloraria l'eficiència de treball.

5.8 NumPy

NumPy és una llibreria per el llenguatge de programació de Python que dona suport a la creació de tensors de diverses dimensions i moltes de les operacions derivats d'aquest contenidors.
És molt útil en el tractament d'imatges i vectors per tant aportarà facilitat a l'hora de modificar o treballar sobre les pròpies imatges que utilitzarem en aquest projecte.

5.9 Matplotlib

El **Matplotlib** és una llibreria per la generació de gràfics de dades dins del programa de programació de Python i depèn de NumPy.
Ofereix la capacitat de mostrar conjunt de dades de forma molt més visual, el qual ajudarà a poder mostrar de manera mes fàcil les dades que utilitzarem durant el treball.

5.10 Pytorch buscador de "Learning Rate"

Durant l'introducció dels conceptes previs s'ha introduït el "Learning Rate" **LRate**, s'ha parlat que aquest factor pot condicionar molt l'aprenentatge de la xarxa i que si no escollim un "Learning Rate" correcte l'aprenentatge ens pot divergir. Per aquesta raó en aquest treball utilitzarem un **buscador de LR**[4].
Aquest buscador de LR ens ofereix la capacitat de no haver-n'us de preocupar per quin "Learning Rate" hem de buscar i automàticament ens mostra quin és "Learning Rate" més óptim que podem escollir. Seguidament a la Figura 5.25 en podem veure un exemple:

5.10 Pytorch buscador de "Learning Rate"

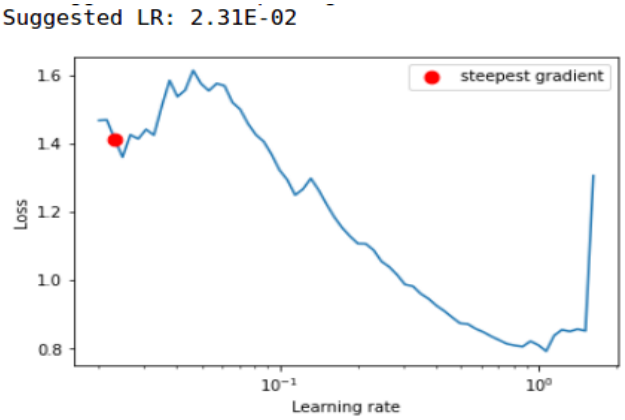


Figure 5.25: Buscador del "Learning Rate" més òptim [4]

El que fa aquesta llibreria és fer varies iteracions d'aprenentatge cadascuna amb un "Learning Rate" diferent, al final de cada iteració no guarda els resultats de manera que la xarxa no aprèn. Amb tots els valors que ha trobat crea una gràfica, dins de la gràfica busca el punt amb el pendent més gran i que sigui decreixent. Finalment retorna aquell valor com el "Learning Rate" més adient.

6 Disseny del sistema, implementació i proves

En aquest apartat s'explicarà com s'ha implementat el projecte i els problemes que hem trobat durant al procés. També s'intentarà explicar les decisions que em pres i per què les hem pres.

Aquest apartat s'estructurarà en 6 majors parts, les xarxes neuronals emprades, el disseny del sistema, el tractament de dades, la xarxa de detecció de CoVid-19, implementacions extres i finalment la xarxa CycleGAN.

6.1 Xarxes neuronals

En aquest projecte s'usaran 3 tipus diferents de xarxes. Totes 3 han sigut presentades amb anterioritat.
En aquest apartat es farà un breu resum de quines xarxes s'utilitzaran i per a què s'utilitzaran.

- **Xarxa de classificació binària**: Intenta solucionar el problema de decisió entre positiu o negatiu de CoVid-19. Són xarxes convolucionals que només tenen un vector de sortida de només dos valors Reals. Un dels valors és la probabilitat que es detecti CoVid-19 en la imatge i l'altre que no s'hi detecti. Finalment s'agafa el més gran. En la Figura 6.26 es mostra un esquema de l'entrenament d'aquesta xarxa:

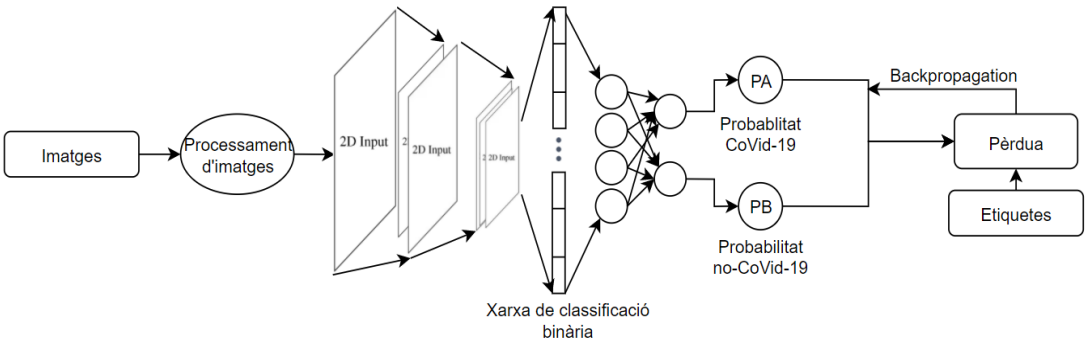


Figure 6.26: Estructura de l'aprenentatge d'una xarxa convolucional de classificació Binària

- **Xarxa de classificació multiclasse**: Intenta solucionar el problema de predir cada imatge entrant quines etiquetes radiològiques té. Igual que la anterior és una xarxa convolucional però el vector de sortida té una longitud superior a 2. L'intenció d'aquesta xarxa és si pot predir amb un encert raonable les etiquetes radiològiques, es podria a posteriori utilitzar per millorar els resultats de la primera xarxa.En la Figura 6.27 es

mostra un esquema de l'entrenament d'aquesta xarxa:

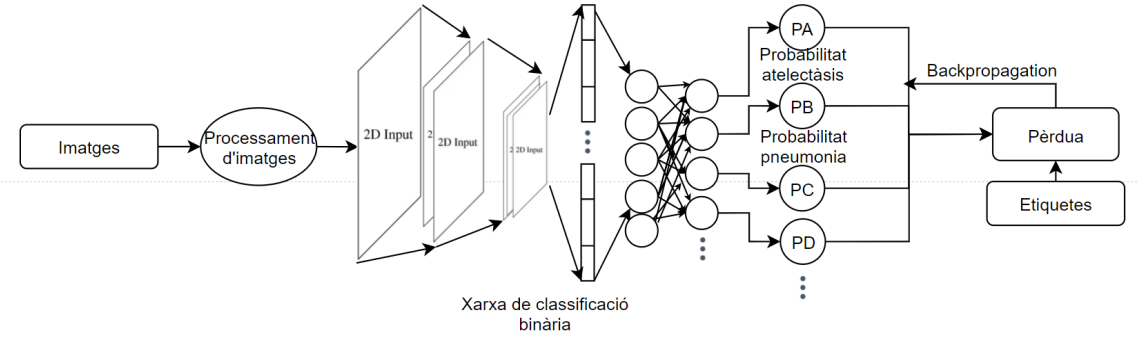


Figure 6.27: Estructura de l'aprenentatge d'una xarxa convolucional de classificació multiclasse

- **CycleGAN**: S'utilitzarà per a generar imatges predictives a partir d'una imatge d'entrada. En aquest cas en concret, s'utilitzarà per donada una imatge radiogràfica del tòrax actual d'un pacient, obtenir una imatge predictiva de com evolucionarà la situació pulmonar del pacient. Seguidament a la Figura 6.28 podem veure un esquema simple de l'estructura de l'entrenament d'una CycleGAN:

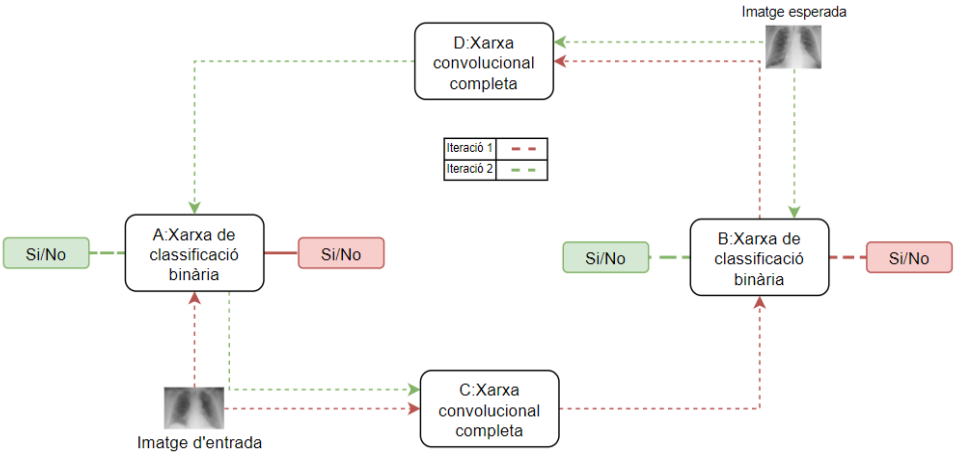


Figure 6.28: Estructura de l'aprenentatge d'una CycleGAN

6.2 Anàlisi i disseny del sistema

L'objectiu d'aquest Projecte no és la creació d'una aplicació com a tal sinó trobar si es pot crear un algorisme que pugui decidir si un pacient sofreix CoVid-19 i com evolucionarà. Al ser un treball de recerca sense un ús directament comercial no tindrem cap aplicació amb interacció

amb l'usuari.

Per tant en aquest apartat es mostrarà un petit esquemàtic de com s'estructura l'implementació usada, vegeu la Figura 6.29 següent:

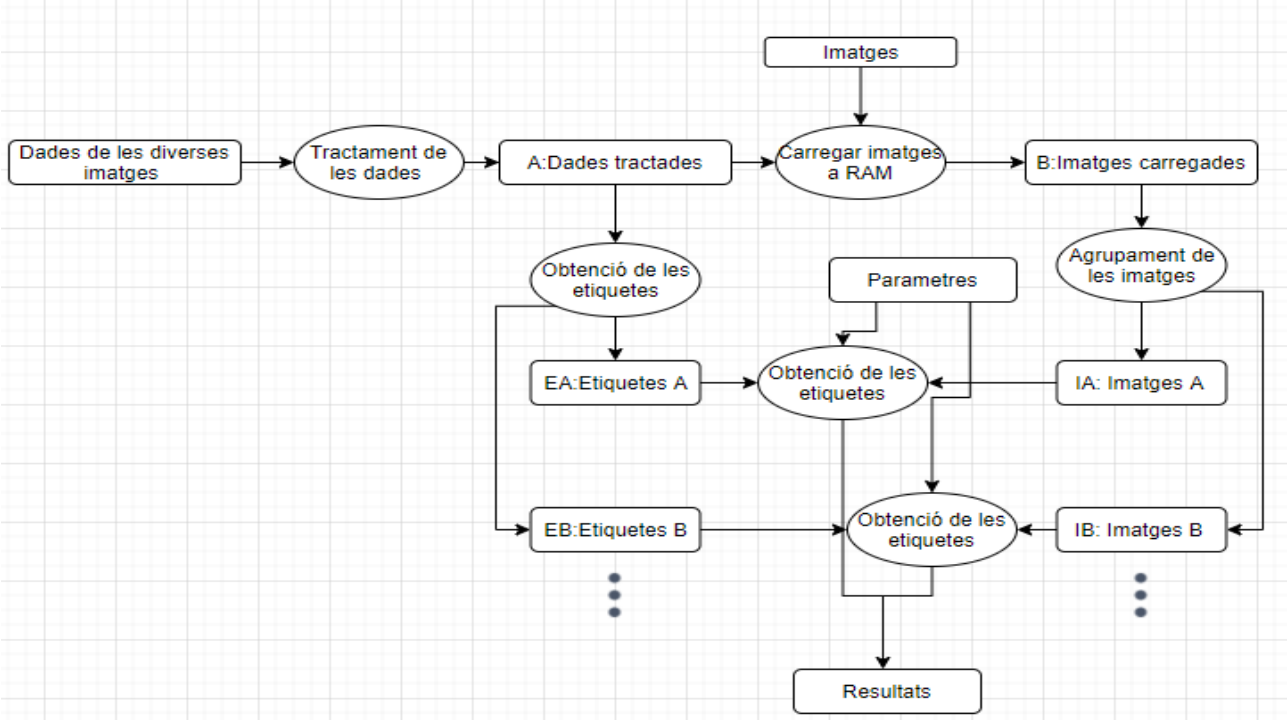


Figure 6.29: Esquemàtic de l'entrenament d'una xarxa neuronal

6.3 Dades utilitzades

Les dades emprades dins d'aquest projecte han sigut aportades per els tutors d'aquest treball Dr. Robert Martí Marly i Dr. Xavier Lladó Bardera. Les dades són un conjunt d'imatges radiogràfiques del tòrax de diferents pacients que han patit o pateixen diferents patologies pulmonars, són en nivells de gris i estan guardades en la extensió ".png".

Aquestes imatges pertanyen al conjunt de dades mèdiques del BIMCV(Banco de Imagenes Mèdicas de la Comunidad de Valencia)[3] que conté en gran quantitat imatges Radiogràfiques(RX) i en menor quantitat Tomografies Computeritzades (TAC), seguidament també ens ofereix varis documents en format ".tsv" que contenen totes les dades i etiquetes derivades d'aquestes imatges. El conjunt de dades va ser creat l'1 de Juny de 2020 però es va actualitzant en el temps segons les dades que es van obtenint i ofereix un nombre total aproximat de 45000 casos tot. Però el cas d'aquest projecte s'usarà **un subconjunt d'aproximadament 19000 dades** que pertany al conjunt de dades BIMCV. Dins d'aquest subconjunt també seran descartades certes dades degut a la manca de les imatges que corresponen a certes etiquetes, la manca de certes etiquetes i finalment per a equilibrar el conjunt de dades.

També hi ha una segona base de dades que està composta per dos bases de dades diferents: Hm Hospitales[9] i RSNA [8]. Només consta de positiu i negatiu de CoVid-19 i el seu propòsit és simplement la comparació amb la base de dades anteriorment citada.

Finalment per a la xarxa CycleGAN, s'utilitzarà la mateixa Base de dades BIMCV. Dins d'aquesta base de dades, només agafarem els pacients que hagin passat per diverses sessions radiogràfiques i ordenarem les sessions de cada pacient segons el seu ordre cronològic. Obtenint un conjunt de pacients cadascun amb un grup d'imatges ordenades

6.4 Tractament de les dades

Llibreries usades Primerament el que farem serà introduir les diverses llibreries tant externes com del propi Python que utilitzarem en aquesta part del projecte.

- Pandas
- Pillow(Incorporada dins de Python)
- Matplotlib
- Numpy
- re(Expressions regulars)

6.4 Tractament de les dades

Seguidament es volen obtenir les dades que estan situades en dos fitxers diferents amb extensió ".tsv" on un guarda la informació referent a les radiografies dels pacients positius en CoVid-19 i un altre que guarda la informació de les radiografies referents a pacients negatiu en CoVid-19. Un cop es tenen ubicats aquest dos fitxers en memòria, s'utilitzarà funció **read_csv** de la llibreria **pandas** per poder llegir les dades, obtenint varis milers de línies de 9+1 columnes.
Podríem imaginar l'estructura amb la que guardem i modifiquem dades similar a una matriu de 10x45000, on cada columna representa un atribut diferent i cada línia una sessió de RX(Rajos X) diferent.
El primer problema que trobarem és que no llegeix correctament les separacions del fitxer i es crea una columna que no existeix al ".tsv", la correcció és molt simple. Vegeu la següent Figura 6.30:

```
df_P=pd.read_csv(path_P,sep="\t")
df_N=pd.read_csv(path_N,sep="\t")
df_P=df_P.drop("Unnamed: 0",axis=1)
df_N=df_N.drop("Unnamed: 0",axis=1)
```

Figure 6.30: Solució al problema de lectura

Com es pot veure a la Figura superior, el que s'ha fet és afegir un **paràmetre "sep"** a la lectura que estableixi les separacions utilitzant el caràcter "\t".
Seguidament per extreure la columna inexistent bàsicament s'executa un cop el codi i trobem el nom de la columna. Un cop obtingut, s'utilitza la **funció "drop"** de Pandas per eliminar la columna utilitzant el seu nom.

Seguidament apareixen dos problemes més, les comes que hi hauria d'haver dins de les taules apareixen com a "\t" i la clau[33] de cada imatge té una estructura diferent entre els positius i els negatius. Vegeu la solució emprada en la Figura 6.31.

Si ens fixem en les dos primeres línies, el que es fa és aplicar a tot el conjunt de dades, una funció que **canvia** totes les seqüències "\t" per el caràcter ",". D'aquesta manera es soluciona el primer problema.
En el cas de l'estructura de "Claus Principals"[34] s'ha creat una funció anomenada **corrector_SES**, que el que fa és que totes les "Claus Principals" que començaven per "SES-", se'ls hi elimina aquesta primera part. Cal destacar que primerament s'ha investigat que no hi hagin claus iguals al eliminar aquesta part i no és el cas.

El següent pas que es fa és fer una **concatenació dels conjunts de dades CoVid-19 positives i CoVid-19 negatives**, però abans d'adjuntar-los el que s'ha de fer és posar una nova

[33] Seguit de caràcters limitats que identifiquen un objecte.
[34] Conjunt de claus úniques on cadascuna representa només a un objecte dins d'un conjunt

```
##Corregir les \t
df_N=df_N.apply(lambda x: x.str.replace("\t",","))
df_P=df_P.apply(lambda x: x.str.replace("\t",","))
#Corregir incoherencia del format de SessionID (alguns són "SES_($ID)" alguns "$ID" )
def corrector_SES(df):
    pattern=re.compile("^SE|^se")
    for i,row in df.iterrows():
        if pattern.match(row['ReportID']):
            df.loc[i,'ReportID']=row['ReportID'][4:]

corrector_SES(df_P)
```

Figure 6.31: Solució al problema de les separacions i de l'estructura de claus.

columna anomenada "COV-19" que ens diu si aquella línia pertanyia en el conjunt de positius o de negatius de CoVid-19.
Finalment es té un últim problema durant el processament de les dades, el nom dels fitxer que contenen les imatges que corresponen a cada sessió estan en un 3er i 4rt fitxer. Per tant utilitzant les funcions que podem veure en la Figura 6.32 següent, podem solucionar aquest problema.

```
##Function to obtain obtain the pacient attributes
def get_data_URLfiles(path):
    data_df=pd.read_csv(path,usecols=[0],sep='"')
    secondary=data_df.columns[0].split(';')
    data_df.columns = ["Test"]
    data_df = data_df.Test.str.split(';',expand=True).apply(pd.Series)
    data_df.columns= secondary
    return data_df

##Function to store URLS
def copy_Urls(iterable_df,storage_df,path):
    for _,row in iterable_df.iterrows():
        row_id=storage_df.index[storage_df['ReportID']==row['Session'][4:]] ##Omitim els caracters del 0-3
        file_path=path+"/processed/"+row['File']+"_8b.png"
        if(os.path.isfile(file_path)):#Mirem si existeix la imatge sinó eliminem el registre
            storage_df.loc[row_id,'File']=file_path
        else:
            storage_df.drop(row_id,inplace=True)
```

Figure 6.32: Relacionar les sessions amb les seves corresponents imatges.

En termes generals el que es fa en la primera funció és obtenir d'un fitxer de dades els nom de les imatges i a quina sessió pertanyen utilitzant les "Claus Principals". I guarda aquesta informació en una estructura de dades del "pandas".
Per altre banda el que fa la segona funció és donada l'estructura de dades que acabem d'obtenir, crear una nova columna o atribut dins de les dades que havíem obtingut anteriorment on posem el nom de l'imatge que li pertoca a cada sessió segons la seva "Clau Principal". També com podem veure hi ha un condicional (Línia que conté: "if(os.path.isfile(file_path)):") que mira si existeix o no l'imatge corresponent a l'objecte. Si l'imatge no és trobada s'elimina l'objecte dins del conjunt de dades.

6.5 Anàlisi de les dades

En aquest apartat el que es farà serà analitzar les diverses dades que s'han obtingut i intentar trobar punts que ens puguin ser d'interès.
El primer que es fa és mirar la distribució de casos positius i negatius de CoVid-19 que tenim.

Positius	Negatius	Total
8731	4243	12974

Table 5: Distribució positius-negatius

Tal com es pot apreciar a la Taula 5, en el pas anterior s'han descarta aproximadament uns 500 objectes del conjunt de dades, passant de 19000 a 12974. També podem veure que el nombre de positius és bastant superior al negatius. Tenir un **nombre desbalancejat** de dades positives i negatives **no és un cas ideal per l'entrenament de xarxes neuronals**. Per tant posteriorment s'hauran de tractar les dades de manera que obtinguem un conjunt de dades balancejades 50/50 de positius i negatiu.

Observant les dades també es veu un altre problema, **algunes dades considerades negatives tenen una etiqueta que les concep com a positives**. Per tant durant el procés de tractament de dades s'haurà d'investigar quines són les més òptimes.
L'ideal seria utilitzar de quin conjunt venen(positiu o negatiu), però si hi ha etiquetes que ens informen que aquella imatge va ser considerada com a CoVid-19 positiu per un radiòleg, vol dir que la xarxa el més segur és que obtingui uns resultats similar. En la següent Taula 6 podem observar el percentatge d'aquest diguem'ls-hi "Falsos-Negatius".

Tal com podem veure la quantitat de "Falsos-Negatius" és inferior al 20%, però aquest valor pot ser molt rellevant ja només aquest petit percentatge pot causar que la xarxa es pensi que cert tipus de positius són negatius i viceversa. Per tant haurem d'explorar quina és la millor distribució de les dades.

Falsos Negatius	Negatius	Total
812	3431	4243

Table 6: Distribució positius-negatius

Finalment en la següent Taula 7 veurem com estan distribuïdes les dades en nombres:

		Conjunt		
		Positiu	Negatiu	Total
Etiquetes	Positiu	8732	0	8732
	Negatiu	812	3431	4243
	Total	9543	3431	

Table 7: Distribució de les dades

6.6 Paràmetres

Primerament és necessari conèixer quins paràmetres són els més rellevants en l'aprenentatge d'una xarxa neuronal i per a què serveix cadascun.

6.6.1 Learning Rate

El paràmetre de "Learning Rate" ja s'ha explicat amb anterioritat dins la Secció 4.6.3, però es necessari destacar certs canvis que apareixen entre al teoria i la pràctica.
Tal com s'ha dit en anterioritat el "Learning Rate" pot causar que una xarxa divergeixi i mai pugui arribar al mínim local[35] o vagi molt lenta i necessiti moltes iteracions per a poder arribar-hi.
La diferència substancial és que el propi PyTorch **té un reductor del "Learning Rate"** que s'aplica cada cert nombre d'iteracions, el nom que se'ls hi dona a aquest reductors és **"Schedulers"**.
El PyTorch té un Scheduler prèviament implementat però els programadors tenim la capacitat de modificar-lo d'entre els que se'ns ofereixen o crear-ne un nosaltres mateixos. En aquest cas en particular d'aquest projecte s'han usat els schedulers ja implementats.

[35]Punt on la derivada és 0 i està situat en una corba que per ambdós costats és creixent

6.6.2 Funció de pèrdua

La funció de pèrdua és una fórmula matemàtica que s'utilitza per a obtenir l'error resultant entre els resultats obtinguts i els desitjat. Aquest error resultant és el que s'utilitzarà posteriorment per a que la xarxa aprengui.
Aquest paràmetre s'estipula depenent del problema que estem intentant solucionar, per exemple:

- Problema de predicció de nombres reals, la millor funció de pèrdua serà la **Mitjana dels errors al cuadrat**
- Problema de classificació binària, la millor funció es una funció de pèrdua logarítmica.

6.6.3 Optimitzador

L'optimitzador és un paràmetre que té dos funcions principals, la primera és de variar el "Learning Rate" segons les iteracions d'aprenentatge que s'han dut a terme i finalment també la d'optimitzar el gradient de que propagarem cap endarrera perquè sigui d'aquesta manera més efectiu l'entrenament.
Existeixen molts optimitzadors i cadascun amb els seus avantatges i inconvenients per tant és un paràmetre que pot variar notablement el funcionament de la xarxa.

6.6.4 Transformacions

PyTorch ens ofereix unes funcions anomenades **Tranformacions**, serveixen per modificar les dades de manera que la xarxa pugi aprendre de manera més ràpida i eficient. També ens serveixen per modificar els imatges quan estan guardades dins de les funcions de PyTorch i ens estalvien l'ús de llibreries externes de processament d'imatges.
Les més utilitzades són:

- **Normalització**: La normalització és una funció que fixa el rang de valors de la imatge i treu els valors atípics de la imatge de manera que la xarxa pugui aprendre de manera més efectiva i eficient.
- **Rotació**: La rotació és una funció que rota les imatges un nombre de graus a l'atzar dins d'un limit establert.
- **RandomCrop**: El RandomCrop es basa en fer retallar la imatge inicial en una imatge més petita per evitar que la xarxa tingui "overfit".

6.7 Implementació del model Binari

Al començar amb al implementació de les xarxes necessitarem importar altres llibreries. Vegeu la Figura 6.33:

```
##Pytorch part
import torch
import torchvision
import torch.nn as nn
import random
import torch.optim as optim
from torchvision.io import read_image
from torchvision import transforms, utils
from tqdm.auto import tqdm, trange
import seaborn as sn
import warnings
from lr_finder import LRFinder
from lr_finder import TrainDataLoaderIter
from lr_finder import ValDataLoaderIter
```

Figure 6.33: Llibreries per a la implementació del model Binari

Un cop s'han explorat i preparat les dades per al seu ús, es començarà a implementar les funcions necessàries per l'entrenament de la primera xarxa neuronal la qual tindrà la funció de separar les imatges entre positius i negatius de CoVid-19.
Com s'ha mencionat anteriorment **és necessàri balancejar les dades** que tenim igualant el nombre d'imatges positives i negatives sinó podem tenir problemes en l'aprenentatge de la xarxa neuronal. Seguidament en la Figura 8 podem veure un exemple d'una matriu de confusió que mostra l'entrenament d'una xarxa utilitzant les dades sense balancejar:

		Classes predites	
		False	True
Classes actuals	False	0	871
	True	0	1294

Table 8: Matriu de confusió d'una xarxa desbalancejada . Precisió: 67.31%

6.7 Implementació del model Binari

Tal com podem veure en la figura anterior, la xarxa al veure que el nombre de casos positius era superior als casos negatius ha decidit optar per dir que tots els resultats són positius.
Per solucionar aquest problema s'ha creat una funció que iguala el nombre de imatges positives de CoVid-19 amb el nombre d'imatges negatives. Obtinguent així un conjunt de dades **positives i negatives de 9760 casos** on 4880 són positives i 4880 són negatives.

El següent pas és la creació d'una funció que separi les dades en grups d'entrenament. Un grup per entrenar, un per validar i un final per testejar. Per fer-ho el que es fa és dividir el conjunt de dades en parts. En aquest projecte s'ha decidit fer les porcions 80/10/10, 80% de dades en l'entrenament, 10% en la validació i un altre 10% en el testeig.

El següent pas és la creació d'un "Dataset", el "Dataset" de Pytorch és una estructura de dades que serveix per processar, obtenir i entrar a RAM les imatges del nostre conjunt de dades, també és on s'associa l'etiqueta de l'imatge amb la pròpia imatge. Seguidament en la Figura 6.34 podem veure la forma que tenen.

```
class OrigLabelsDataset_TF(torch.utils.data.Dataset):
    def __init__(self,dataframe, transform=None):
        self.transform=transform
        self.df=dataframe

    def __len__(self):
        return len(self.df['File'])

    def __getitem__(self,idx):
        if torch.is_tensor(idx):
            idx=idx.tolist()
        image=read_image(self.df.loc[idx,'File'])
        if self.transform:
            image = self.transform(image)
        return np.float32(image),self.df.loc[idx,'COV-19']
```

Figure 6.34: Exemple d'un Dataset de PyTorch

Com podem veure es dur a terme una sobreescriptura dels mètodes ja existents dins de la classe "Dataset" de PyTorch. La part més important és la funció "__getitem__" la qual ha de retornar una tupla conformada per (imatge,etiqueta). Cal destacar que s'han de transformar les dades en un **nombre "Real"** abans de retornar les dades.

Ara s'ha d'implementar el model que volem fer servir i extreure'n les dades que s'obtinguin en forma de matriu de confusió. Per crear el model ho podem resumir hi ha tres majors passos:

- Anàlisis: En aquest pas el model rep unes dades i ens dona uns resultats.
- Validació: En aquest pas es validen les dades i se'n extreu una pèrdua entre l'obtingut i l'esperat.

- Aprenentatge: La pèrdua es propaga cap endarrere modificant el pesos del model.

L'anàlisi el veiem representat com la funció **outputs=model(batch_img)** on el que es fa és que el model classifiqui les imatges entrades i guardi els resultats a la variable "output" .
La validació està representada amb la funció **"loss=loss_criterion(outputs,batch_labels)"** la qual calcula la pèrdua entre l'obtingut("outputs") i l'esperat("batch_labels"). **L'aprenentatge** està representat per la funció **loss.backward()** i el que fa es propagar cap endarrera el gradient generat per la validació.
Vegeu la Figura 6.35, requadrat en en negre es mostra l'anàlisis, en blau la validació i en vermell l'aprenentatge.

```
for (batch_img, batch_labels) in t:
    batch_img, batch_labels = batch_img.to(sel_gpu), batch_labels.to(sel_gpu).to(torch.long)
    optimizer.zero_grad()
    with torch.set_grad_enabled(phase == 'train'):
        outputs=model(batch_img)
        _, preds = torch.max(outputs.data, 1)
        #Optimize the CNN
        loss=loss_criterion(outputs,batch_labels)

        if phase=='train':
            loss.backward()
            optimizer.step()
    if phase=='train' and scheduler:
        scheduler.step()
    #Get data form the batch
    total_iterations+=batch_labels.size(0)
    true_batch += (preds==batch_labels.data).sum().item()
    total_loss +=loss.item()
    for i in range(0,len(preds)):
        conf_matrix[batch_labels.data[i]][preds[i]]+=1
    percentage=true_batch/total_iterations
    t.set_description(str.upper(phase)+"loss:"+str(loss.item())+" percentage: "+str(percentage))
    t.refresh()
```

Figure 6.35: Part de l'entrenament de la xarxa neuronal Binària

El següent i últim que farem amb la xarxa de classificació binària serà valorar si és millor utilitzar les etiquetes radiològiques o la classificació amb la que estaven ja separades.
Per a dur a terme aquest pas el que es farà serà fer un aprenentatge fent cas a les etiquetes i un altre aprenentatge fent cas al grup en el que pertanyien. I posteriorment es decidirà quin és el millor.

6.8 Implementació de la classificació multiclasses

En aquest apartat s'intentarà fer un model que pugui detectar múltiples etiquetes en una imatge donada.
Per poder-ho fer el primer que s'ha de fer és transformar totes les etiquetes que hi ha dins les dades i transformar-les en dígits. Aquest pas s'ha de dur a terme degut a que les xarxes neuronal no pot donar com a sortida paraules. El que fan és donar una possibilitat de pertànyer a cada classe. Per tant l'última capa de la xarxa neuronal tindrà tants nodes com etiquetes diferents hi ha dins les dades i ens donarà una possibilitat de que aquella imatge pertanyi a

cada classe.
També és necessari, si transformem cada classe en un nombre, tenir un diccionari, que faci la operació inversa, o sigui pugui correlacionar cada nombre amb l'etiqueta corresponent.
Vegeu la Figura 6.36 per veure com s'ha fet:

```
dic_labels={}
labels_to_add=[]
for n in range(len(df_labels)):
    for i in re.sub("[^a-zA-Z0-9,\ ]",'',str(df_labels.loc[n,'Labels'])).split(','):
        i=i.rstrip().lstrip()
        if not(i in dic_labels.keys()) and i!='' and i!='COVID 19' and i!='normal':
            dic_labels[i]=len(dic_labels)
            labels_to_add.append(dic_labels[i])
        elif i in dic_labels.keys():
            labels_to_add.append(dic_labels[i])

    df_labels.loc[n,'Ev_Labels']=str(labels_to_add)
    labels_to_add=[]
```

Figure 6.36: Transformació de les etiquetes a dígits representatius.

Dins el requadre blau de la Figura anterior, el que es fa és una iteració per totes les etiquetes que té un pacient, eliminant tots els símbols que no són lletres.
Seguidament en el requadre vermell, si es troba una etiqueta nova, es fa una nova entrada en el diccionari que guarda la relació entre etiqueta i un nou nombre. I seguidament es substitueix l'etiqueta per el nou nombre generat.

Seguidament hem de canviar el Dataset que havíem creat en l'apartat anterior perquè ara ja no tenim només una etiqueta sinó un vector amb moltes etiquetes dins.
Només serà necessari un petit canvi, vegeu mireu la Figura 6.37:

6.8 Implementació de la classificació multiclasses

```
def __getitem__(self,idx):
    if torch.is_tensor(idx):
        idx=idx.tolist()
    image=read_image(self.df.loc[idx,'File'])
    #image2=np.array(image)
    #image2=np.transpose(image,(1,2,0))
    #image2=Image.fromarray(image2)
    #plt.imshow(image2)
    #plt.show()
    if self.transform:
        image = self.transform(image)
    labels=self.df.loc[idx,self.col]
    labels=labels[1:len(labels)-1]
    if self.len!=1:
        labels=labels.split(',')
        labels=[1 if str(n) in labels else 0 for n in range(0,self.len)]
        labels = torch.from_numpy(np.asarray(labels))
        labels=np.float32(labels)
    else:
        labels=torch.tensor(int(labels))
        labels=np.longlong(labels)
    return (np.float32(image),labels)
```

Figure 6.37: Dataset del model multiclasse

Si ens fixem en el requadre blau dins la figura anterior veurem que s'han fet certs canvis respecte com es tractaven les etiquetes en el model anterior. El primer que es fa és, al estar les dades guardades en vectors, eliminar tan les comes com els claudàtors que separen al informació. Seguidament crear un vector amb tantes posicions com etiquetes hi ha on cada posició serà 0 menys les posicions que pertanyen a les etiquetes de la imatge entrada.

També serà necessari modificar l'algorisme d'entrenament del model ja que al modificar l'entrada s'haurà de tractar diferent i finalment també serà necesari un canvi en la matriu de confusió.

Ara apareix un gran problema, hi ha aproximadament moltes més de 50 etiquetes diferents i en la gran majoria dels casos cada imatge té 2 o 3 etiquetes. Per tant apareixarà el mateix problema que hem nombrat en el cas binari, la xarxa aprendrà a sempre dir que no té cap etiqueta.
Per a solucionar aquest problema s'ha acotat el nombre de etiquetes utilitzades a 8 etiquetes i seran totes les etiquetes que continguin: **pneumothorax, edema, consolidation, pleural effusion, atelectasis, interstitial, pulmonary mass, nodule**.

El que s'ha fet primer és seleccionar només les dades que tenen una d'aquestes etiquetes i eliminar tota la resta de dades. Aquesta acció es posa en pràctica perquè no és necessàri tenir imatges que no tinguin cap etiqueta de les seleccionades, no ens oferiran cap aprenentatge.
Seguidament es tornen a transformar totes les etiquetes en nombres tal com s'ha fet anteriorment, creant així un nou diccionari per aquestes etiquetes seleccionades.
És molt important en aquest punt utilitzar **una funció de pèrdua amb pesos**, PyTorch ens ofereix aquesta eina amb el nom de "BCEWithLogitsLoss" i intentar donar més importància als positius amb l'argument **"pos_weight"**. Vegeu la Figura 6.38:

```
loss_criterion=nn.BCEWithLogitsLoss(pos_weight=torch.tensor([1.5 for i in range(0,8)]).to(sel_gpu))
```

Figure 6.38: Introducció de la BCELoss amb Logits

El que fa aquesta funció de pèrdua és prioritzar que la xarxa tendeixi a donar més valors positius i no tants negatius. D'aquesta manera intentant semi-obligant-la a que no sigui tan conservadora donant sempre zero.

6.9 Implementació de classificador d'etiquetes

En aquesta implementació el que es vol aconseguir és fer un classificació binari com el primer problema que sàpiga decidir entre les 8 etiquetes que utilitzàvem en l'apartat anterior.
Bàsicament es crea un subconjunt per cada etiqueta usada en el conjunt de dades del problema anterior (Problema multiclasses) i posteriorment es distribueixen les dades usades segons l'etiqueta o etiquetes que pertanyen. Seguidament utilitzant un model de classificació binària intentarem comprovar si cadascuna d'aquestes classes pot ser classificada de manera binària o no.
El problema principal és la necessitat de balancejar el conjunt de dades i afegir dades negatives. Per a fer-ho es crea una funció que per una etiqueta donada obté un nombre igual al nombre d'objectes que tenen aquella etiqueta, d'objectes que no la tinguin. Vegeu la Figura 6.39:
La línia dins la figura anterior que està dins d'un requadre negre, és un condicional que busca

```
def get_extra_data(inverter,dataset,label,length,initial,column='Ev_Labels'):
    count_picked=0
    count=0
    while count_picked<length:
        if not(re.match(('[^0-9]+'+inverter[label]+'[^0-9]+'),str(dataset.loc[count,column]))):
            count_picked+=1
            aux=dataset.loc[count]
            aux[column]="[0]"
            initial=initial.append(aux)

        count+=1

    return initial.reset_index(drop=True)
```

Figure 6.39: Funció d'obtenció d'un nombre donat de dades que no pertanyin a una etiqueta donada.

per una dada donada si té o no l'etiqueta que volem classificar. En cas que no la tingui l'afegim a les dades d'entrenament. En cas que la tingui no farà res perquè ja la tindrem com a positiva. Finalment quant el nombre de positius i negatius és igual la funció pararà.

6.10 Implementació de la CycleGAN

L'implementació de la CycleGAN en aquest projecte, té com a punt principal intentar predir com un pacient evolucionarà en el futur.
Per poder ensenyar a una màquina a predir l'evolució d'un pacient necessitem imatges que estiguin correlacionades en el temps. Per tant el que s'ha fet, és explorar les dades que tenim i

extreure'n només aquelles que tinguin un registre posterior amb una imatge associada.
Per portar-ho a terme hem utilitzat la següent funció, vegeu Figura 6.40:

```
sesion_user={}
for i,row in df_Pred.iterrows():
    if(row['PatientID'] in sesion_user.keys()):
        sesion_user[row['PatientID']].append(row['ReportID'])
    else:
        sesion_user[row['PatientID']]=[row['ReportID']]

sesion_user={key:sorted(value) for (key,value) in sesion_user.items() if len(value)>1}
```

Figure 6.40: Funció de selecció de data amb registres posteriors

Si ens fixem en la Figura anterior, podem veure una funció que itera per tots els casos dins les dades que tenim. En cada iteració busca si el pacient en el qual iterem, està registrat si no ho està i creem una nova entrada amb el seu id i li associem el cas iterat. En cas de tenir-lo registrat només associa el cas a el pacient.
El requadre blau és la part de la funció que busca si tenim el pacient registrat.
Mentrestant la part dins el requadre vermell borra del registre tots els pacients que només tinguin un cas associat amb ells i ordena els casos que tenen associats.
Seguidament s'ha de preparar el DataSet en aquest cas és més simple que l'anterior, tal com s'ha dit a la Secció 4.9, les CycleGAN reben tan d'entrada com d'etiqueta una Imatge, per tant l'únic canvi serà que s'haurà de carregar una imatge en comptes d'un digit. També en aquest en el cas d'aquest projecte passarem el camí de cada imatge per a posteriorment poder-les mostrar quan validem.
On hi ha un canvi substancial és en l'entrenament del model. El meu model de CycleGAN es basa en el model implementat dins d'aquesta Web[29].
El primer i més necessàri és importar les llibreries de la web citada anteriorment. Les quals es mostren en la següent Figura 6.41 es mostren:

```
import CGAN.networks as networks
import itertools
from CGAN.util.image_pool import ImagePool
```

Figure 6.41: Llibreries usades pertanyents a [29]

6.10 Implementació de la CycleGAN

El següent punt és la creació de les xarxes que compondran la CycleGAN, que tal com s'ha explicat són 4 (2 classificadors i 2 GANs). Vegeu la Figura 6.42:

```
self.netGA= networks.define_G(in_channels,out_channels,ngf,net['G'],norm,use_dropout,'normal', init_gain,gpu)
self.netGB= networks.define_G(out_channels,in_channels,ngf,net['G'],norm,use_dropout,'normal', init_gain,gpu)
self.netDA= networks.define_D(out_channels,ngf,net['D'],opt['layers'],norm,'normal', init_gain,gpu)
self.netDB= networks.define_D(in_channels,ngf,net['D'],opt['layers'],norm,'normal', init_gain,gpu)
```

Figure 6.42: Creació de models que componen la CycleGAN

Seguidament el que s'ha de fer és cridar les funcions de pèrdua i els optimitzadors i guardar-los en les seves variables corresponents.Un cop guardades s'ha d'implementar el mètode de validació i el de propagació cap endarrera, tots dos són molt necessaris en qualsevol entrenament de xarxes neuronals. El gran problema resideix en que en la CycleGAN, la propagació cap endarrera, és molt diferent a la de les xarxes normals. Donat que a la xarxa li entren dos imatges A i B on A és una **Imatge prèvia** i B és una **Imatge en temps posterior**. Les imatges B (imatges obtingudes posteriorment) representen imatges posteriors mentrestant les imatges A (imatges obtingudes prèviament a els imatges B) representen imatges prèvies.
Per a explicar el següent punt s'estipularan 4 abreviacions. Com s'ha explicat a la Secció 4.9, les CycleGANs estan conformades per 4 subxarxes. Dos Xarxes Convolucionals complertes que les anomenarem GAN-A i GAN-B i dos xarxes convolucionals de classificació binària que les anomenarem D-A i D-B.
Seguidament s'explicara de manera molt simplificada els passos que fa l'algorisme per aprendre a transformar les imatges són:

- S'entra l'imatge A en la GAN-A i es transforma en un imatge, que anomenarem "Imatge AB", que s'intenta semblar en una imatge posterior.
- Es classifica si "l'imatge AB" es pot considerar una imatge "posterior". I es calcula la pèrdua.
- Seguidament s'intenta transformar un altre cop "l'imatge AB" a l'imatge A que teníem i es calcula l'error durant la transformació.
- S'entra l'imatge B en la GANB i es transforma en un una imatge prèvia la qual anomenarem "Imatge BA".
- Seguidament es classifica si "l'imatge BA" es pot considerar una imatge "prèvia". I es calcula la pèrdua.
- Acabem intentant transformant un altre cop "l'imatge BA" a l'imatge B i es calcula l'error durant la transformació.
- Per finalitzar es sumen totes les pèrdues i errors i es propaguen cap endarrera.

7 Resultats

En aquest apartat es mostraran els resultats obtingut en cada model emprat, l'entrenament sempre s'ha aturat quan les pèrdues entre l'entrenament i la validació han divergit i en tot cas s'ha usat el buscador de "Learning Rate" presentat en la Secció 5.10.

7.1 Resultats model de classificació Binari

En aquest apartat mostrarem els resultats obtingut per els models de classificació binaria i com han desenvolupat la seva pèrdua en el temps.
Tal com s'havia trobat amb anterioritat aquest classificador podia ser entrenat segons les etiquetes radiològiques o segons l'etiqueta que ens havien donat.
Per a aquesta raó hem entrenat el model en ambdós casos per veure quin resultat era el més òptim. Els paràmetres que s'han emprat per obtenir aquest resultats an sigut els següents:

- **Optimitzador**: Adam
- **Tamany de les imatges**: 600x600
- **Transformacions**: Rotació de les imatges $10^{\underline{o}}$ aleatòriament i Normalització de l'imatge.
- **Funció de pèrdua**: CrossEntropyLoss
- **Model**: ResNet34

I amb aquest paràmetres'han obtingut els següents resultats:

Depenent de les etiquetes inicials

En aquest cas s'han utilitzat les etiquetes inicials de CoVid-19 o no CoVid-19 per a entrenar el model. Es mostrarà quin ha sigut el millor resultat que hem arribat i com s'ha comportat la pèrdua durant l'entrenament d'aquest.

7.1 Resultats model de classificació Binari

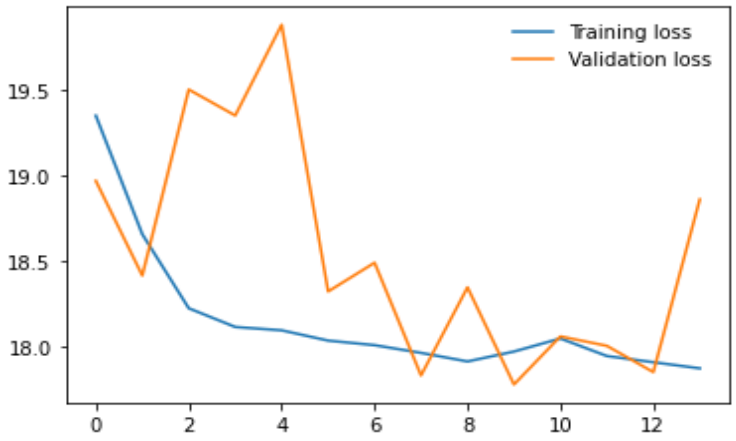


Figure 7.43: Comportament de les pèrdues durant l'entrenament

Tal com podem veure ne la Figura 7.43 l'entrenament ha durat bastantes èpoques(iteracions). Tal com podem veure la 13 iteració les pèrdues han divergit i s'ha aturat l'entrenament. Vegeu les Figura 9 i 10 per veure els resultats obtinguts:

		Classes predites: False	Classes predites: True
Classes actuals	False	2470	1434
	True	1502	2402

Table 9: Matriu de confusió. Entrenament Etiqueta Original. Precisió: 62.397%

		Classes predites: False	Classes predites: True
Classes actuals	False	328	160
	True	210	278

Table 10: Matriu de confusió. Validació Etiqueta Original. Precisió: 62.090%

7.1 Resultats model de classificació Binari

Tal com podem veure, el model prediu correctament aproximadament un 62% de les imatges en l'entrenament i aproximadament un 62% durant la validació. Posteriorment haurem de comparar aquest resultats amb els obtinguts en el següent apartat.

Depenent de les etiquetes radiològiques

En aquest cas s'han utilitzat les etiquetes radiològiques per a entrenar el model. Seguidament mostrarem com s'ha comportat el valor de pèrdua durant l'entrenament i tal i com hem fet amb l'anterior en el moment en què divergeixi pararem l'entrenament.

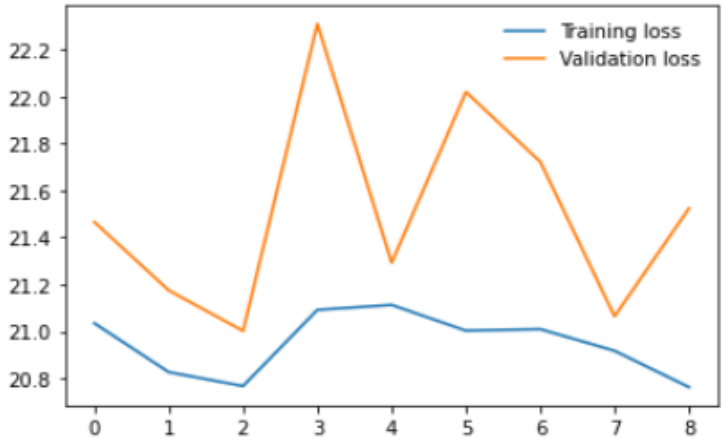


Figure 7.44: Comportament de les pèrdues durant l'entrenament

Tal com podem apreciar en aquest model amb 8 iteracions ha sigut suficient per arribar a una divergència entre les pèrdues de la validació i de l'entrenament.
Els resultats de les matrius de confusió són els següents:

		Classes predites: False	Classes predites: True
Classes actuals	False	2212	1692
	True	1660	2244

Table 11: Matriu de confusió. Validació Etiqueta Original. Precisió: 57.069%

		Classes predites: False	Classes predites: True
Classes actuals	False	270	218
	True	192	296

Table 12: Matriu de confusió. Validació Etiqueta Original. Precisió: 57.991%

En el cas de l'aprenentatge utilitzant les etiquetes radiogràfiques no hem obtingut un resultat tant bo com el que havíem obtingut amb l'anterior. Per tant podem entendre que la xarxa aprèn més fàcilment a classificar segons el conjunt segons les etiquetes no radiogràfiques. Per tant per a dur a terme la classificació binària s'utilitzaran aquest conjunt d'etiquetes.

La primera millora que s'ha provat és en incrementar la profunditat del model, però degut a la mida del model hem hagut de reduir la mida de les imatges. Aquesta prova ens ofereix la capacitat de valorar quin component és més rellevant si la profunditat del model o la resolució de les imatges.

		Classes predites: False	Classes predites: True
Classes actuals	False	2575	1329
	True	1401	2503

Table 13: Matriu de confusió. Entrenament Etiqueta Original ResNet101. Precisió: 65.035%

Si ens fixem amb la Figura 14, veurem l'encert obtingut de la validació, entrenant una xarxa amb més profunditat. La prova que s'ha dut a terme ha sigut implementar una de les xarxes més profundes de ResNet en aquest cas la ResNet101 i baixar al resolució de les imatges a [400x400] píxels. S'ha hagut de baixar la resolució perquè sinó la memòria de la gràfica no era suficient.
Com podem veure els resultats no han sigut millors tot i utilitzar una xarxa amb més paràmetres. Per tant podem assumir que és millor mantenir-se amb el model anterior de més resolució i menys profunditat.

		Classes predites: False	Classes predites: True
Classes actuals	False	333	155
	True	256	232

Table 14: Matriu de confusió. Validació Etiqueta Original ResNet101. Precisió: 57.889%

Degut a aquests resultats es van voler fer proves amb altres conjunts de dades de CoVid-19, les proves es van dur a terme utilitzan les bases de dades de Hm Hospitales[9] i RSNA [8] . Els resultats que es varen obtindre van ser bastant satisfactoris. Vegeu les Figures 15 i 16:

		Classes predites: False	Classes predites: True
Classes actuals	False	1759	271
	True	204	1826

Table 15: Matriu de confusió entrenament. Precisió: 88.305%

		Classes predites: False	Classes predites: True
Classes actuals	False	503	31
	True	32	502

Table 16: Matriu de confusió validació. Precisió: 94.101%

Com podem veure en la matriu de validació s'ha assolit un valor superior al 94.101% d'encert el

qual és un resultat molt bo per un classificador i segurament amb més "fine tunning" es podria arribar al 95% d'encert. Cal destacar que l'entrenament només ha arribat al 88.305% per tant ens diu que l'obtenció del 94.101% té un cert factor sort. Per tot i així segueix sent un molt bon valor.

7.2 Resultats model de classificació Multiclasses

Els resultats derivats del classificador de múltiple classes tampoc han sigut els esperats. El model tendia a donar sempre un valor negatiu. Hi ha dos raons principals una és que el model no troba el patró correcte per separar les imatges i acaba donant sempre un resultat negatiu. L'altre és que degut a la quantitat de diverses etiquetes que ha de tractar no aprèn correctament.

Vegeu els resultats obtinguts en la Figura 17:

		Classes a predir							
		Consolidation	Pleural effusion	Interstitial pattern	Pneumothorax	Pulmonary edema	Atelectasis	Nodule	Pulmonary mass
Classes actuals	Falsos positus	0	2	0	0	0	2	2	2
	Falsos negatius	938	710	1114	40	14	162	88	36
	Verdaders negatius	2164	2392	1990	3064	3090	2940	3014	3066
	Verdaders positius	2	0	0	0	0	0	0	0

Table 17: Matriu de confusió. Validació Multiclasse . Precisió: 87.467%

Com podem veure el percentatge d'encert és molt alt, però en aquest cas no vol dir res. Al haver-hi tants valors negatius el model quasi sempre dona 0 d'aquesta manera obté un 80% llarg de precisió.
L'important i el que ens hem de fixar és amb la matriu 4x8 que tenim, la qual ens informa que la xarxa té 8 Falsos Positius, 3102 Falsos Negatius, 21720 verdaders negatius, 2 verdader positius. Per tant tot i que sembla que té un bon encert en veritat falla quasi tots els cops quan ha de predir una etiqueta. I si calculem la sensibilitat de la matriu, que és el numero de casos positius que la xarxa ha classificat correctament, obtenim **una especificitat de 0.000644** la qual és molt baixa tenint en compte que el millor valor és 1.

7.3 Predicció d'evolució amb la CycleGAN

Els resultats de la CycleGAN són més difícils d'evaluar ja que depèn d'una persona que els validi. Al no tenir unes labels que ens diguin quin és exactament el percentatge d'encert, només ens podem guiar per a la pèrdua de l'entrenament i aquesta mesura és relativa a la passada iteració de la xarxa.
Quan es diu que la pèrdua és relativa a la passada iteració, és perquè la pèrdua és una mètrica que mostra com ha millorat la xarxa respecte a la iteració anterior. Però diferents paràmetres poden causar una pèrdua inicial diferent i un model amb una pèrdua major pot tenir uns millors resultats i viceversa. Per tant l'ideal seria que un expert ens digues com de bé funciona el sistema.

En el conjunt de dades utilitzades no es té unes dates donades de en quin moment exacte es fa cadascuna de les radiografies per tant els resultats no mostraran una predicció exacte de la progressió del pacient. Sinó una imatge posterior a la donada amb un temps indefinit. En el cas que es tinguessin les dates de les radiografies es podria dur a terme una xarxa que pogués tenir un temps determinat de predicció. I entrar tantes vegades l'imatge dins la xarxa fins a aconseguir el temps desitjat de predicció.
Seguidament s'introduiran dos xarxes creades amb diversos paràmetres per veure si suposaven aquest un canvi rellevant a la sortida:

		Parametres		
		Lambda_id	Lambda_A	Lambda_B
Proves	Prova Conservativa	0.5	10	10
	Prova Avara	1	7	10

Table 18: Paràmetres de la CycleGAN importants en l'aprenentatge.

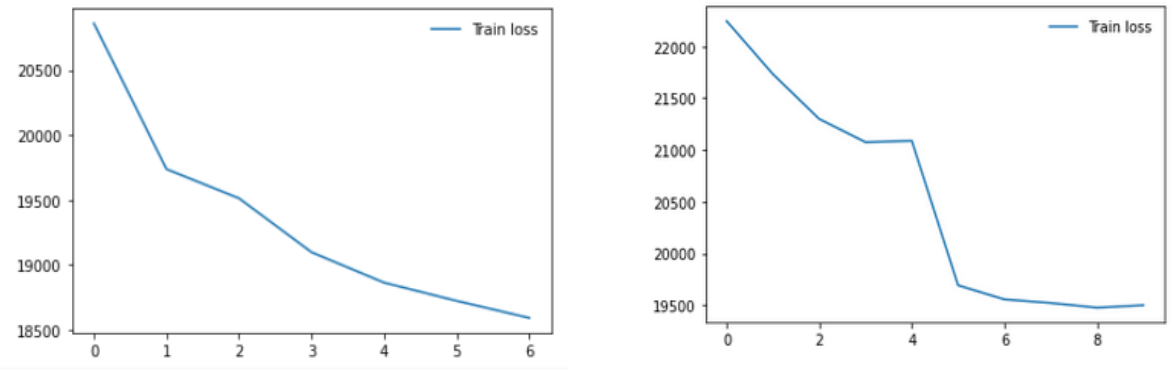
Cal recordar abans de tot que una xarxa neuronal de tipus CycleGAN té dos entrades una de "Tipus A" i una altre de "Tipus B" tal com es menciona a la Secció 6.10.
Seguidament s'explicarà quina funció tenen els paràmetres que es poden veure a la Taula 18 i en qué l'aprenentatge de la xarxa neuronal. Tots tres són pesos de les funcions de pesos que ajuden a entrenar la xarxa. Els podríem definir d'aquesta manera:

- **Lambda identitat**: Serveix per a mantenir el color de la imatge Tipus A. Al tenir imatges en escala de grisos en teoria no hauria de tenir un efecte important. Però al ser

imatges en escala de grisos de 3-dimensions podria tenir un efecte en mantenir els els grisos que pertoquen.

- **Lambda_A**: Com més gran és aquesta lambda més rellevant serà l'aparició de característiques de la imatge Tipus A en la imatge de sortida.
- **Lambda_B**:Com més gran és aquesta lambda més rellevant serà l'aparició de característiques de la imatge Tipus B en la imatge final.

Per tant el que es vol aconseguir amb la Prova Avara és donar més importància a les dades pertanyents a la sortida o Tipus B d'aquesta manera podent obtenir imatges més similars a aquestes. Tot i que amb menys percentatge, es seguirà tenint en compte les imatges inicials. Seguidament es mostrarà com han evolucionat les pèrdues en cadascuna de les dos xarxes implementades:



(a) Prova Conservadora

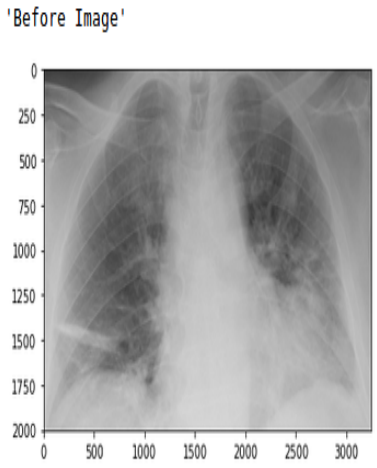
(b) Prova Avara

Figure 7.45: Comparació entre les pèrdues durant l'entrenament de les dos xarxes CycleGAN

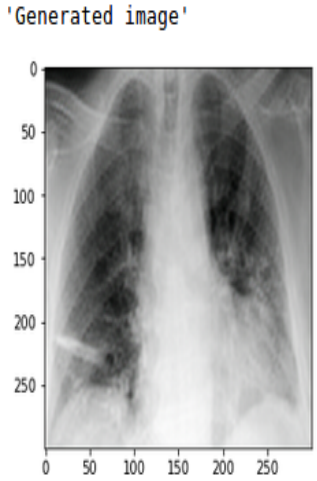
Tal com podem veure l'estructura de la pèrdua i el temps d'entrenament ha sigut molt variat. En tots dos casos s'utilitza el model obtingut abans de la divergència de la funció de pèrdua (El moment quant la línia blava torna a pujar). Ja que entrenar després de la divergència no sol aportar millors resultats.

7.3 Predicció d'evolució amb la CycleGAN

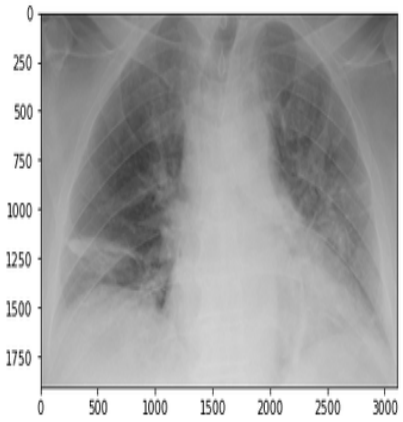
Seguidament mostrarem dos exemples de resultats per a cada xarxa, primerament començarem per la xarxa "Prova Conservadora":



(a) Imatge d'entrada



(b) Imatge generada



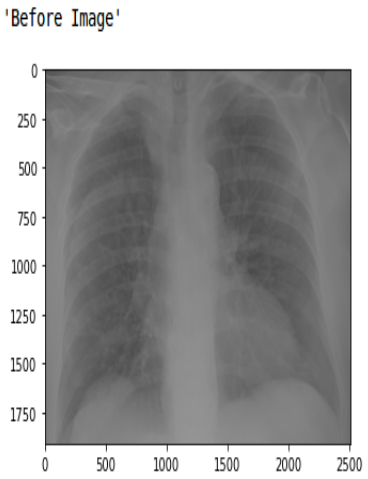
(c) Imatge esperada

Figure 7.46: Resultat de l'aplicació de la CycleGAN Prova Conservadora

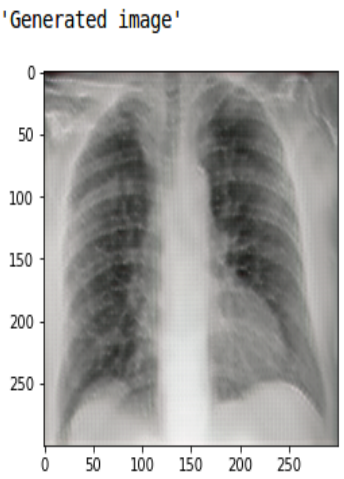
Tal i com podem veure en la Figura 7.46 anterior el model fa una petita millora en l'estat dels pulmons de la figura d'entrada, fent-la una mica més similars a la foto desitjada. Però si ens fixem en les taques dins del pulmó, anomenades també densitats, en certes àrees no fa un canvi el suficientment rellevant

7.3 Predicció d'evolució amb la CycleGAN

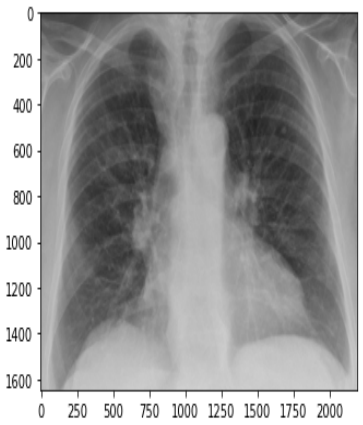
Seguidament es mostraran l'imatge original, l'esperada i l'obtinguda per la xarxa CycleGAN "Prova Avar":



(a) Imatge d'entrada



(b) Imatge generada



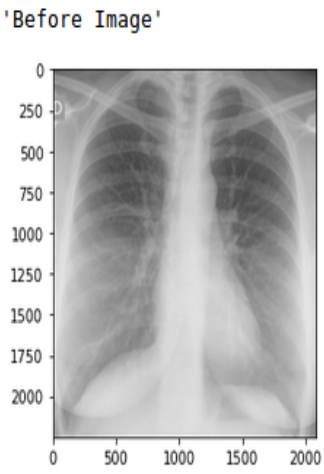
(c) Imatge esperada

Figure 7.47: Resultat de l'aplicació de la CycleGAN Prova Avar

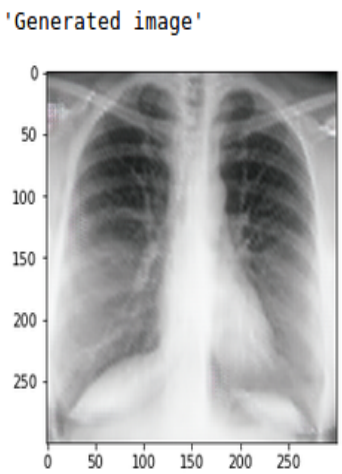
Com podem veure els resultats són bastant satisfactoris, fins al punt que crec que podria ser utilitzada si s'optimitzés més i podria donar resultats bastant òptims. Tot i això crec que s'hauria d'aplicar filtres de passa baixa (comentats en la Secció 4.6.4) per a reduir el soroll de l'imatge ja que és bastant alt. Tot i això cal destacar que al ser una xarxa menys conservativa tendeix també a crear molt més soroll en els casos més difícils, el qual és un punt a tenir en compte.

7.3 Predicció d'evolució amb la CycleGAN

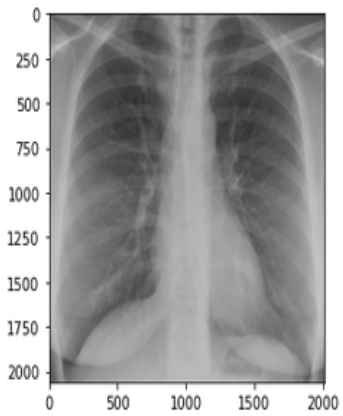
Tot i això en els casos més simples o que menys diferencia hi ha la xarxa no crea tant soroll i les imatges són molt correctes. El problema que el ser menys diferents la xarxa tampoc canvia l'imatge tant com seria desitjat que ho fes. Vegeu la Figura 7.48:



(a) Imatge d'entrada



(b) Imatge obtinguda



(c) Imatge esperada

Figure 7.48: Resultat de l'aplicació de la CycleGAN Prova Avar, amb imatges més simples

8 Conclusions i treball futur

8.1 Conclusions

Tal com s'ha mencionat a la secció de Motivacions 1.1, sempre m'ha interessat l'àmbit de la imatge mèdica i més si està unit a la informàtica. Per tant poder dur a terme un treball que agafi parts dels dos mons ha sigut una molt bona experiència. Tot i això en els punts on no es trobava un resultat adient o un resultat esperat costava avançar. Segurament el fet de buscar solucions i que no es puguin obtenir resultats òptims és el punt més difícil durant el procés de desenvolupament d'un TFG de recerca.

Una altre part que crec que es important de ressaltar i al final es fa bastant interessant, és llegir articles sobre els temes que es toquen. En el cas del CoVid-19 vaig aprendre l'existència de variants que desconeixia i que al ser minoritàries molts cops es desconeixen (vegeu la Referència[5]). També el fet que certes grips comunes són ocasionades per virus pertanyents a la família Coronaviridae, la mateixa família que el conegut CoVid-19.
El diagnosis radiogràfic també és una part que vaig trobar important. Molts cops les pròpies radiografies durant les primeres setmanes després de l'inici de la malaltia no poden detectar res. I quan es detecten normalment es pot definir que és CoVid-19 per el fet de fer "consolidacions"[36] rodones que estan situades en ambdós costats en la perifèria i sobretot en les parts inferiors(Vegeu la Referència[7]).
La part però que més em va interessar va ser l'estudi de les CycleGAN. No va ser un estudi molt òptim amb termes de temps degut a que el vaig aprendre analitzat el codi que podem trobar en aquesta Referència [29].
A més l'informació tècnica sobre aquestes xarxes és molt escassa en internet, si que existeixen llocs que expliquen el seu funcionament o quins són els seus punts principals. Però a l'hora d'implementar és difícil trobar informació al respecte.

També m'agradaria fer un petit incís en les parts que crec que aquest projecte m'ha aportat:

- La lectura de codi o articles: Actualment, almenys en el meu cas, tendim a buscar solucions als problemes molts cops directament buscant directament l'error a "Google"[37]. Oposadament en aquest treball no s'ha pogut aplicar aquest procés en quasi cap solució. I molts cops d'havia d'obtenir la informació a partir d'articles o extraient codi de programes ja implementats.
- Canvia de nou la base del programa: Molts cops quant volem canviar un programari el que es fa es simplement un "pegat" (una funció que solucioni el problema) i llestos. En el

[36] És un punt atenuat dins la imatge Radiogràfica que borra els vasos sanguinis i les parets de la via aèria.
[37] Motor de cerca online

cas de les xarxes neuronals potser que l'error es basi en no aplicar la xarxa correcta. Un exemple és la xarxa "CycleGAN" en un principi volia solucionar el problema amb una U-net però els resultats eren borrosos o distorsionats. I la solució va ser buscar una xarxa totalment diferent per a poder-ho dur a terme.

- Paciència: Normalment i sobretot amb els ordinadors actuals compilar el codi a no ser que siguin de magnituds molt grans és qüestió de minuts. Contràriament una xarxa neuronal tarda hores en entrenar-se sobretot la CycleGan que necessita tota una nit per a poder ser entrenada correctament.

Seguidament m'agradaria parlar sobre els objectius. El primer objectiu, un algorisme que pugui de manera correcta valorar si un pacient sofreix o no CoVid-19,**s'han assolit però tenen bastant de marge de millora**, tot i això els resultats trobats són massa baixos per ser considerats uns resultats que puguin ser usats en un hospital o un centre que necessitin implementar-lo.
Seguidament m'agradaria parlar sobre la comparació que s'ha fet amb anterioritat amb els valors obtinguts entre la xarxa classificadora binaria utilitzada per a classificar la base de dades BIMCV i la base de dades creada entre HM Hospitales i RSNA **??**.
Tinc tres hipòtesis possibles del perquè els resultats són tan diferents:
Una és que el conjunt de dades que s'ha decidit utilitzar **és massa complicat per la xarxa**, el que causa que la xarxa no pugui aprendre i es quedi amb "Underfit". La solució a aquest problema seria explorar les imatges del conjunt de dades i **extreure els cassos atípics** que causen que la xarxa no aprengui correctament. El que passa amb aquesta solució és que ja no es faria una xarxa per detectar el CoVid-19 sinó una xarxa per a detectar "certs" cassos de CoVid-19.
La segona és que potser que les etiquetes donades per aquest conjunt de dades poden tenir errors els quals causen que la xarxa no pugui trobar patrons dins les dades. Aquesta opció la veig difícil ja que venen d'una Base de dades d'un hospital clínic.
I l'última hipòtesis és que les dos bases de dades utilitzades en la segona comparació tinguin en algun punt de l'imatge un distintiu que no es veu a simple vista de a quina base de dades pertanyen. I al ser la base de dades RSNA la que aporta els cassos de CoVid-19 i la de HM Hospitales la pneumònia i pulmons sans. La xarxa determini fàcilment si són positius o negatius.

En el cas del **problema de multiclasses** els resultats tampoc han sigut òptims la xarxa tendeix a donar en gran majoria sempre negatiu en totes les classes que classifica. Per trobar la raó del perquè, vaig fer un altre model binari que classificava de manera binaria cadascuna de les etiquetes que tenen les imatges. I no vaig trobar resultats esperançadors.
Per tant, la xarxa de classificació de múltiples classes, sempre dona negatiu perquè no es poden classificar aquestes etiquetes i no pas perquè hi hagi masses classes. Per tant el que fa la xarxa és anar al millor resultat que és donar tot 0.

En el cas de la **CycleGAN** els resultats han sigut bastant correctes i són resultats que tot

i que alguns cops tenen cert soroll s'assemblen bastant a la imatge de sortida desitjada.
També s'ha pogut apreciar com hi ha un canvi bastant important depenent de si prioritzes que la xarxa tendeixi a ser més conservadora o innovadora . Cada tipus de xarxa té els seus punts forts i febles, la xarxa Tipus A que és més conservadora quasi no crea soroll en la imatge però no obté grans resultats. Mentrestant la xarxa Tipus B crea més error però també crea imatges més similars al esperat.

8.2 Treball futur

La manera per poder intentar millorar el treball fet seria intentar **buscar una nova base de dades** i intentar entrenar els models presentats en aquest treball amb aquella nova base de dades.

També crec que seria molt interessant en cas d'obtenir uns bons resultats, **crear una aplicació** que pogués fer us d'aquest models. D'aquesta manera poder distribuir el model si es necessàri en algun àmbit que ho necessiti sobretot en hospitals i clíniques que tracten amb pacients de CoVid-19.

Es podria intentar **aplicar una segmentació**, tal com s'ha explicat durant l'introducció de la U-net, a les imatges per reconèixer quines àrees del pulmó del pacient estan afectades i quin tipus d'afectació tenen. El qual també implicaria la necessitat de fer les màscares[38] d'entrenament.

Es podria intentar **millorar els resultats de la CycleGAN**, provant provar noves combinacions de paràmetres i fer les xarxes més profundes per que així puguin detectar més paràmetres i l'error es redueixi. També seria idoni buscar una base de dades en la qual les imatges radiogràfiques estiguessin etiquetades amb la data en la qual es van generar per així poder entrenar una xarxa amb una predicció més exacte.

[38]Imatges segmentades amb patxos de diversos colors, on cada color representa una zona diferent

References

[1] Genevieve B. Orr Yann LeCun Leon Bottou and Klaus-Robert Müller, eds. *"Efficient BackProp"*. 1998.

[2] David G.Lowe. "Method and apparatus for identifying scale invariant features in an image and use of same for locating an object in an image". Pat. US6711293B1. Mar. 8, 1999.

[3] BIMCV. *BIMCV-COVID19, Datasets related to COVID19's pathology course*. URL: https://arxiv.org/abs/2006.01174.

[4] davidtvs. *pytorch-lr.-finder*. URL: https://github.com/davidtvs/pytorch-lr-finder.

[5] Centers for Disease Control and Prevention. *SARS-CoV-2 Variant Classifications and Definitions*. URL: https://www.cdc.gov/coronavirus/2019-ncov/variants/variant-info.html.

[6] Vincent Dumoulin and Francesco Visin. "A guide to convolution arithmetic for deep learning". Guide. MILA, Université de Montréal,AIRLab, Politecnico di Milano.

[7] A. Díez Tascónb L. Ibáñez Sanza S. Ossaba Vélezb S. Borruel Nacentaa E. Martínez Chamorroa. *Diagnóstico radiológico del paciente con COVID-19*. URL: https://www.elsevier.es/es-revista-radiologia-119-avance-resumen-diagnostico-radiologico-del-paciente-con-S003383382030165X.

[8] Matthew P. Lungren Michelle Hershman Leonid Roshkovan Errol Colak Bradley J. Erickson George Shih Anouk Stein Jayashree Kalpathy-Cramer Jody Shen Mona Hafez Susan John Prabhakar Rajiah Brian P. Pogatchnik John Mongan Emre Altinmakas Erik R. Ranschaert Felipe C. Kitamura Laurens Topff Linda Moy Jeffrey P. Kanne Carol C. Wu Emily B. Tsai* Scott Simpson*. *The RSNA International COVID-19 Open Radiology Database (RICORD)*. URL: https://pubs.rsna.org/doi/full/10.1148/radiol.2021203957.

[9] HM Hospitales. *COVID DATA SAVE LIVES*. URL: https://www.hmhospitales.com/prensa/notas-de-prensa/comunicado-covid-data-save-lives.

[10] IBM. *Deep Learning*. URL: https://www.ibm.com/cloud/learn/deep-learning.

[11] IBM. *El modelo de redes neuronales*. Spanish. URL: https://www.ibm.com/docs/es/spss-modeler/SaaS?topic=networks-neural-model.

[12] IBM. *Supervised Learning*. URL: https://www.ibm.com/cloud/learn/supervised-learning#:~:text=Supervised%20learning%2C%20also%20known%20as,data%20or%20predict%20outcomes%20accurately..

[13] Rachel Thomas Jeremy Howards. *Fast.ai*. URL: https://www.fast.ai/about/.

[14] Arun K. *UNderstanding Learning Rate Machine Learning*. URL: https://www.mygreatlearning.com/blog/understanding-learning-rate-in-machine-learning/.

[15] Morton Kuo (Yu-Cheng Kuo). *ML03: PyTorch vs. TensorFlow*. URL: https://medium.com/analytics-vidhya/ml03-9de2f0dbd62d.

References

[16] Pytrick L. *PCA| K-means Clustering | |Unsupervised Learning Algorithms*. URL: https://towardsdatascience.com/into-to-pca-k-means-clustering-unsupervised-learning-algorithms-5cc5acea274d.

[17] Iasonas Kokkinos Kevin Murphy Alan L. Yuille Liang-Chieh Chen George Papandreou. *DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs*. URL: https://arxiv.org/abs/1606.00915v2.

[18] Sambit Mahapatra. *Why Deep Learning over Traditional Machine Learning?* URL: https://towardsdatascience.com/why-deep-learning-is-needed-over-traditional-machine-learning-1b6a99177063.

[19] John McCarthy. "What is artificial Intelligence?" In: *Computer Science Department Stanford University* ().

[20] Yali NIe. *A Multi-stage Convolution Machine with Scaling and Dilation for Human Pose Estimation*. URL: https://www.researchgate.net/publication/326531654_A_Multi-stage_Convolution_Machine_with_Scaling_and_Dilation_for_Human_Pose_Estimation_salam_jase_chujeong-eul_wihan_seukeilling_mich_hwagjang_giban_dadan_konbollusyeon_meosin.

[21] OpenCV. *Fourier Transform*. URL: https://docs.opencv.org/master/de/dbc/tutorial_py_fourier_transform.html.

[22] Opengenus. *Fully Connected Layer: The brute force layer of a Machine Learning model*. URL: https://iq.opengenus.org/fully-connected-layer/.

[23] World intellectual property organitzation. *Artificial Intelligence*. URL: https://www.wipo.int/edocs/pubdocs/en/wipo_pub_1055.pdf.

[24] David Rotman. *TR10: IA capaz de descubrir molèculas*. URL: https://www.technologyreview.es/s/11970/tr10-ia-capaz-de-descubrir-moleculas.

[25] Sumit Saha. *A comprehensive guide to convolutional naural networks*. URL: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[26] Aljosa Smolic Sebastian Lutz. *AlphaGAN: Generative adversarial networks for natural image matting*. URL: https://www.researchgate.net/figure/The-generator-is-an-encoder-decoder-network-with-skip-connections_fig1_326632662.

[27] Gurav Singhal. *Transfer Learning with ResNet in PyTorch*. URL: https://www.pluralsight.com/guides/introduction-to-resnet.

[28] UPC. *Studying the characteristics of deep CNN neurons*. URL: https://www.google.com/url?sa=i&url=https%3A%2F%2Fupcommons.upc.edu%2Fbitstream%2Fhandle%2F2117%2F168420%2F144234.pdf&psig=AOvVaw0zFiaUrpw5xyjMFMxqWG8l&ust=1628711108454000&source=images&cd=vfe&ved=0CAwQjhxqFwoTCLiDyv2bp_ICFQAAAAAdAAAAABAP.

[29] Jun-Yan Zhu. *CycleGAN and pix2pix in PyTorch*. URL: https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix/.

9 Instal·lació del programari

En aquesta secció es mostrarà com es crea un "Virtual Environment" i com s'instal·la el programari utilitzat en aquest projecte en un sistema basat en Linux.

9.1 Venv

Per a crear un "virtual environment" primer es necessita una versió de Python dins del servidor. Seguidament posem les següents comandes al Terminal:

- cd ∼
- mkdir "Nom virtual environment"
- cd "Nom virtual environment"
- python -m venv

Un cop instal·lat per activar-lo s'ha d'utilitzar les comandes:

- cd ∼
- source "Nom virtual environment"\bin\activate

9.2 Python i pip3

En aquest apartat es veuran les comandes necessàries per instal·lar Python en Ubuntu:

- sudo apt update
- sudo apt install python3.6
- sudo apt install python3-pip

9.3 Instalació llibreria genèrica

Per a instal·lar una llibreria genèrica el que es fa és executar la comanda:

- sudo apt install "Nom llibreria"