GAT, GOS, OCELL O CONILL? LA CLASSIFICACIÓ A PARTIR DE LES XARXES NEURONALS

PSEUDÒNIM AUTOR/A: Científica

CURS: 2n de Batxillerat

ANY: 2019/2020

DEPARTAMENT: Tecnologia

ABTRACT:

Nowadays, artificial intelligence is constantly evolving, and its use is increasing in our daily lives. The objective of this project is to create a neural network capable of distinguish between four different types of animals. To do this, it will be trained through a set of images which contain photos of these animals. The hypothesis of the project is to achieve a high performance of classification effectiveness. In addition, a program has been created in order to put the name of the animal in each image that it has classified. In this way it shows us what kind of animal we are seeing in the image. The project and especially the practical part, has been carried out with the help of the Computer Vision Center in the “Universitat Autònoma de Barcelona”. The network has been trained in the best possible way with the objective of achieving the best results of it. Creating a network from scratch is not easy and it takes time and more data to get an accurate performance. Thanks to this project I have evolved in the knowledge of neural networks and programming. Two topics that I had never done before and that will be a great help to me in my academic and professional future.

RESUMEN:

La inteligencia artificial está evolucionando constantemente y su uso cada vez es mayor. El objetivo de este proyecto es crear una red neuronal capaz de distinguir entre cuatro tipos de animales diferentes. Para ello, se entrenará a través de un conjunto de imágenes que contengan fotos de estos animales. La hipótesis del trabajo consiste en conseguir un rendimiento alto de efectividad en la clasificación. Además, se ha creado un programa con la finalidad de poner el nombre del animal en cada imagen que haya clasificado. De esta manera nos enseña que tipo de animal estamos viendo en la imagen. El proyecto y especialmente la parte práctica, se ha realizado con la ayuda del Centro de Visión por Computador de la Universidad Autónoma de Barcelona. La red se ha entrenado de la mejor manera posible con la intención de obtener los mejores resultados. Crear una red desde cero no es fácil y se necesita tiempo y más cantidad de datos para conseguir buen rendimiento. Gracias a este proyecto he evolucionado en el conocimiento de redes neuronales y programación. Dos temas que nunca había tratado antes y que me serán de gran ayuda en mi futuro académico y profesional.

Índex

1. Introducció

1.1 Qui sóc

Sóc estudiant de 2n de Batxillerat en la modalitat tecnològica i tinc 17 anys.

Un cop acabi el batxillerat m'agradaria estudiar enginyeria física, enginyeria en tecnologies industrials o enginyeria informàtica. És per això que gràcies a aquest treball de recerca he pogut experimentar amb tot el tema de la programació i poder veure si en el futur m'agradaria treballar en aquest àmbit.

1.2 Tria del tema

En el moment que havíem de triar un tema pel nostre treball de recerca, vaig pensar que el podria fer sobre Criptografia. Desxifrar missatges secrets i conèixer diferents mètodes per fer-ho. Però l'institut em van donar l'oportunitat de fer el treball al Centre de Visió per Computador (CVC) a la Universitat Autònoma de Barcelona amb l'ajuda d'un tutor. Acceptar aquesta meravellosa oportunitat, suposava canviar el tema i escollir-ne un relacionat amb les xarxes neuronals. Finalment, vaig prendre la decisió de no fer-lo sobre criptografia i introduir-me al món de la programació. Va ser una molt bona oportunitat, ja que vaig poder tenir accés al servidor GPU del CVC i realitzar els meus propis experiments. Poder anar a la universitat m'ha facilitat el desenvolupament del projecte a través de la pròpia experimentació, ja que no hagués tingut accés a aquest servidor des de casa, ni a tant coneixement com el que he pogut obtenir al CVC.

1.3 En què consisteix el treball?

Aquest treball de recerca consisteix a crear un sistema de classificació d'imatges. L'objectiu és que la xarxa aprengui a distingir entre classes d'imatges analitzant-les, i al final ser capaç de diferenciar entre una classe i l'altra amb un percentatge alt de fiabilitat.

El sistema permet introduir diversos tipus de classes. Aquest projecte s'ha basat en la classificació d'imatges d'animals i entre aquests he escollit quatre tipus: gos, gat, conill i ocell. Se li introdueix un gran nombre d'imatges de diferents races, mides i colors de cada tipus d'animal perquè tingui un ampli coneixement de les característiques d'aquests. Després de diversos entrenaments ha de ser capaç de classificar correctament la nova imatge que se li ensenyi. I per últim, he afegit que un cop distingida la classe, hi escrigui el nom d'aquesta a sobre.

A més, durant el procés es compara aquest xarxa anomenada Anna_net amb una altra, que a diferència d'ella, no parteix des de zero. Aquesta última és anomenada Vgg16. Realitzar aquest procediment permetrà observar la gran diferència que s'experimenta entre un model entrenat prèviament i un que parteix de zero.

1.3.1 Ampliació

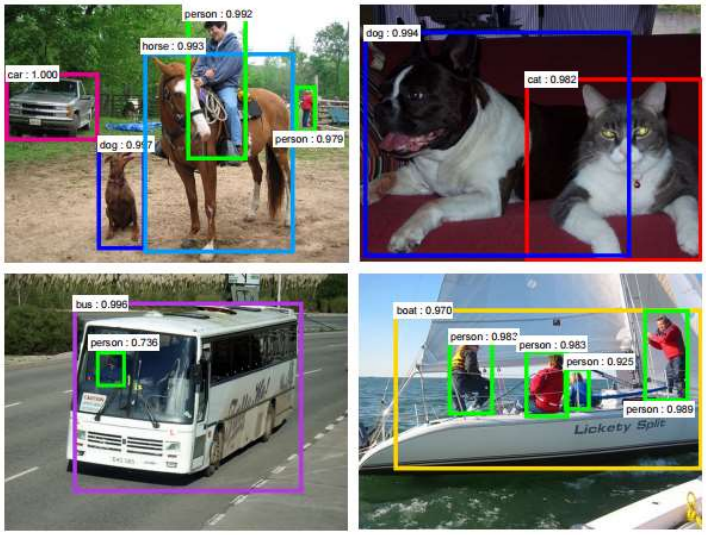
Aquest treball està enfocat a la classificació, però a part de la classificació hi ha altres mètodes de xarxes neuronals com són la segmentació i la detecció. Aquestes dues són més complexes perquè no solament analitzen un objecte sinó que analitzen un paisatge distingint cada objecte dins d'aquesta.

La segmentació pinta cada element de la imatge del color que correspongui. Per exemple, els vianants de color vermell, la vorera de color rosa, etc.

Il·lustració 1.1 Exemple de segmentació

Font: Nanonets

En canvi, la detecció col·loca una capseta al voltant de cada objecte, també representada amb un color diferent per a cada classe. Aquest n'és un exemple:



Il·lustració 1.2 Exemple de detecció

Font: Sigmoidal

Cap d'aquests mètodes són senzills i demanen més dedicació i coneixement. Donat el poc temps que tenim per fer el treball, era impossible endinsar-me en aquests camps. De cara el futur però, són dos possibles camps els quals podria estudiar i investigar, ja que la base ja la tindria integrada en els meus coneixements.

1.4 Hipòtesi

La hipòtesi que es planteja en aquest treball és la següent:

Una xarxa neuronal entrenada des de zero és capaç d'obtenir un alt percentatge de fiabilitat en la classificació.

1.5 Metodologia

La teoria dels conceptes de Machine Learning i sobretot Deep Learning és bastant complicada. Per documentar-me vaig estar realitzant diferents cursos per internet durant força temps. Quan vaig tenir una visió general de tot, el tutor de la UAB, va acabar d'explicar-m'ho amb més detall perquè quan comencés la pràctica, tingués més amplis coneixements del que estava fent.

Des d'un inici vaig organitzar quina metodologia empraria per crear el treball. Com que era nova en aquest món de la programació en Python i en el tema de xarxes neuronals, vaig decidir dedicar-me tot el mes de juliol a anar al CVC de la UAB per aprendre la teoria necessària i dur a terme la part pràctica.

A finals del mes de juliol i durant el mes d'agost, vaig posar-me l'objectiu de redactar la memòria. I per últim, el mes de setembre i principis d'octubre, acabar-la d'escriure i corregir petits errors.

1.6 Agraïments

Agrair principalment al meu tutor de la Universitat Autònoma de Barcelona, per tota l'ajuda que m'ha donat en aquest tema, per com s'ha implicat en explicar-me la complicada teoria que és necessària saber i per tota l'estona que ha estat al meu costat solucionant-me qualsevol dubte que tingués.

Seguidament, donar les gràcies al meu tutor de l'institut que sempre m'ha intentat ajudar amb tot el que ha pogut, tant en l'objectiu del treball com en la formalitat de l'escriptura. A més, agrair a la meva segona tutora, per ajudar-me en la part de redacció.

I per últim donar les gràcies a la meva família, sobretot a la meva mare per ajudar-me els últims dies a millorar l'expressió escrita i per rectificar-me tot allò complicat d'entendre des d'un punt genèric i inexpert del tema. A més, agrair als meus amics i a la gent del meu voltant per ajudar-me i donar-me totes les forces per continuar endavant amb aquest treball.

2. Fonaments teòrics

2.1 Introducció a la Visió per computador

La visió per computador és una disciplina científica i s'aplica a molts camps, com per exemple, a la intel·ligència artificial. Utilitza mètodes per adquirir, analitzar, processar i comprendre les imatges del món real amb la finalitat de produir informació numèrica o simbòlica perquè puguin ser tractats per un ordinador. Les principals aplicacions on s'utilitza la visió per computador són en la classificació i detecció d'objectes, reconeixement facial i accions humanes, la reconstrucció d'una escena i la restauració d'imatges. Una de les aplicacions més importants són els cotxes autònoms, que utilitzen sensors, com per exemple, càmeres per reconèixer elements de la carretera. A més, la visió per computador s'utilitza també en la conducció per sistemes d'assistència a la carretera que milloren la seguretat al volant.

Per aconseguir que el sistema realitzi aquestes accions, s'ha d'haver entrenat prèviament donant-li diferents imatges perquè les pugui anar aprenent i diferenciar-les unes de les altres. Per realitzar això, un dels recursos que utilitza és l'anomenat Machine Learning. El Machine Learning utilitza algoritmes per processar dades, aprendre d'elles i finalment fer una predicció. Aquest és l'àmbit general on trobem un concepte anomenat Deep Learning que consisteix en algoritmes capaços d'aprendre sense intervenció humana prèvia, traient ells mateixos les conclusions. El funcionament d'aquests algoritmes intenta imitar el funcionament del cervell humà. En aquest últim àmbit hi formen part les xarxes neuronals, que són un paradigma d'aprenentatge i processament automàtic inspirat en el funcionament del sistema nerviós.

2.2 Breu història sobre les xarxes neuronals

El fet d'arribar a tota la tecnologia d'avui en dia no ha estat fàcil. Les xarxes neuronals, com qualsevol altre camp tecnològic, tenen una història que és important saber.

Tot va començar el 1936 quan Alan Turing va ser el primer a pensar que podria haver-hi una gran semblança entre el món de la computació i el cervell humà. Tretze anys més tard, Donald Hebb va establir el concepte d'aprenentatge. Va intentar trobar una semblança entre l'aprenentatge i l'activitat nerviosa descobrint les bases de la teoria de les xarxes neuronals que s'estudien actualment.

El 1950 Karl Lashley va dur a terme una sèrie de proves que van demostrar que la informació no és emmagatzemada de forma centralitzada, sinó que aquesta es distribueix en diverses regions del cervell. Sis anys després es va celebrar la Conferència de Dartmouth que va suposar el naixement de la intel·ligència artificial.

El 1960 Bernard Windroff i Marcian Hoff van desenvolupar el model Adaline (ADAptative LINear Elements) que és la primera aplicació d'una xarxa neuronal artificial a la realitat i que s'ha implementat durant vàries dècades.

El 1974 Paul Werbos va desenvolupar la idea bàsica de l'algoritme de propagació cap endarrere (BackPropagation). Tres anys més tard, Stephen Grossberg anuncia la Teoria de la Ressonància Adaptada que és una arquitectura totalment diferent de les existents fins aquell moment. Aquesta permet simular la memòria a llarg i curt termini.

I per últim, John Hopfield el 1985 aconsegueix fer renéixer les xarxes neuronals gràcies a la presentació del seu llibre "Computació neuronal de decisions en problemes d'optimització".

Les xarxes neuronals tot i que es van introduir fa bastant temps, cap als anys 40 i 50, no s'han utilitzat fins fa relativament poc a causa de la gran quantitat de memòria d'ordinador que consumeixen per entrenar-se, executar-se i aconseguir bons resultats. En els últims anys s'han obtingut grans avenços gràcies a l'eficiència dels ordinadors i a l'ús de GPUs per aquest tipus de computacions.

La primera aplicació amb èxit i àmpliament coneguda de les xarxes neuronals convolucionals va ser LeNet-5, caracteritzada per la seva senzillesa i fàcil d'entendre, descrita per Yann LeCun en el seu document de 1998 anomenat "Gradient-based learning applied to document recognition". Aquesta consistia al reconeixement de caràcters escrits a mà i impresos a màquina a la dècada del 1990.

El treball considerat com un interès renovat a les xarxes neuronals i el principi de domini d'aprenentatge profund en moltes aplicacions de visió artificial va ser el document de 2012 d'Alex Krishevsky titulat "ImageNet Classification with Deep Convolutional Neural Networks". En l'article descriu un model anomenat AlexNet el qual va aconseguir un major rendiment de les xarxes neuronals comparat amb l'anterior.

2.3 Estat de l'art

En aquest apartat s'exposa quines són les investigacions més recents per a classificació d'imatges. Un important treball que va buscar el millor disseny d'arquitectura per xarxes convolucionals i desenvolupar models molt més profunds i de millor rendiment en el procés, va ser el document de 2014 titulat "Very Deep Convolutional Networks for Large-Scale Image Recognition" per Karen Simonyan i Andrew Zisserman. La seva arquitectura es coneix generalment com Vgg.

Una última innovació arquitectònica en xarxes neuronals important va ser proposada per Kaiming He en el seu document anomenat “Deep Residual Learning for Image Recognition”. En el treball l’autor va presentar un model molt minuciós anomenat ResNet.

Totes aquestes xarxes de classificació mencionades s’explicaran amb més profunditat en un altre apartat.

2.4 Com funciona una xarxa neuronal?

El funcionament d’una xarxa neuronal no és senzill perquè està compost per diversos elements. Analitzem un per un quins són aquests elements.

2.4.1 El Dataset

Per saber com funciona una xarxa neuronal, primer s’ha d’entendre què significa el Dataset. El Dataset és un conjunt d’imatges, números, paraules, dades, etc. En aquet cas es tracta d’un conjunt d’imatges digitals. Aquestes imatges tenen un format anomenat RGB el qual està compost per tres canals com indica el nom (Red, Green i Blue). Les imatges també estan definides per una resolució (ample x altura) que defineix la quantitat de píxels que té una imatge, on cada píxel té un valor entre 0-255 per cadascun dels canals RGB. El conjunt d’imatges es divideix en tres parts. Una part és la d’entrenament, una altra la de validació i l’última la de test.

La part d’entrenament és el conjunt d’imatges que la xarxa aprèn gràcies al fet que té el Ground Truth (GT). El Ground Truth és la solució de quin tipus de classe correspon. Per exemple, si estem tractant una imatge d’un gos, li donem la solució que pertany a la classe gos.

La part de validació, és aquella en que la xarxa valida les imatges coneixent també el GT. És a dir, es fa un examen a sí mateixa i es corregeix amb la solució al davant.

I per últim, la part de test, són les imatges amb que li fem com una mena d'examen a la xarxa, aquest cop sense el GT, per veure si realment ha après.

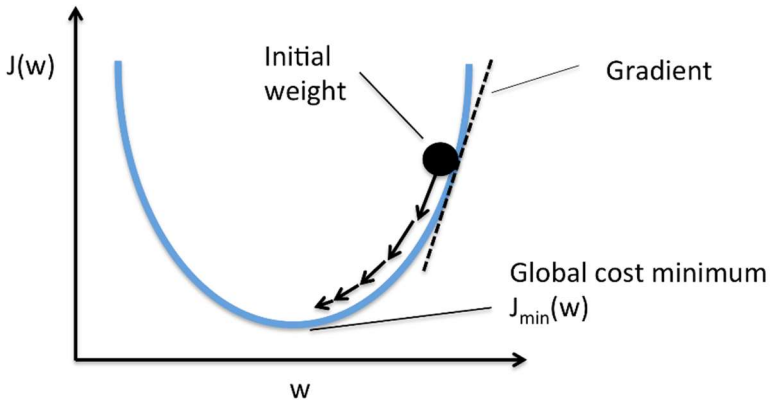
2.4.2 Data Augmentation

El Data Augmentation és un recurs que s'utilitza per augmentar el nombre d'imatges del dataset a partir de fer-hi modificacions perquè aquestes siguin diferents de les originals. Per exemple, afegir diferents tipus de soroll fent la imatge borrosa (Random dist), realitzar zooms en algunes parts, donar la volta a les imatges (Hflips), entre d'altres. Aquestes operacions ens faciliten no haver de buscar noves imatges per afegir-hi, sinó que amb les mateixes que tenim en podem crear de noves i així no hem d'ampliar el volum.

2.4.3 Com s'entrena?

Quan s'entrena la xarxa, l'objectiu és que determini a quina classe pertany la imatge i això ho farà amb un valor numèric. Per entrenar-la i que aprengui de sí mateixa establim la Loss Function. Quan s'insereix una imatge, la xarxa diu a quina classe pertany, inicialment ho dirà erròniament i estarà parametritzat per un percentatge. Per exemple, 10% gos, 50% conill i 40% gat, i en canvi el GT dirà que és 100% conill. La diferència entre el que classifica el sistema i el GT es parametritza i s'obté un nombre anomenat Loss que s'aconsegueix a partir de la Loss Function. El concepte de Loss ens indica que com més a prop de zero més a prop estem de la solució correcta. Per això, busquem minimitzar la Loss i com

a conjunt, estem davant un problema de minimització. Per buscar una solució, utilitzem un algoritme per trobar el mínim en una funció, anomenat descens de gradient. Aquest seria el representat a la següent imatge:



Il·lustració 2.1 Gràfic descens de gradient

Font: Hackernoon

Com podeu veure en la il·lustració, la idea és anar baixant fins a trobar el mínim error que és a on es troba la solució òptima al nostre problema. Per fer-ho, s'utilitza un paràmetre, anomenat learning rate. Aquest és un valor més o menys entre 1^{-3} i 1^{-7}. Com més alt sigui el número, el descens per trobar el mínim serà més gran i per tant més difícil de concretar. Però com més baix el número, més detalladament intenta arribar a la solució i per tant més llarg el camí per arribar-hi. L'important és buscar el nombre adient que no sigui ni molt alt ni molt baix.

Hi ha tècniques que ens ajuden a trobar amb més facilitat el learning rate. Aquestes són el SGD, el RMSprop i l'Adam. El SGD (Stochastic Gradient Descent / Descens estocàstic de gradient), tracta de trobar màxims i mínims per iteració. L'Adam és una extensió del SGD, que ho combina amb el RMSprop i tenen en compte dos moments (el moment és utilitzat en el SGD) per ajustar més precisament el learning rate i respondre millor que els altres dos mètodes. El SGD és el bàsic i els altres dos són més complexos i tenen millor rendiment.

És molt important saber que la xarxa no observa una imatge i directament diu la classe. La imatge quan està digitalitzada (dins un PC amb un format jpg, png…) no deixa de ser un tipus de dada estructurada que permet emmagatzemar un conjunt de dades del mateix tipus i relacionades (array). La xarxa rep aquestes imatges en forma de matriu amb els valors corresponents en cada píxel. El valor però, el normalitzem per facilitar-nos els càlculs, és a dir que ho transformem en valors de 0-1, sent 1 el 255. Si no normalitzéssim els valors, a l'hora de fer càlculs el valor incrementaria molt i la xarxa no donaria l'abast.

El que vertaderament fa la xarxa és convertir la imatge en números. És a dir, que quan diu a quin tipus de classe pertany, també ho diu amb números i per tant som nosaltres que ho hem de convertir a una paraula per trobar-li el sentit.

A més a més, durant l'entrenament hi ha un concepte anomenat Back Propagation. La seva tasca és rectificar els seus propis errors, és a dir, quan el sistema identifica un objecte i aquest no és correcte, torna enrere per rectificar l'error comès i acabar identificant de forma òptima l'objecte.

2.4.4 Validant entrenament

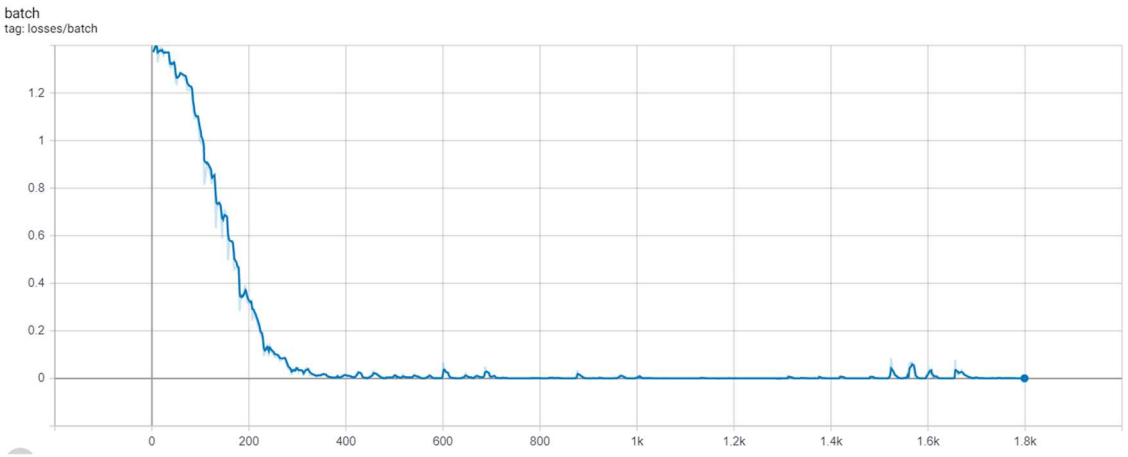
Cada cop que la xarxa s'entrena passant un cop per tot el conjunt d'imatges de l'entrenament s'anomena epoch. Depenent la quantitat d'imatges que tinguem i la complexitat de la xarxa, completar un epoch pot consumir molt de temps. Si tenim un gran volum de dades, és molt probable que amb pocs epochs obtinguem resultats apropiats i tanmateix si el temps ens ho permet, podem augmentar el nombre. Cada cop que acaba un epoch, el sistema informa dels resultats obtinguts.

Per donar aquesta informació utilitza mètriques que mesuren com de precís ha estat el sistema. Les utilitza fent una comparativa respecte a quines imatges ha

agafat i les ha classificat correctament, i respecte a quines ha agafat però no les ha classificat correctament. Les primeres s'anomenen True Positive i les segones False Negative. Els noms de les mètriques utilitzades són Recall, Precision, f1 score i el seguiment de Loss. Aquestes mètriques es comentaran més endavant del treball.

2.4.5 Com sabem si ha après?

Quan es realitza l'entrenament, cal saber com ha anat i si ha estat capaç de millorar o no envers l'anterior vegada. La mètrica més important és la loss. Aquesta informa de com baixa la loss (resultat d'aplicar la funció) respecte el temps. Si la loss va baixant en cada epoch, és símptoma que està aprenent i adaptant-se al conjunt d'entrenament. I com més a prop a zero, més capaç de classificar correctament aquest conjunt.

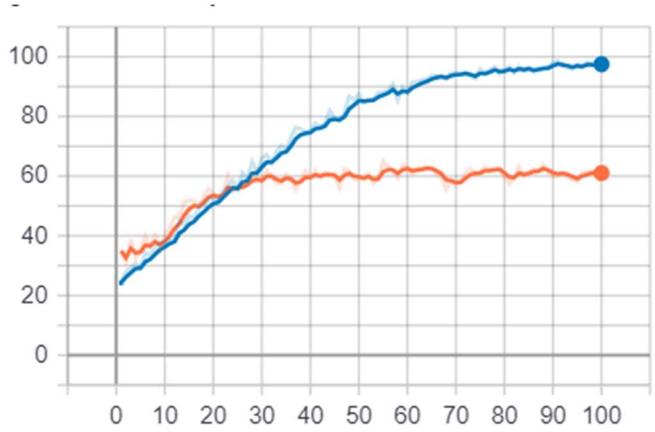


Il·lustració 2.2 Gràfica los

Font: captura Tensorboard

Això però, pot tenir una conseqüència negativa anomenada Overfitting. Per això cal interpretar les gràfiques. Si les línies d'entrenament i de validació estan molt separades entre elles, significa que hi ha hagut un Overfitting. L'Overfitting (sobre

entrenament) significa que s'ha adaptat massa al conjunt d'imatges utilitzades en l'entrenament. Això pot ser degut a que són molt simples (imatges fàcils) o perquè ha entrenat durant molta estona (alt nombre d'epochs). En aquest cas, el sistema s'acostuma amb les imatges d'entrenament i un cop utilitza les de validació el resultat no és tan òptim. Per solucionar aquest problema es pot utilitzar una tècnica anomenada Early Stopping (para quan vegis que no entrenes més), mitjançant la qual es pot aturar l'entrenament quan es vegi que durant la validació les mètriques no milloren.



Il·lustració 2.3 Gràfic mètrica amb Overfitting

Font: captura Tensorboard

Un cop explicat l'Overfitting, s'ha de tenir en compte que també existeix l'Underfitting (baix entrenament). Aquest últim significa que no ha entrenat prou i que per tant no tindrà suficients coneixements per fer la validació o el test. Per millorar-lo, senzillament cal que entreni més, és a dir, afegir-li epochs.

2.4.6 Com es testeja?

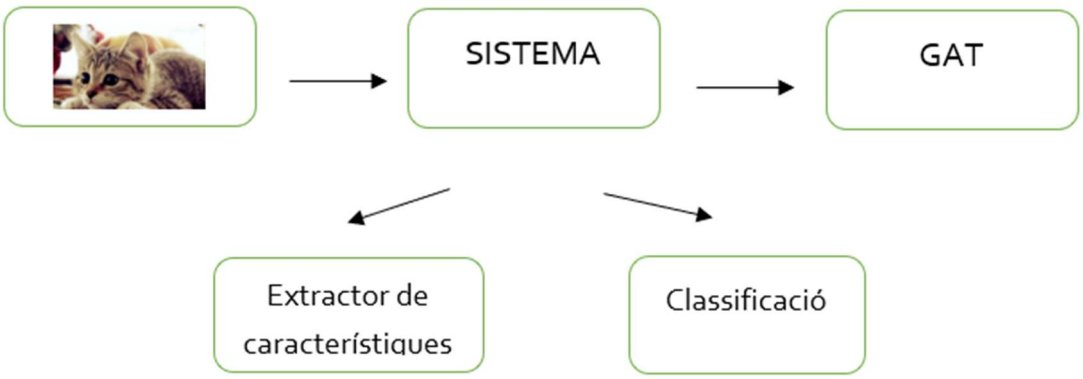
Tant bon punt ha finalitzat l'entrenament amb les validacions corresponents es fa el test. Quan testeja, com ja s'ha comentat anteriorment, no té el GT, per tant l'important és que aconsegueixi endevinar correctament les imatges. Amb el test es pot dir que disposem d'un model entrenat per utilitzar-se en la vida real. El conjunt de test simula aquesta realitat amb imatges que no s'han vist anteriorment. En aquest punt es poden crear aplicacions per diferents dispositius electrònics com ordinadors, tablets o càmeres que a partir d'un sistema de captació d'imatges podrien utilitzar-se per testejar el nostre model final. Normalment és l'humà qui s'encarrega de fer, mitjançant un anàlisi qualitatiu, l'avaluació final del nostre model.

2.5 Problema de classificació

En l'apartat anterior s'ha explicat els conceptes més teòrics, en aquest es tracta d'explicar com funciona a la pràctica. En primer lloc, tot el procediment que es fa perquè el sistema de Deep Learning aprengui, és a través de xarxes neuronals convolucionals. Les xarxes de classificació es divideixen en dues parts. A continuació s'explica quina és la seva estructura.

2.5.1 Arquitectura de la xarxa

El procediment que faria la xarxa per acabar identificant l'objecte, seria el següent:

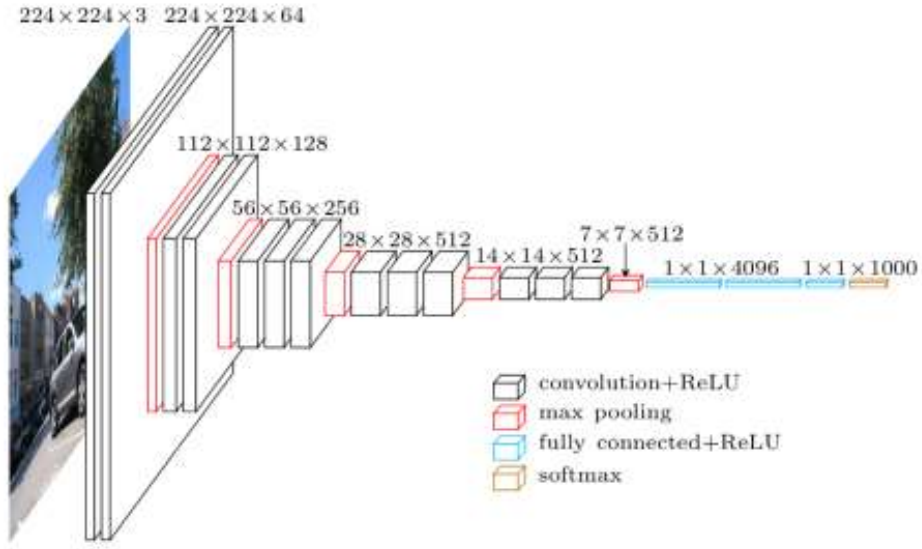


Esquema 2.1 Esquema funcionament xarxa

Font: Elaboració pròpia

El sistema per classificar objectes està creat a partir de l'extractor de característiques i la classificació. L'extractor de característiques és el que analitza profundament la imatge a partir de diferents passos que s'explicaran a continuació extraient les característiques necessàries d'aquesta. I la part de classificació és la responsable de treure les conclusions de quin animal es tracta.

Aquest seria un exemple amb l'extractor de característiques:



Il·lustració 2.4 Funcionament Vgg

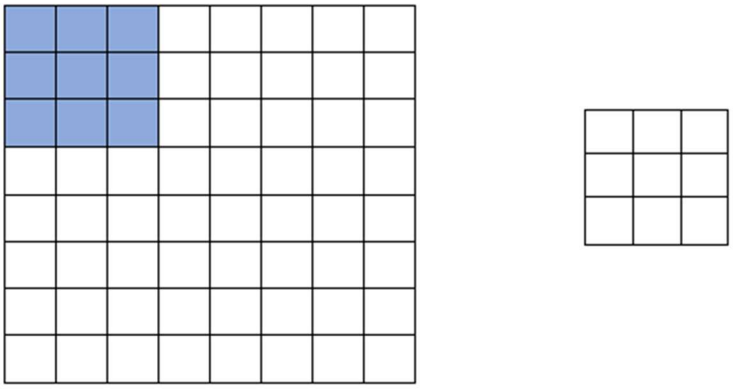
Font: Pàgina web En mi local funciona

Es pot veure com la imatge anirà passant per diferents capes. Aquestes capes també anomenades Layers són les següents:

1. Convolucions
2. Pooling
3. Fully Connected
4. Softmax
5. Dropout
6. Batch Normalization

1.CONVOLUCIONS:

Primer de tot s'introdueix la imatge amb el seu vector corresponent. Seguidament, la imatge passa per dues convolucions on el que abans ens indicava un 3, ara s'ha convertit en un 64 que és el nombre de neurones que hi ha. Una convolució és el següent:



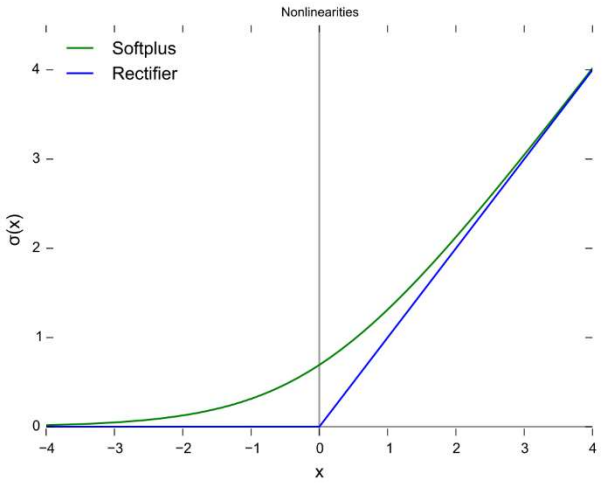
Esquema 2.2 Representació convolucions

Font: Elaboració pròpia

Les convolucions són a dues dimensions ja que s'opera amb imatges (matrius). Per la convolució es posa un filtre de 3x3 píxels on cada un d'aquests té un valor diferent. A cada convolució aquest filtre va passant per tota la imatge obtenint les característiques necessàries. Com més convolucions s'apliquin, més augmentarà la grandària de la quadrícula.

Una part important d'aquest procés, és que quan es realitza aquest pas s'ha d'afegir un Padding (marge). El Padding crea un marge on els seus valors seran tots zero i d'aquesta manera s'evita que el resultat perdi dimensions i es mantingui la dimensió de la imatge. L'operació que s'utilitza entre el filtre i la imatge és un producte on cada valor del filtre es multiplica amb el corresponent a la imatge.

Entre convolucions hi ha la ReLU (Il·lustració 7), que és una funció d'activació i determina quant important és la neurona a partir de la informació que ha rebut. A més, és responsable de treure els nombres negatius i deixar passar només els positius.



Il·lustració 2.5 Gràfic ReLU

Font: Viquipèdia

2.POOLING:

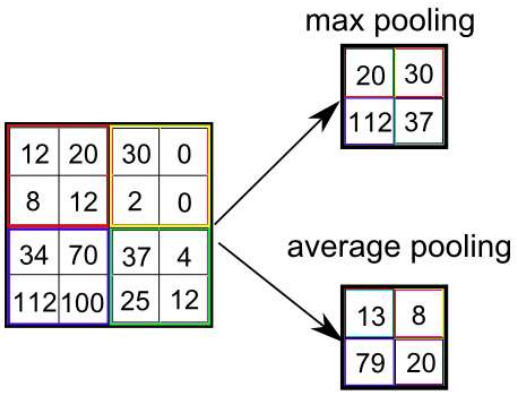
Passem a la següent fase que és el Pooling. El Pooling redueix la imatge en escales de potències de dos. En aquest cas s'ha dividit entre dos. L'explicació d'això és que quan es disminueix la mida de la imatge, al tornar fer convolucions, aquestes ara agafaran unes característiques més generals i no tan concretes. Les convolucions en què obtenim diferents tipus de característiques, s'anomenen convolucions de baix nivell, de mig nivell i d'alt nivell.

Les de baix nivell busquen característiques molt profundes i petites de la imatge, com per exemple detalls que només es poden apreciar amb una lupa.

Les de mig nivell busquen elements una mica més generals com podrien ser petites curvatures de l'objecte que compon la imatge.

I per últim, les d'alt nivell, que busquen la forma més general de l'objecte com seria el seu contorn.

Per fer aquest reducció que fa el Pooling, es poden utilitzar dos tipus. O bé mitjançant l'anomenada Max Pooling, que en la reducció agafa el número més gran que hi ha en els valors, o bé mitjançant l'anomenada Average Pooling, que fa la mitjana dels valors.

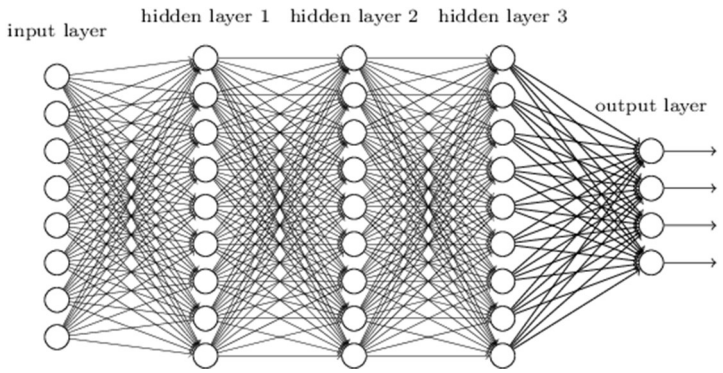


Il·lustració 2.6 Esquema de Max Pooling i Average Pooling

Font: Quora

3.FULLY CONNECTED:

Tornant a la il·lustració 6, ara correspon explicar el Fully Connected que es representaria així:



Il·lustració 2.7 Fully Connected

Font: Ferrnando Sancho Caparrini

En aquesta imatge es pot veure com les neurones entrants estan connectades amb les neurones sortints i a la vegada aquesta connexió les interrelaciona a totes elles per tenir característiques globals de la imatge. La fully connected pren tota la informació que s'ha anat utilitzant durant tot el procediment de la xarxa i l'ajunta per trobar un sentit. Per exemple, ha detectat pèl, quatre potes, bigoti i cua. Ho posa tot en context i acaba deduint que es tracta d'un gat. Aquest exemple es mostra perquè s'entengui el que fa el sistema, però realment aquest ho analitza d'una manera més abstracta i no tan simple.

4.SOFTMAX:

La següent capa s'anomena Softmax, responsable de traduir la probabilitat de ser de cada una de les classes definides a la xarxa. D'aquesta probabilitat escull la més gran per donar-nos a conèixer la classe a la que pertany aquella imatge.

5.DROPOUT:

El dropout és una Layer encarregada de desactivar algunes neurones de forma aleatòria, amb l'objectiu que la xarxa no s'acostumi a que unes neurones aprenguin més que les altres, sinó que al desactivar-ne algunes, totes acaben aprenent per igual. Per exemple, si una neurona aprèn una característica concreta, al desactivar-la, obligues a una altra neurona a aprendre el mateix que aquesta. És una bona tècnica per crear diferents arquitectures als entrenaments. També se l'anomena Congelació de pesos.

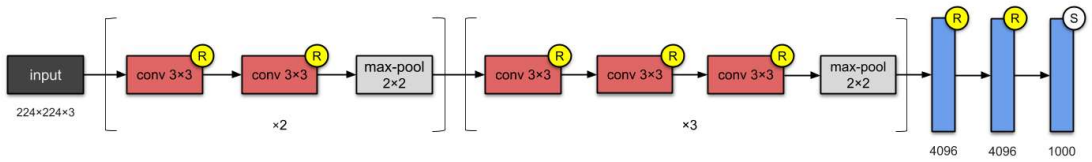
6.BATCH NORMALIZATION:

I per últim tenim el Batch normalization que normalitza les imatges. Per exemple, si tenim una imatge fosca i una clara, normalitza aquestes imatges per corregir aquests canvis i que el mateix objecte amb un color, es vegi el més semblant possible entre totes elles. Aquesta normalització és per Batch. El Batch és un conjunt d'imatges que entra a la xarxa (en aquest cas és de 40) i es fa aquesta operació sobre aquest conjunt. Al fer-ho, facilitem el Back propagation a trobar els errors comesos.

2.5.2 Xarxa de classificació

En la il·lustració 6 comentada anteriorment, es pot observar un tipus de xarxa de classificació. Aquest tipus s'anomena Vgg. Hem utilitzat aquesta com a exemple perquè és la més senzilla per entendre el procediment. Però a part de la Vgg, hi ha altres com l'AlexNet, la Resnet i la DenseNet les quals en fem una breu descripció a continuació.

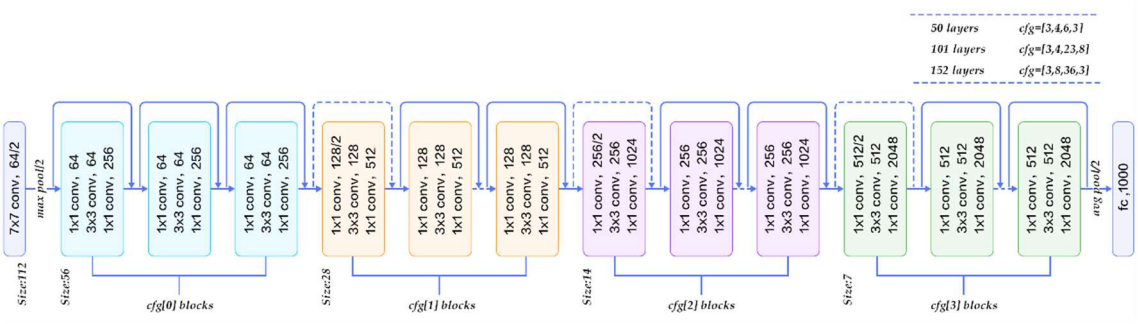
L'AlexNet conté 8 layers (capes). Les primeres 5 són convolucions, algunes seguides per un MaxPooling, i les últimes tres són fully-connected. Està composta per 60 milions de paràmetres. Utilitza la funció ReLU, Dropout i Data Augmentation.



Il·lustració 2.8 Funcionament AlexNet

Font: Towards Data Science

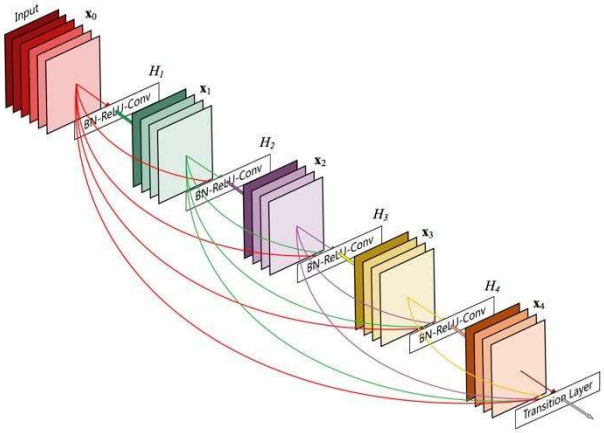
La ResNet es basa en augmentar el nombre de capes introduint una connexió residual (una capa d'identitat). Aquesta capa passa a la següent millorant el procés d'aprenentatge. La idea d'introduir aquestes capes residuals és per fer xarxes molt profundes de 101 o 150 layers (ResNet 50, ResNet 101 i ResNet 150) ja que sense aquestes connexions residuals no és viable fer-les aprendre perquè són massa grans.



Il·lustració 2.9 Funcionament ResNet

Font: Pàgina web En mi local funciona

I la DenseNet, bàsicament el que fa és connectar cada sortida de les capes amb totes les següents, que a diferència dels altres tipus, això no succeïa. La diferència amb la ResNet, és que aquesta última suma la sortida de la convolució amb la connexió residual, mentre que la DensNet es concatena amb la informació, conservant-la al final.



Il·lustració 2.10 Funcionament DenseNet

Font: Pàgina web En mi local funciona

2.5.3 Mètriques

Les mètriques són les que valoren l'experiment i informen si està evolucionant correctament.

Tant durant l'experiment com al final, atorga un tant per cent de les següents mètriques:

- **Loss** (pèrdua): Indica quina pèrdua ha tingut la el sistema. Perquè sigui una bona loss, hauria de ser de 0,01 aproximadament. Cal recordar que cap màquina és perfecte al 100%. Per calcular la pèrdua es realitza a través d'aquesta fórmula:

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log\big(p(yi)\big) + (1 - y_i) \cdot \log\,(1 - p(y_i))$$

- **Accuracy** (acurat): La resposta que s'obté d'aquesta mètrica són les prediccions correctes més aquelles que s'ha deixat perquè no les ha pogut detectar, dividides entre la totalitat. Aquesta totalitat està formada per les calculades correctament (True Positive), les correctes que no ha calculat (False Negative), les que ha calculat errròniament (False Positive) i les que no ha calculat perquè no s'havien de fer (True Negative).

- **Precision** (precisió): Aquesta mètrica es calcula dividint les prediccions correctes entre la suma de les prediccions correctes (True Positive) i les incorrectes (False Positive).

- **Recall** (memòria): El Recall és el resultat de dividir les prediccions correctes entre aquestes mateixes més les prediccions correctes que no ha detectat.

- **F1score** (nota): I per últim, aquesta mètrica es calcula dividint el producte entre la suma. És com una mena de mitjana entre la Precision i el Recall.

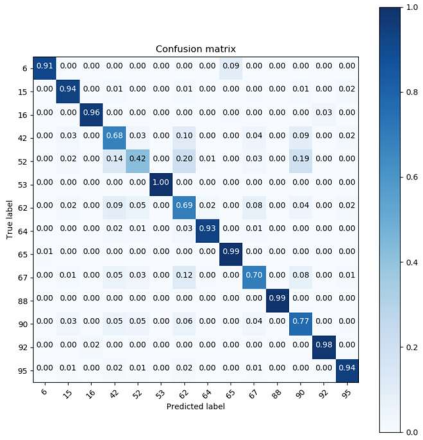
A part d'aquestes cinc mètriques mencionades, existeix una altra molt important anomenada matriu de confusió. La matriu quadrada de confusió el que pretén aconseguir és que quedi una línia clara en diagonal. Que totes les imatges que ha rebut, per exemple de gat, les hagi classificat a la classe gat i el mateix amb les altres tres classes. És a dir, en el cas d'aquest treball, el quadrat és de 4x4, ja que hi ha 4 classes. Hauria de quedar així:

	Gat	Gos	Conill	Ocell
Gat	1	0	0	0
Gos	0	1	0	0
Conill	0	0	1	0
Ocell	0	0	0	1

Esquema 2.3 Matriu de confusió

Font: Elaboració pròpia

Com es pot observar, hi ha una línia perfecta en diagonal. Això voldria dir que no ha fallat en cap imatge i que les ha encertat el 100%. Aquest seria un exemple més complex:



Il·lustració 2.11 Matriu de confusió complexa

Font: Kaggle

3. Part pràctica

3.1 Entorn de treball

Per dur a terme la part pràctica, m'han autoritzat a accedir a un dels servidors del Centre de Visió per Computador com a eina de treball. En aquest servidor disposo d'un framework.

El framework (entorn de treball) que s'ha utilitzat està especialitzat en problemes de classificació d'imatges. Es basa en el llenguatge de programació Python i utilitza moltes llibreries en aquest llenguatge per fer funcions. Per exemple, per llegir imatges utilitza OpenCV, per fer xarxes neuronals utilitza PyTorch o per realitzar operacions amb matrius fa servir Numpy. Totes aquestes llibreries estan instal·lades i preparades al servidor. Utilitzen aquest llenguatge de programació i serveixen per facilitar-nos la feina no havent de programar-les des de zero nosaltres.

El framework utilitza la GPU i la CPU per funcionar. La CPU (Central Processing Unit) és el processador de l'ordinador on es realitzen operacions i càlculs. Bàsicament el processador amb el que els nostres mòbils, ordinadors i tablets funcionen. I la GPU (Graphics Processing Unit) és una targeta gràfica. Ens serveix per fer càlculs relacionats amb gràfiques, com poden ser videojocs, vídeos i altres funcions relacionades amb el que visualitzem a la pantalla. Una GPU està formada per petits processadors com els de la CPU però menys potents i més simples. La GPU però, té molts processadors petits que van bé per realitzar moltes operacions simples simultàniament. En canvi, la CPU pot fer operacions més complexes. Una GPU té 1500 processadors dels simples i la CPU fa servir 64 processadors potents.

A les xarxes neuronals es necessita realitzar moltes operacions, i operar amb 1500 processadors per càlculs simples és molt més eficient que fer-ho amb 64 de potents. Per això i perquè és més ràpida, és més útil utilitzar la GPU.

Per utilitzar el servidor framework, vaig fer-ho a través d'una aplicació anomenada MobaXTerm, dins la qual vaig poder crear la meva carpeta anomenada Anna on penjava tots els meus configs, programes, el meu Dataset, etc.

3.2 Preparant el dataset

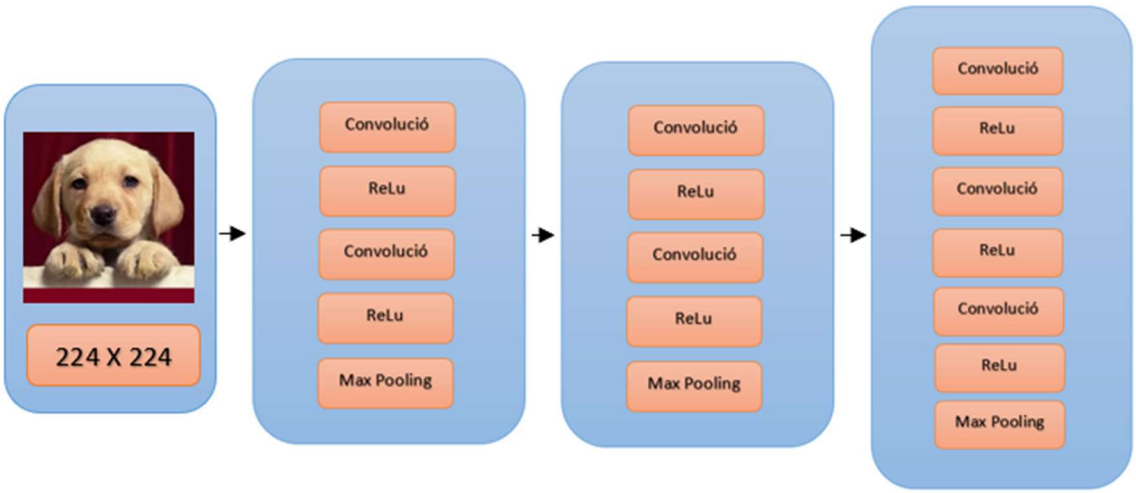
Com ja s'ha avançat anteriorment, el conjunt d'imatges es divideix en tres grups: entrenament, validació i test. Perquè és important aquesta divisió? Perquè si s'utilitza tot el contingut del Dataset, a l'hora de fer el test utilitzaria les mateixes imatges que l'entrenament i per tant obtindria uns molt bons resultats els quals no serien vàlids. A la que s'incloguessin imatges noves, la xarxa no seria capaç de classificar-les bé. Per aquest motiu és molt important separar bé el Dataset. Una bona manera de fer-ho, és fer que entreni amb el 60% de les imatges, validi amb el 20% i faci el test amb el 20% restant. El Dataset utilitzat en aquest treball està compost per 800 imatges, 200 de cada classe (gat, gos, conill i ocell).

3.3 La meva xarxa neuronal

Vaig crear la meva pròpia xarxa neuronal per classificació anomenada Anna_net (Anna Network). Però abans, vaig experimentar amb la Vgg16, un model que ja havia estat entrenat anteriorment i per tant, em va servir de base per desenvolupar posteriorment la meva.

L'Anna_net és una xarxa sense model entrenat anteriorment, i això suposa més dificultat a l'hora d'aprendre. Cal anar fent proves i comprovar si els resultats milloren. Mentre s'entrena la xarxa, es van detectant errors que cal corregir. Per exemple, un dels problemes que em vaig trobar va ser la normalització. La normalització consisteix en transformar els valors de la imatge que van de 0-255, a 0-1. Aquesta normalització no era correcta i es va haver d'arreglar.

Aquest seria l'esquema de la meva xarxa neuronal:



Esquema 3.1 La meva xarxa neuronal

Font: Elaboració pròpia

La xarxa neuronal està composta per tres blocs de convolucions amb un Pooling cada una. El primer i el segon bloc estan compostos per dues convolucions i dues ReLU. I el tercer per tres convolucions i tres ReLU.

Per què he utilitzat només dues o tres convolucions seguides? Bé, recordem que cada convolució analitza totes les parts de la imatge per aconseguir informació. Això és bo, però sotmetre-la, per exemple, a fer cinc convolucions seguides ens ocuparia molta memòria i la xarxa no donaria l'abast. Per això el que es fa és posar un Pooling, per transformar la imatge més petita i continuar amb les convolucions. Realitzar Poolings no només ajuda a estalviar memòria sinó que també ajuda a obtenir unes característiques més generals.

És important també el nombre de neurones que es volen utilitzar. A poc a poc es va incrementant el nombre perquè cada cop s'obté més informació. En aquest projecte entra un tensor que té la informació de les imatges així com el RGB explicat anteriorment i surten 512 neurones perquè ha estat el nombre amb el qual he obtingut millors resultats. Ara bé, és important no passar-se perquè podríem tornar a causar un excés de memòria.

I per últim, recordar que la funció ReLU determina quant important és la neurona a partir de la informació que ha rebut i fa de filtre per deixar passar només els nombres negatius. És per això que és important posar-la després de cada convolució.

3.4 Millora dels resultats a partir dels hípers paràmetres + gràfiques

Els hípers paràmetres tenen la funció de controlar l'experiment i els podem canviar per aconseguir que la xarxa doni millors resultats. A continuació s'explica els paràmetres més utilitzats, i entre parèntesis la paraula escrita perquè el programa ho entengui.

1) **Pretrained model**: S'escriu si és un model entrenat anteriorment ('basic'), un model nou ('None') o un model personal diferent dels altres ('custom').

2) **Resize image**: s'especifica quina grandària té la imatge (224, 224).

3) **Labels**: Aquest paràmetre s'especifica el nom dels tipus de classes que s'utilitzen per classificar. Concretament, en aquest projecte els noms són: ('Perros', 'Gatos', 'Conejos', 'Pájaros').

4) **Map labels**: És el número que associa el sistema a cada classe ('Perros': 2, 'Gatos': 1, 'Conejos': 0, 'Pájaros': 3).

5) **Num classes**: Defineix el nombre de classes que té el nostre sistema (4).

6) **Epochs**: Nombre de vegades que analitzarà cadascuna de les imatges (100).

7) **Optimizer**: Són els paràmetres que ajuden a trobar el descens de gradient més fàcilment: (adam), (RMSprop), (SGD).

8) **Learning rate**: Nombre negatiu amb el qual farà el descens de gradient. Com més gran més general busca la solució, i com més petit més detalladament. (-4)

9) **Scheduler**: Canvia el learning rate. S'especifica el número d'epochs que ha d'analitzar perquè disminueixi el número de learning rate ('Step').

També es poden indicar exactament els números d'epochs els quals vols que decreixi el learning rate ('MultiStep'). I per últim es pot fer que no canviï el nombre i que sempre s'utilitzi el mateix (None).

10) **Early stopping**: Aquest paràmetre s'activa en cas que es vulgui aturar l'experiment perquè la xarxa veu que no aprèn més (True) o bé no s'activa i es deixa que continuï l'experiment (False).

11) **Hflips**: Aquest dóna la volta a les imatges perquè siguin diferents i el sistema es pensi que li entra una nova (True) o en cas que no es vulgui realitzar (False).

12) **Random Dist**: Modifica les imatges fent-les més borroses perquè el sistema també sàpiga operar amb imatges que no siguin nítides (True) o en cas que no es vulguin modificar (False).

3.4.1 Anna_net

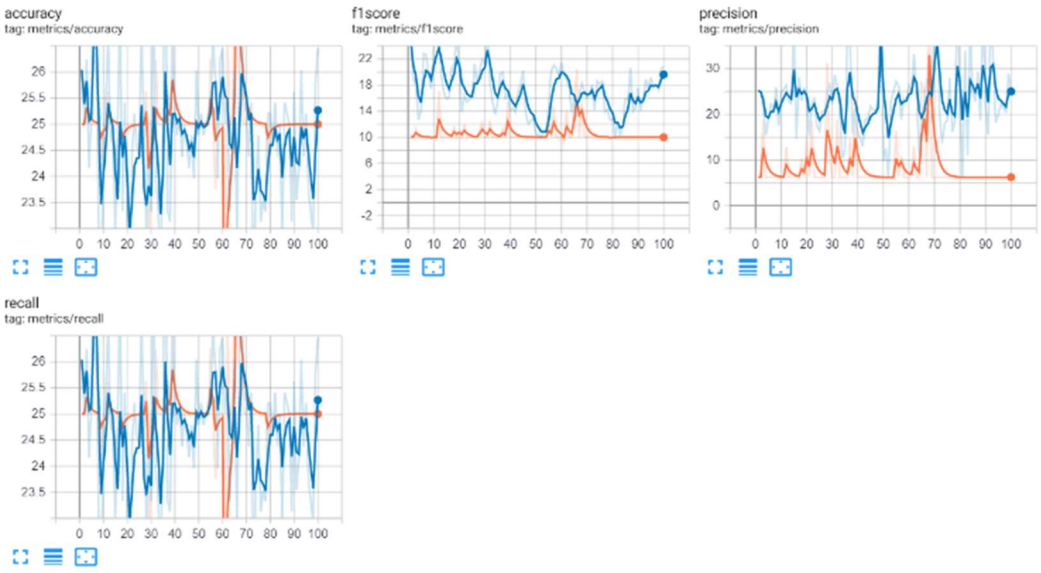
Ara mostro els paràmetres utilitzats pel model Anna_net amb cada una de les opcions explicades anteriorment.

1) **Pretraied model**: 'None' → Ja que és un model que comença des de zero.
2) **Resize image**: [224, 224]
3) **Labels**: ('Perros', 'Gatos', 'Conejos', 'Pájaros')
4) **Map labels**: ('Perros': 0, 'Gatos': 1, 'Conejos': 2, 'Pájaros': 3)
5) **Num classes**: 4
6) **Epochs**: 150
7) **Optimizer:** 'adam'
8) **Learning rate:** -4
9) **Scheduler: '**Step' (cada 30 epochs disminueix el learning rate)
10) **Early stopping:** False

11) **Hflips:** True
12) **Random Dist:** False

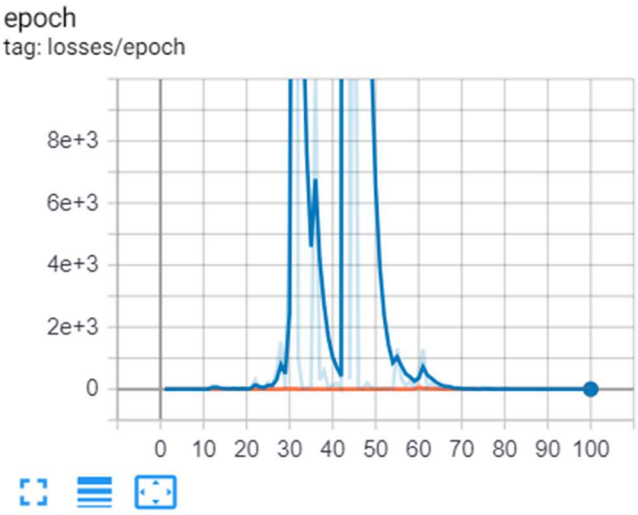
Per cadascun d'aquests paràmetres disposo gràfiques que mostren els resultats obtinguts. Però si les inclogués totes dins d'aquest apartat, ocuparia molt d'espai. Per això he decidit posar aquí un d'aquests paràmetres com a exemple i la resta incloure'ls a l'apartat d'Annex, perquè considero que són importants i significatius en aquest treball.

Així, el paràmetre escollit és l'Optimizer. En ell es mostra perquè he elegit l'adam i no el RMSprop o el SGD. Les gràfiques mostrades seguidament són visualitzades en una plataforma anomenada Tensorboard la qual només és accessible des del CVC de la UAB.



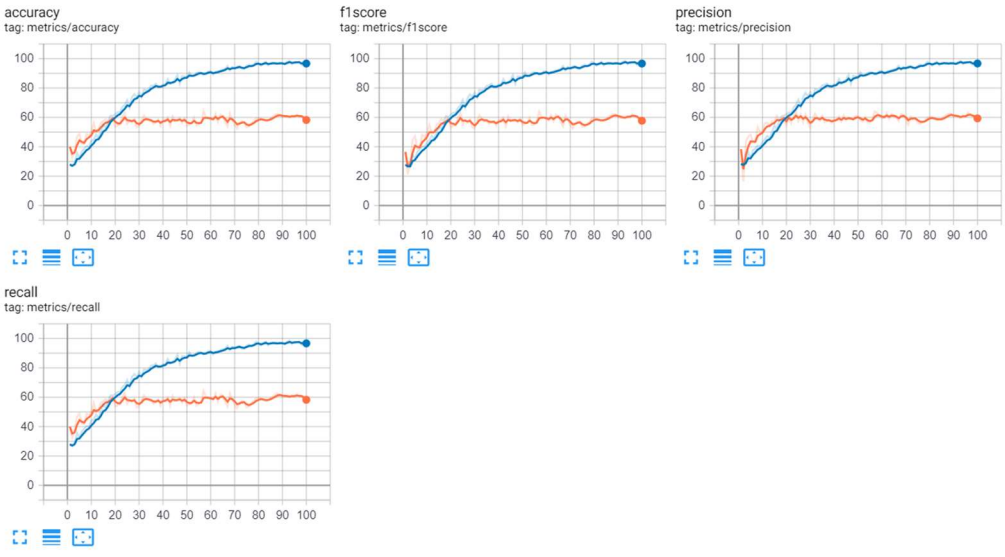
Il·lustració 3.1 Mètriques RMSprop

Font: Captura Tensorboard



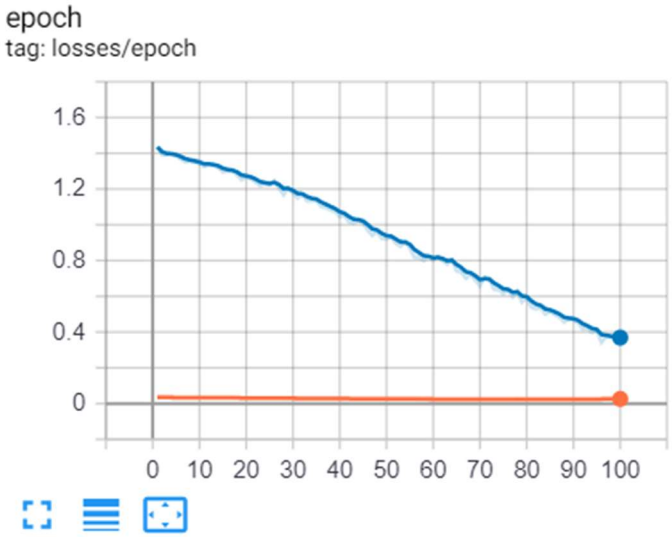
Il·lustració 3.2 Loss RMSprop

Font: Captura Tensorboard



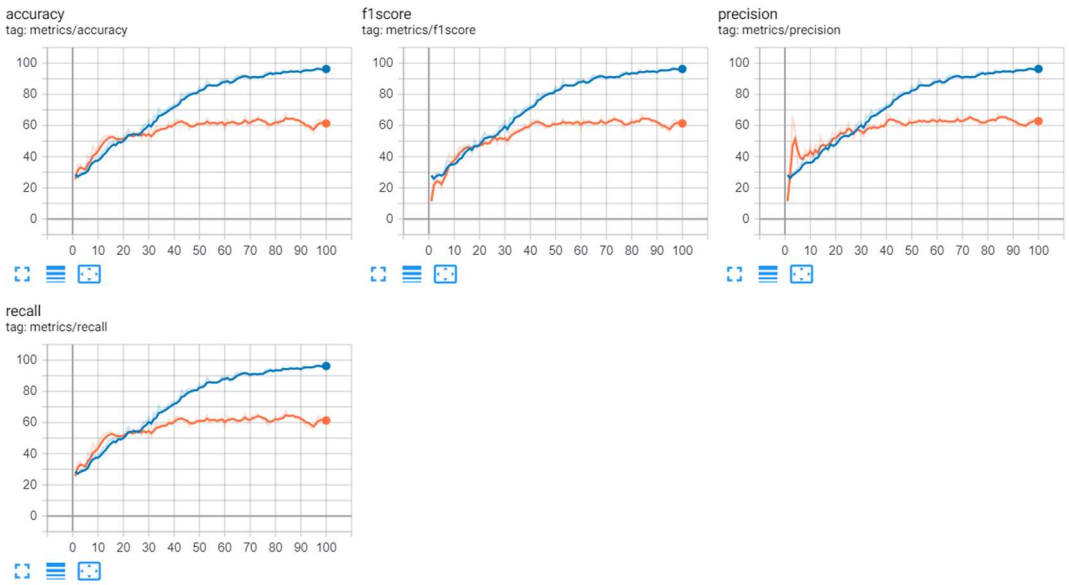
Il·lustració 3.3 Mètriques SGD

Font: Captura Tensorboard



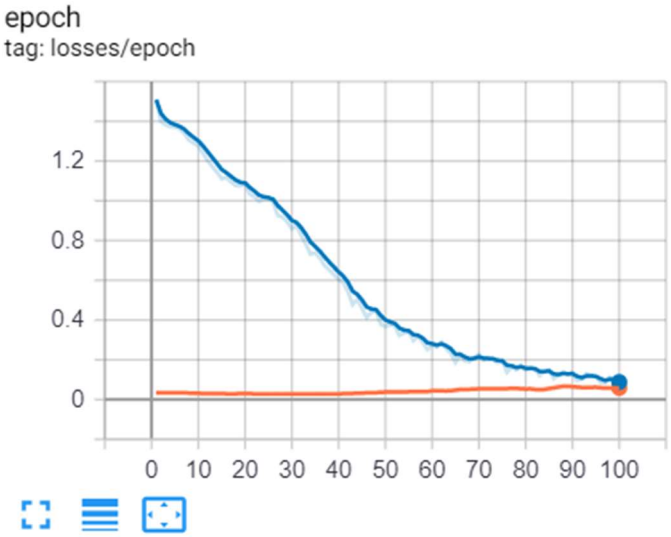
Il·lustració 3.4 Loss SGD

Font: Captura Tensorboard



Il·lustració 3.5 Mètriques Adam

Font: Captura Tensorboard



Il·lustració 3.6 Loss Adam

Font: Captura Tensorboard

A partir de comparar les mètriques i la loss de cadascun dels paràmetres, s'aprecia que en les il·lustracions 3.1 i 3.2 (RMSprop), el resultat ha estat erroni. Seguidament, en les il·lustracions 3.3 i 3.4 (SGD) podem apreciar una gran millora. I en les últimes que són les 3.5 i 3.6 (adam) s'obté millor loss i millors resultats, i en conseqüència, aquest paràmetre és el que s'ha utilitzat pels experiments.

3.4.2 Vgg16

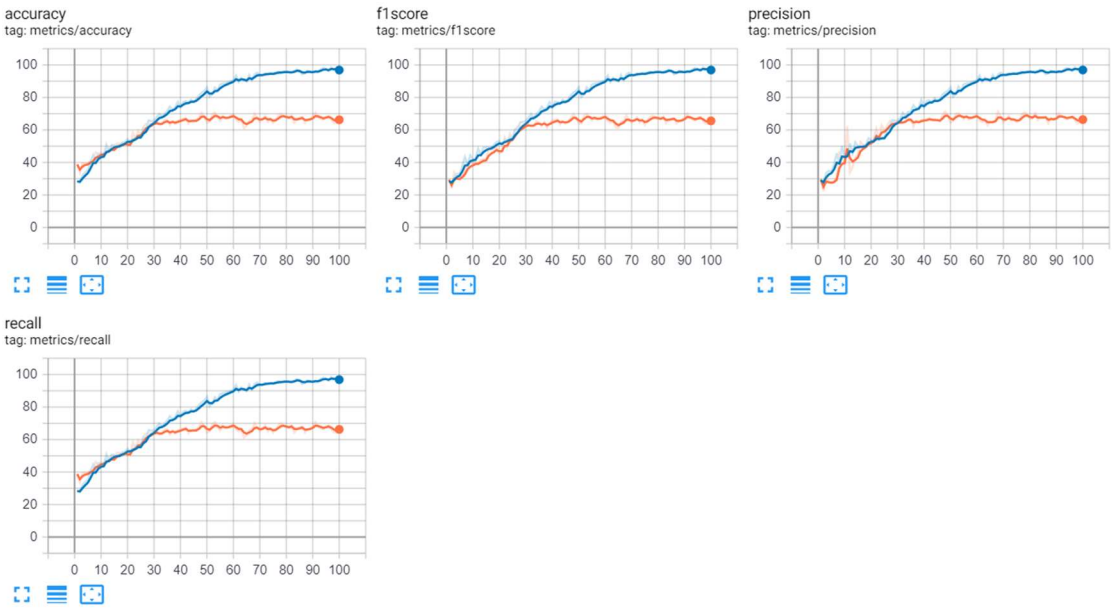
Com s'ha explicat, també s'han realitzat experiments amb un segon model el qual ja havia estat entrenat prèviament. Es mostren a continuació els paràmetres utilitzats en aquest model anomenat Vgg16.

- **Pretraied model:** 'basic'
- **Resize image**: [224, 224]
- **Labels**: ('Perros', 'Gatos', 'Conejos', 'Pájaros')
- **Map labels**: ('Perros': 0, 'Gatos': 1, 'Conejos': 2, 'Pájaros': 3)
- **Num classes**: 4
- **Epochs**: 100
- **Optimizer**: 'adam'
- **Lerning rate:** -4
- **Scheduler**: Step (cada 20 epochs disminueix el learning rate)
- **Early stopping**: True
- **Hflips:** True
- **Random Dist:** False

3.5 Gràfiques dels millors resultats + Matriu de confusió

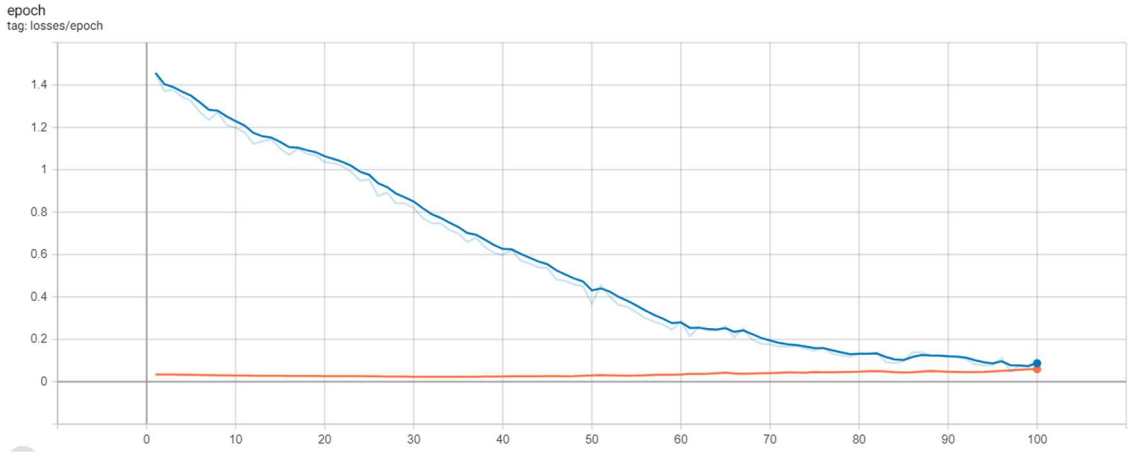
Analitzem la diferència entre els dos models fent servir els millors hípers paràmetres utilitzats anteriorment. Compararem les seves gràfiques i les seves matrius de confusió.

3.5.1 Anna_net



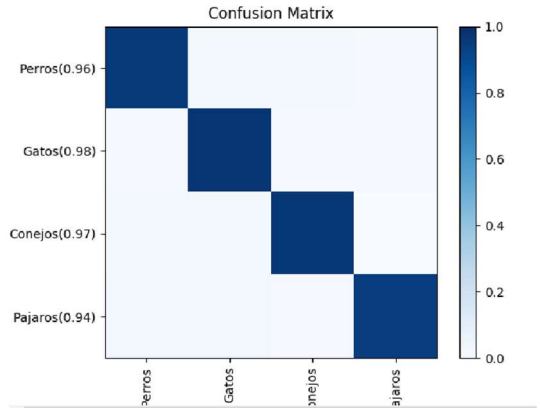
Il·lustració 3.7 Mètriques Anna_net

Font: Captura Tensorboard



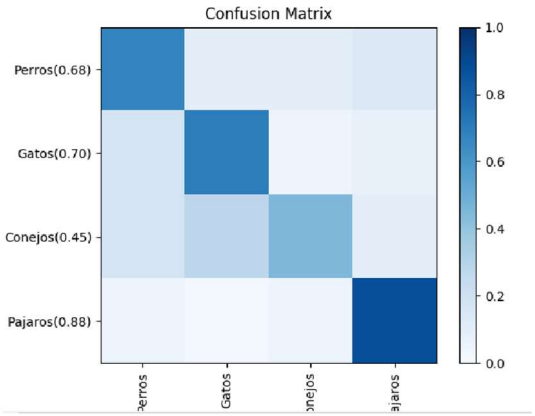
Il·lustració 3.8 Loss Anna_net

Font: Captura Tensorboard



Il·lustració 3.9 Confusió de matriu entrenament

Font: Captura Tensorboard



Il·lustració 3.10 Confusió de matriu validació

Font: Captura Tensorboard

A la il·lustració 3.7 s'observa com la xarxa aprèn ja que la línia blava, la qual correspon a l'entrenament, augmenta. Però en canvi la línia taronja que és la de validació, no és tan bona i aquí podem apreciar un problema d'Overfitting.

En la il·lustració 3.8 es veu com la loss baixa, el que ens indica que està aprenent correctament.

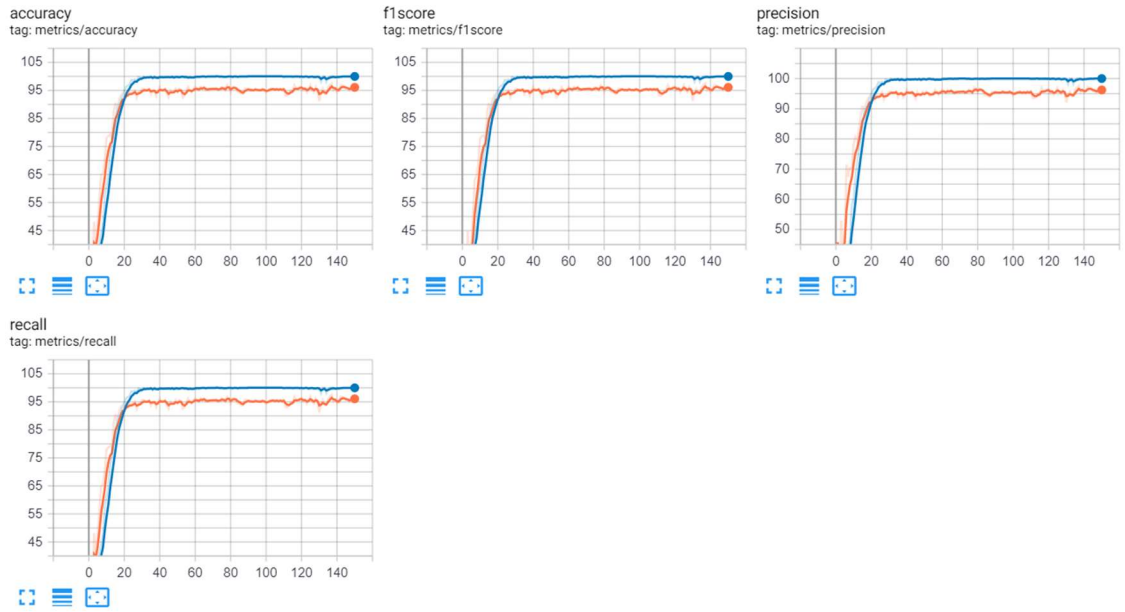
Seguidament a la il·lustració 3.9, que és la matriu de confusió de l'entrenament, és perfecte ja que s'aprecia una línia clarament en diagonal.

I per últim, la il·lustració 3.10, la qual correspon a la de validació, no és tan bona com l'entrenament, però tot i així es pot apreciar bastant bé la diagonal.

El problema que la validació no sigui tan eficient com l'entrenament es deu a que al tractar-se d'un model nou, és molt bo aprenent, però a la validació els resultats ja no són tan òptims a causa de la poca disponibilitat d'imatges o per l'excés d'entrenament.

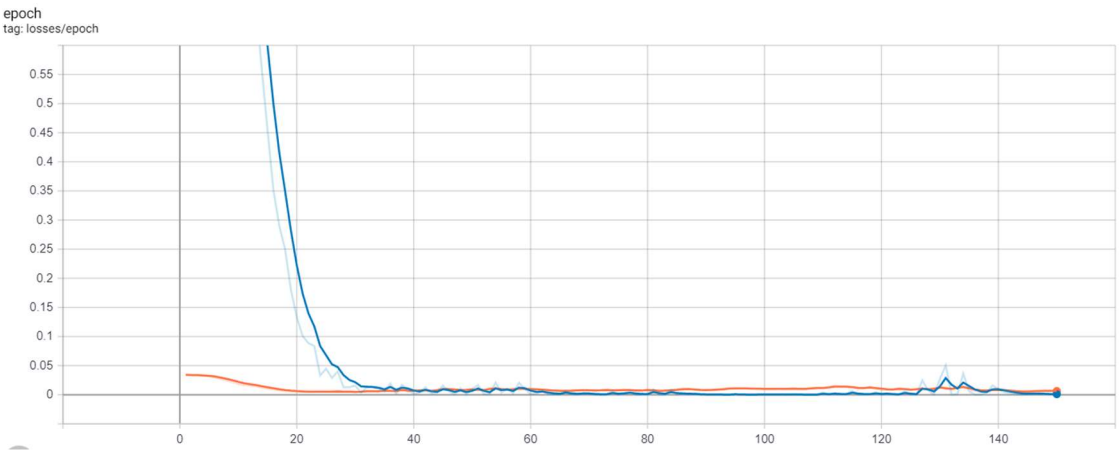
3.5.2 Vgg16

Ara observem el mateix però amb el model de Vgg16:



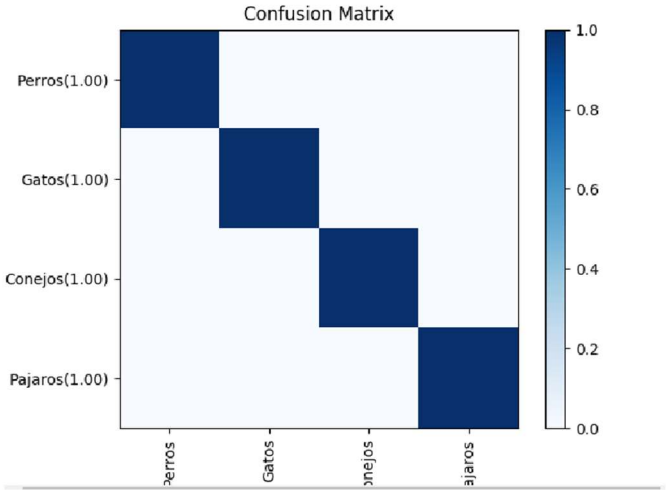
Il·lustració 3.11 Mètriques Vgg16

Font: Captura Tensorboard



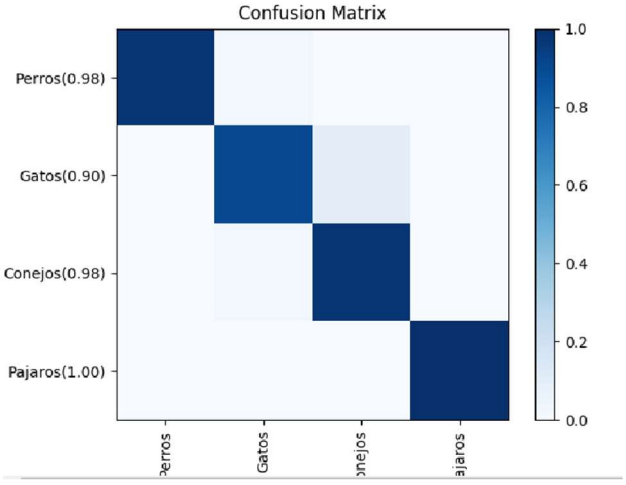
Il·lustració 3.12 Loss Vgg16

Font: Captura Tensorboard



Il·lustració 3.13 Confusió de matriu entrenament

Font: Captura Tensorboard



Il·lustració 3.14 Confusió de matriu validació

Font: Captura Tensorboard

En la il·lustració 3.11 s'observen unes mètriques perfectes. L'Overfitting és bastant reduït i la xarxa és molt constant en l'entrenament.

En la il·lustració 3.12 s'observa una loss molt definida i clara.

En la il·lustració 3.13 tenim una línia perfecte en la matriu de confusió que ens indica com ha estat l'entrenament.

I per últim, en la il·lustració 3.14 tenim la matriu de confusió de la validació, la qual és gairebé perfecte.

El fet que l'entrenament sigui tan constant, es deu a que la Vgg16 és un model entrenat anteriorment i per tant aquesta part la té molt treballada. En aquest cas, el model no necessita tantes epochs per l'entrenament, i en conseqüència li podem reduir el nombre.

3.6 Programa per dividir el dataset

És important fer correctament la divisió del dataset en els tres grups abans esmentats: imatges per entrenament, per validació i per test. Per dur a terme aquesta divisió vaig crear un programa específic amb els següents paràmetres:

- Un 60% de les imatges s'utilitzen per l'entrenament.
- Un 20% de les imatges s'utilitzen per la validació.
- I el 20% últim de les imatges s'utilitzen pel test.

3.7 Programa per etiquetar les imatges

Un cop obtinguts els resultats dels experiments, vaig voler anar una mica més enllà. Els resultats són visualitzats a través d'una llista i classificats segons el número que li correspongui. Per aquest motiu vaig decidir fer un programa que permetés etiquetar cada imatge amb el nom de la classe a la qual s'havia classificat. D'aquesta manera si obrim la imatge, podem veure el nom de la classe escrit en un costat. Aquest és un exemple per cada classe.

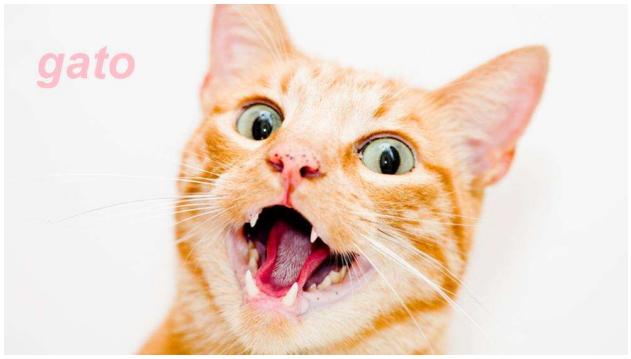
Il·lustració 3.15 Imatge etiquetada "pajaro[1]"

Font: Elaboració pròpia

[1] A "Pajaro" no se li posa accent perquè en programació és preferible evitar-los per no causar problemes de sintaxis.

Il·lustració 3.16 Imatge etiquetada "conejo"

Font: Elaboració pròpia



Il·lustració 3.17 Imatge etiquetada "gato"

Font: Elaboració pròpia

Il·lustració 3.18 Imatge etiquetada "perro"

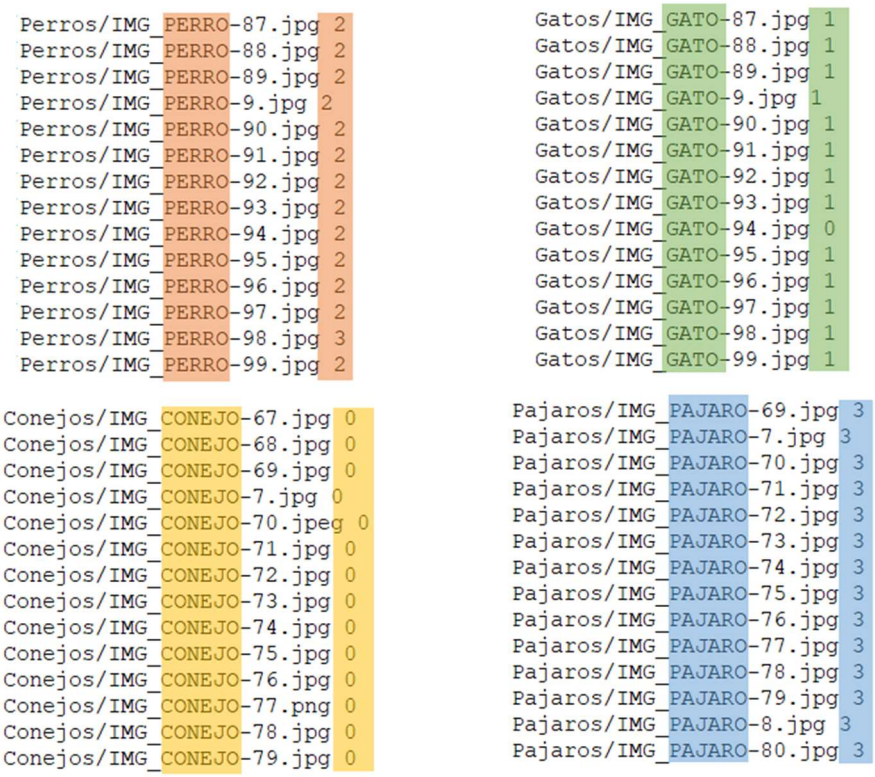
Font: Elaboració pròpia

3.8 Resultats en la classificació

Un cop entrenat el sistema, se li fa el test per observar els resultats. Aquests resultats s'obtenen en una llista amb un número del 0 al 3, sent 0 conill, 1 gat, 2 gos i 3 ocell. És possible que falli en algun d'aquests elements perquè el sistema coneix la classe d'animal segons les imatges que entrena. Si en l'entrenament hi havia conills blancs, i al test se li ensenya un gos pelut blanc, hi ha molta probabilitat que s'equivoqui i digui que és un conill.

3.8.1 Vgg16

Aquí es mostren alguns dels resultats obtinguts amb el model que ja ve entrenat anomenat Vgg16:



Il·lustració 3.19 Captures test

Font: Captura MobaXTerm

En aquestes captures del test, es pot observar com la Vgg16 les encerta totes menys algun cas en concret. Quan observem que es produeix una errada d'aquest tipus, anem a analitzar quines poden ser les causes.

Recordem que la xarxa aprèn què és cada cosa a partir de les imatges que disposa a l'entrenament, si a l'hora del test li ensenyem una imatge que no ha vist anteriorment, és difícil que la classifiqui correctament. A més, també és possible que dues imatges de classes diferents siguin molt semblants. Observem aquest exemple.

Aquí tenim la imatge 94, que és la que ha fallat en la captura de resultats de gat en la il·lustració 3.19:

Il·lustració 3.20 Imatge etiquetada erròniament

Font: Font: Elaboració pròpia

A l'hora de fer el test ha dit que era un conill i no un gat. Anem a fer la hipòtesi suposant que una imatge com aquesta no l'havia vist a l'hora de l'entrenament.

Observem les imatges utilitzades per l'entrenament i mirem si tenim una imatge semblant a aquella. I efectivament la tenim:

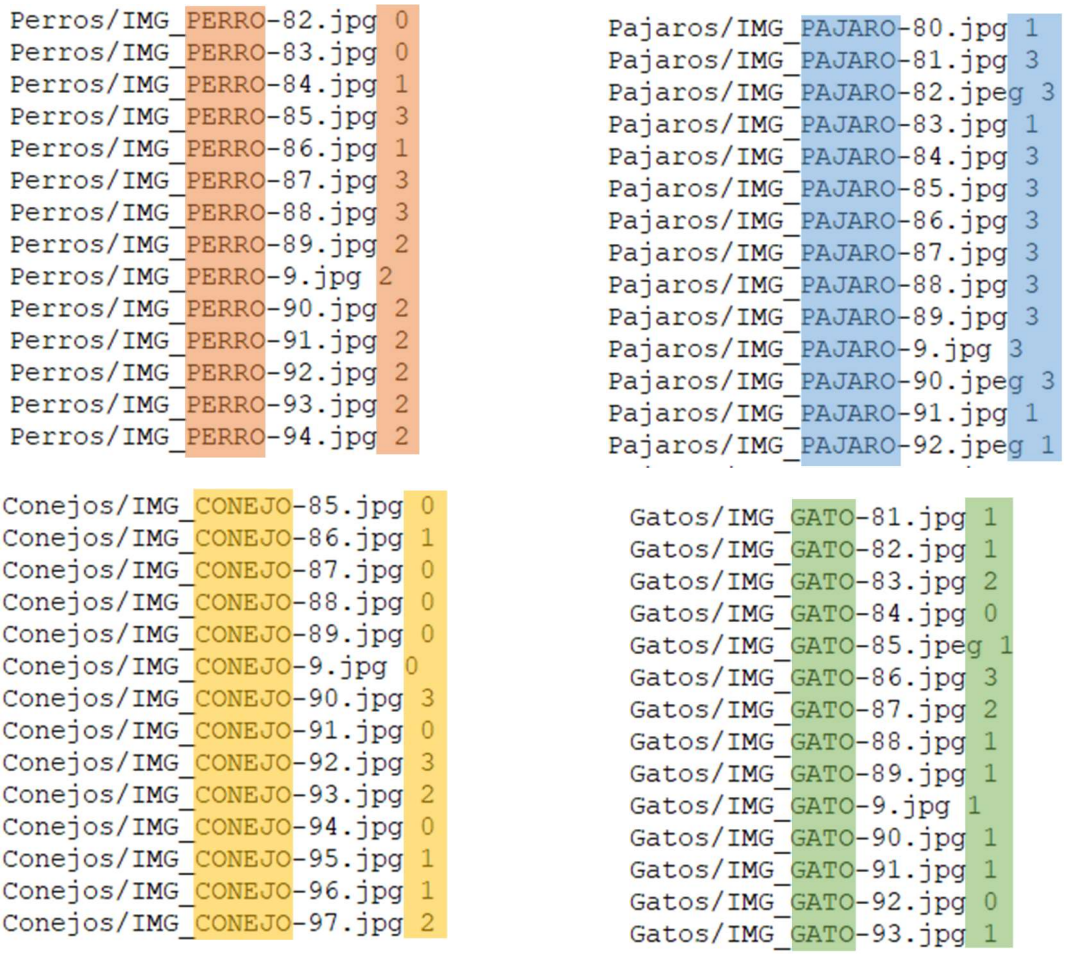
Il·lustració 3.21 Imatge gat conjunt entrenament

Font: Elaboració pròpia

Per tant, la conclusió que obtenim és que l'errada no ha estat per falta d'aprenentatge, sinó per semblança física de dos animals de classe diferent, és a dir, tot i haver vist una imatge d'un gat amb les mateixes característiques a l'entrenament, no ha estat capaç de classificar-ho correctament per la gran semblança que té aquest gat amb un conill.

3.8.2 Anna_net

Aquests són els resultats amb el model Anna_net:



Imatge 3.22 Captures test

Font: Captura MobaXTerm

Podeu veure que aquí el sistema falla molt més que en la Vgg16. Això és normal, ja que és un model personal que parteix des de zero. Per solucionar aquests problemes es pot recórrer al model i provar estructures diferents, o fer algun canvi als hípers paràmetres. Tot i això, es fa molt difícil aconseguir els mateixos bons resultats que la Vgg16 perquè no es parteix del mateix inici.

Dels resultats de la xarxa Anna_net, us mostro un exemple de l'error en cada cas:

Il·lustració 3.23 Imatge etiquetada errròniament

Font: Elaboració pròpia

Il·lustració 3.24. Imatge etiquetada errròniament

Font: Elaboració pròpia

Il·lustració 3.25 Imatge etiquetada erròniament

Font: Elaboració pròpia

Il·lustració 3.26 Imatge etiquetada erròniament

Font: Elaboració pròpia

4. Conclusions

S'han aconseguit els objectius plantejats inicialment?

El meu objectiu era crear un model de xarxa neuronal capaç de distingir entre quatre classes d'animals. Aquest objectiu s'ha aconseguit, i per tant disposo d'una xarxa entrenada per diferenciar entre si és gat, gos, ocell o conill.

S'ha complert la hipòtesi plantejada?

La hipòtesi d'aquest treball era investigar sobre la possibilitat que un model de xarxes neuronals entrenat des de zero, fos capaç d'obtenir un alt percentatge de fiabilitat. La resposta a la hipòtesi és parcialment correcta, la xarxa Anna_net ha aconseguit obtenir un rendiment del 65-70%. Al començament del treball pensava en la possibilitat d'obtenir un rendiment elevat aproximat a un 100%. Però ara, a partir de tot el coneixement que he adquirit i després d'haver estat treballant tan de temps amb aquest projecte, m'he adonat que realment no és possible aconseguir el màxim rendiment perquè totes les xarxes neuronals tenen un mínim d'error per molt petit que sigui. Dins del marc limitat que constitueix un treball de recerca de batxillerat, no és possible aconseguir un objectiu tan alt com el que em vaig proposar.

L'Anna_net ha estat sotmesa a diversos entrenaments i canvis per augmentar el percentatge d'encerts. Tot i això, no s'ha aconseguit un rendiment òptim. La xarxa està composta per 800 imatges, 200 per cada classe. Hi ha una alta probabilitat que la causa d'aquest baix rendiment sigui l'escassedat de dataset ja que molts datasets utilitzen 10k, 100k o fins i tot 1 milió d'imatges. La meva xarxa era petita, i per tant vaig decidir agafar només aquestes quatre classes amb aquest nombre d'imatges. Afegint més quantitat d'imatges al dataset, pot ser una solució al problema, el qual més endavant es pot comprovar.

Comparant un model entrenat anteriorment, la Vgg16, amb el meu personal, l'Anna_net, he observat la gran dificultat que suposa entrenar una xarxa neuronal des de zero. L'Anna_net ha d'estar sotmesa a més entrenaments i ha d'estar composta per una gran varietat d'imatges en el dataset perquè pugui aprendre de tot tipus de característiques.

Quines dificultats s'han trobat durant el procediment?

Durant el treball he trobat petites dificultats a l'hora de programar. Quan realitzava un experiment, en alguns casos no podia a causa d'algun error, com per exemple de sintaxis de codi el qual em costava trobar i poder rectificar. Entendre en profunditat el llenguatge de programació no és fàcil, i tot i que ara tinc més coneixement sobre el tema, encara tinc petites dificultats en aquest àmbit.

La metodologia emprada ha estat l'adequada?

La metodologia sí que ha estat l'adequada. Els recursos utilitzats per la comprovació de la hipòtesi han estat adients, ja que s'ha dut a terme tots els passos per arribar a la conclusió. Aquests passos, per exemple, són el canvi d'arquitectura de xarxa o de paràmetres per intentar optimitzar els resultats.

Durant el projecte, la part pràctica que tenia menys clara era la de si ampliar el meu treball i fer-lo enfocat a la detecció i segmentació o quedar-me amb la classificació. Finalment vaig decidir no fer-ho per falta de temps i d'aquesta manera centrar-me en un únic objectiu que és el de classificació d'imatges.

4.1 Valoració personal

A l'inici d'aquest projecte tenia un gran desconeixement sobre xarxes neuronals i programació. Realitzar aquest treball de recerca m'ha aportat un aprenentatge considerable de programació en Python i de Machine Learning, concretament Deep Learning.

Al principi dubtava de fer aquest treball perquè era molt inexperta amb aquest tema i temia no ser capaç de portar-lo a terme. Finalment, estic molt contenta que m'oferissin aquesta oportunitat, perquè tot i les hores que hi he dedicat, que són moltes, me n'he sortit molt bé i estic molt satisfeta amb el meu treball.

A més a més, aquest projecte ha fet adonar-me de la gran dificultat que hi ha en un àmbit que sembla tan senzill com separar una imatge en gat o gos. Cal remarcar que a pesar de ser un treball molt basat en la part pràctica, no ens podem deixar tota la part teòrica perquè sense ella no és possible realitzar o entendre el que estàs fent.

I per últim, una part que m'ha resultat una mica més complicada, és la part de redactar conceptes tan abstractes com és el cas del learning rate o el descens de gradient. Aquests són conceptes importants de saber però difícils d'explicar i d'entendre.

5. Bibliografia

- Blogthinkbig.com (08-02-17). Diferencias entre machine learning y deep learning. Consultat el dia 15 de juliol de 2019 a https://blogthinkbig.com/diferencias-entre-machine-learning-y-deep-learning
- En mi local funciona (05-12-18). Deep Learning básico con Keras (Parte 4): ResNet. Consultat el dia 19 de juliol de 2019 a https://enmilocalfunciona.io/deep-learning-basico-con-keras-parte-4-resnet/
- En mi local funciona (04-02-19). Deep Learning básico con Keras (Parte 5): DenseNet. Consultat el dia 19 de juliol de 2019 a https://enmilocalfunciona.io/deep-learning-basico-con-keras-parte-5-densenet/
- Wikipedia (26-06-19). AlexNet. Consultat el dia 19 de juliol de 2019 a https://en.wikipedia.org/wiki/AlexNet
- Neural Networks Framework. Breve Historia de las redes Neuronales. Consultat el dia 21 de juliol de 2019 a http://www.redes-neuronales.com.es/tutorial-redes-neuronales/historia-de-las-redes-neuronales.htm
- Diego Calvo (10-12-18). Función de coste – Redes neuronales. Consultat el dia 21 de juliol de 2019 a http://www.diegocalvo.es/funcion-de-coste-redes-neuronales/
- Towards Data Science (14-08-16). A Concise History of Neural Networks. Consultat de dia 26 de juliol de 2019 a https://towardsdatascience.com/a-concise-history-of-neural-networks-2070655d3fec
- Wikipedia (03-07-19). Visión artificial. Consultat el dia 13 d'agost de 2019 a https://es.wikipedia.org/wiki/Visi%C3%B3n_artificial

- YouTube (24-01-16). Historia Redes Neuronales Artificiales. Consultat el dia 14 d'agost de 2019 a https://www.youtube.com/watch?v=a2_JHFtbFZA
- Xataka (28-10-16). Deep Learing: qué es y por qué va a ser una tecnologia clave en el futuro de la intel·ligència artificial. Consultat el dia 27 d'agost de 2019 a https://www.xataka.com/robotica-e-ia/deep-learning-que-es-y-por-que-va-a-ser-una-tecnologia-clave-en-el-futuro-de-la-inteligencia-artificial
- Decsai. Capítulo 5: arays y cadenes. Consultat el dia 30 d'agost de 2019 a http://decsai.ugr.es/~jfv/ed1/c/cdrom/cap5/f_cap52.htm
- Xataka (21-01-16). Las redes neuronales: qué son y por qué están volviendo. Consultat el dia 4 de setembre de 2019 a https://www.xataka.com/robotica-e-ia/las-redes-neuronales-que-son-y-por-que-estan-volviendo
- Sitiobigdata. Innovaciones arquitectónicas en redes neuronales. Consultat el dia 4 de setembre de 2019 a http://sitiobigdata.com/2019/05/01/innovaciones-arquitectonicas-redes-neuronales-clasificacion-imagenes/#
- EngMRK (30-09-18). LeNet-5 – A Classic CNN Architecture. Consultat el dia 6 d'octubre de 2019 a https://engmrk.com/lenet-5-a-classic-cnn-architecture/

6. Bibliografia d'imatges

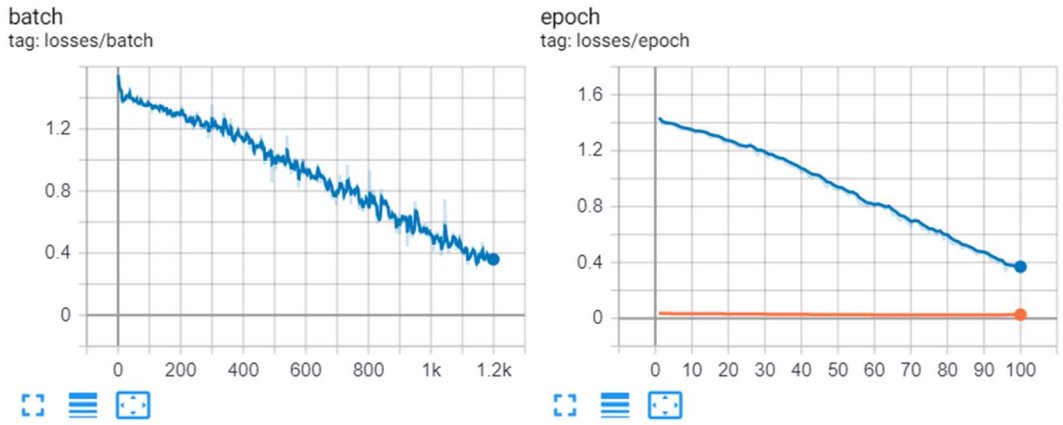
- Il·lustració 1.1 Exemple de segmentació. Nanonets (2018). Consultat el dia 1 d'octubre de 2019 a https://nanonets.com/blog/how-to-do-semantic-segmentation-using-deep-learning/
- Il·lustració 1.2 Exemple de detecció. Sigmoidal. Consultat el dia 1 d'octubre de 2019 a https://sigmoidal.io/dl-computer-vision-beyond-classification/
- Il·lustració 2.1 Gràfic descens de gradient. Hackernoon (10-03-18). Consultat el dia 4 d'agost de 2019 a https://hackernoon.com/gradient-descent-aynk-7cbe95a778da
- Il·lustració 2.4 Funcionament Vgg. En mi local funciona (01-08-18). Consultat el dia 7 d'agost de 2019 a https://enmilocalfunciona.io/deep-learning-basico-con-keras-parte-3-vgg/
- Il·lustració 2.5 Gràfica ReLU. Viquipèdia (14-07-19). Consultat el dia 5 d'agost de 2019 a https://es.wikipedia.org/wiki/Rectificador_(redes_neuronales)
- Il·lustració 2.6 Esquema de Max Pooling i Average Pooling. Quora (16-06-17). Consultat el dia 6 d'octubre de 2019 a https://www.quora.com/What-is-max-pooling-in-convolutional-neural-networks
- Il·lustració 2.7 Fully-Connected. Fernando Sancho Caparrini (23-04-17). Consultat el dia 16 d'agost de 2019 a http://www.cs.us.es/~fsancho/?e=165
- Il·lustració 2.8 Funcionament AlexNet. Towards Data Science (29-07-19). Consultat el dia 21 de setembre de 2019 a https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d

- Il·lustració 2.9 Funcionament ResNet. En mi local funiona (05-12-18). Consultat el dia 21 de setembre de 2019 a https://enmilocalfunciona.io/deep-learning-basico-con-keras-parte-4-resnet/
- Il·lustració 2.10 Funcionament DenseNet. En mi local funciona (04-02-19). Consultat el dia 21 de setembre a https://enmilocalfunciona.io/deep-learning-basico-con-keras-parte-5-densenet/
- Il·lustració 2.11 Matriu de confusió complexa. Kaggle (desembre de 2018). Consultat el dia 9 de setembre de 2019 a https://www.kaggle.com/c/PLAsTiCC-2018/discussion/74564

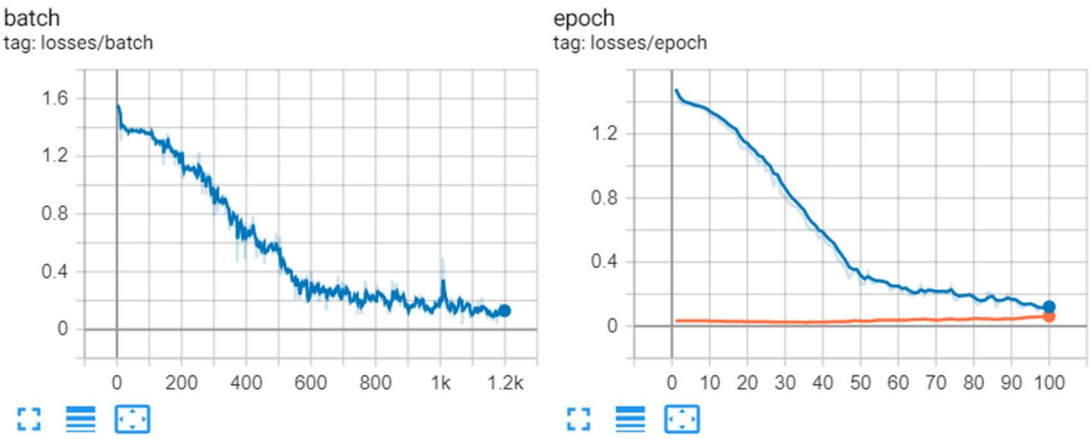
Annex

A continuació es mostren la resta de gràfiques de loss de cada paràmetre del model Anna_net i les conclusions de per què he elegit un tipus o un altre.

Learning rate:



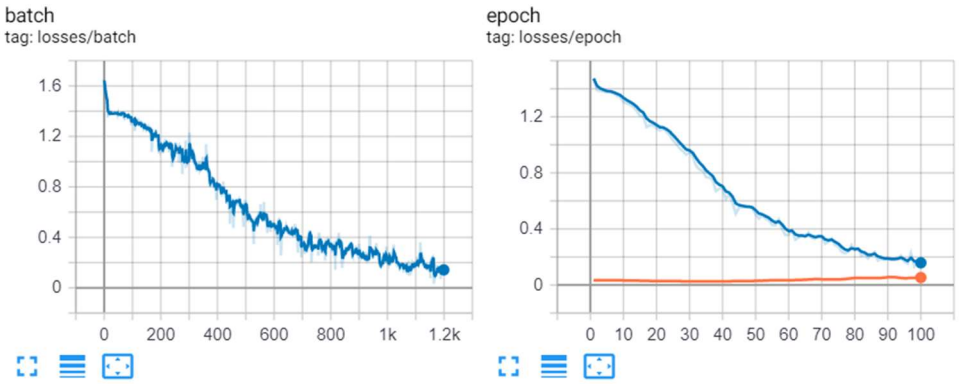
Il·lustració 1. Learning rate a -5



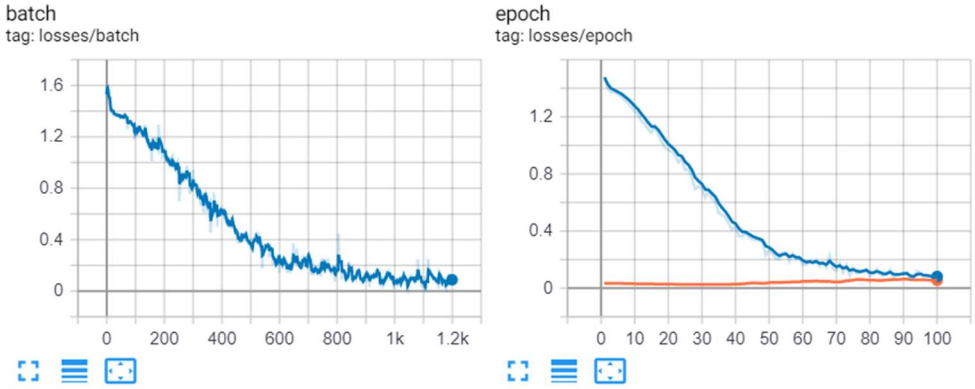
Il·lustració 2. Learning rate a -4

Amb els dos tipus de learning rate que s'han provat, a partir de la loss podem observar que el millor és el learning rate a -4 (Il·lustració 2).

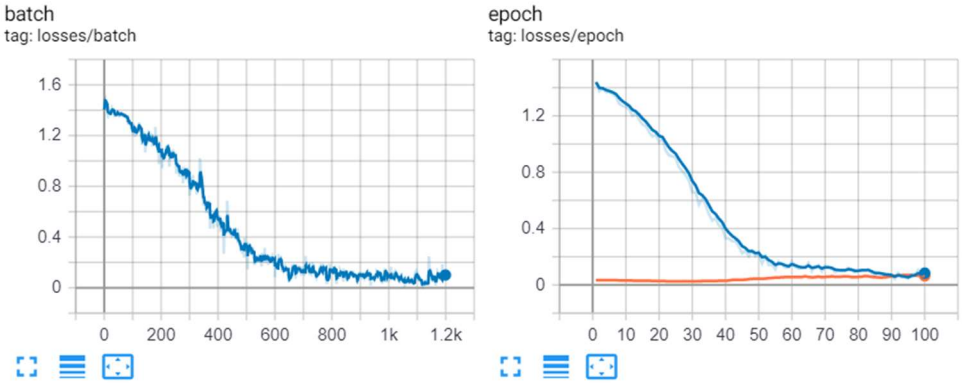
Scheduler:



Il·lustració 3. Multistep



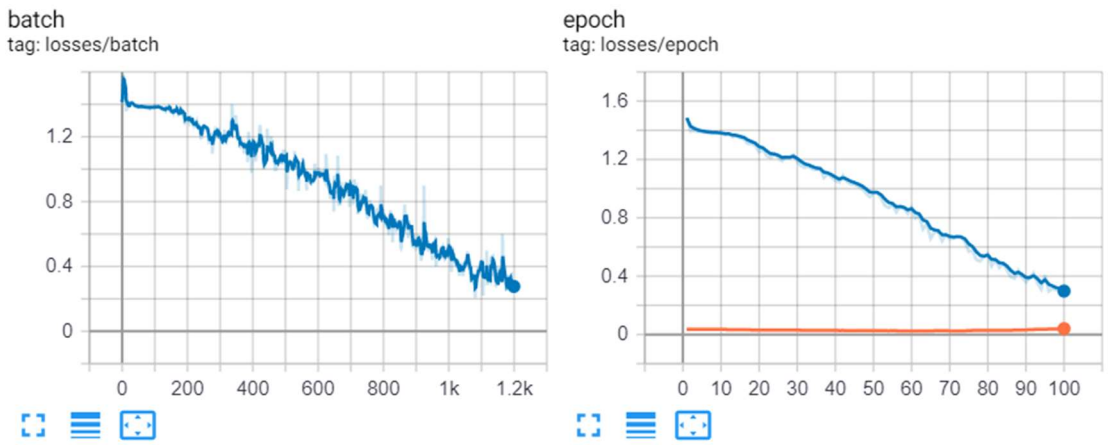
Il·lustració 4. None



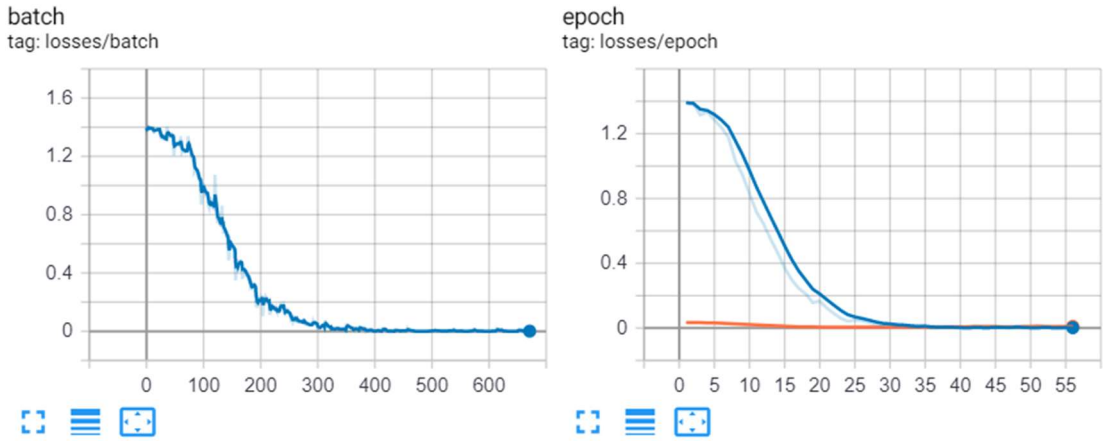
Il·lustració 5. Step

En el cas del Scheduler, la millor loss obtinguda ha estat la última gràfica amb el paràmetre a Step (il·lustració 5). La segona més precisa ha estat la None (il·lustració 4), i per últim, la menys precisa ha estat la MultiStep (il·lustració 3).

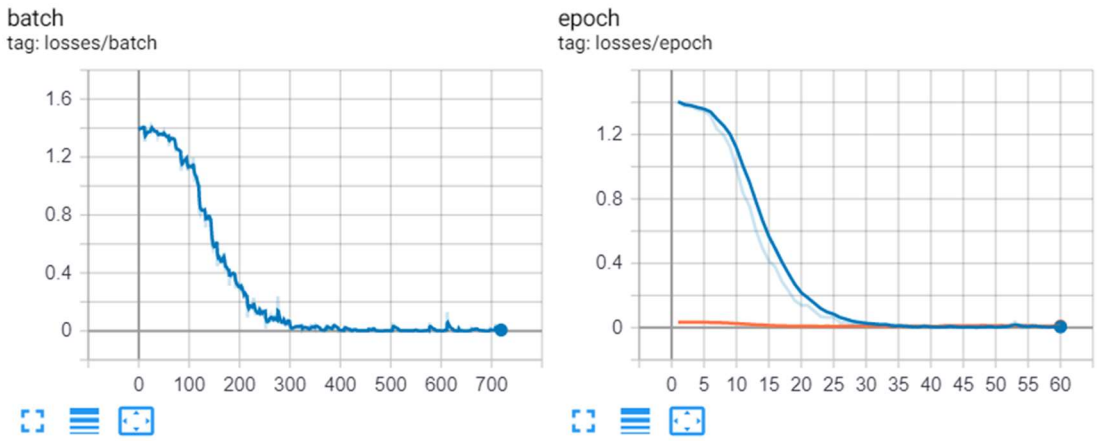
Hflips i Random Dist:



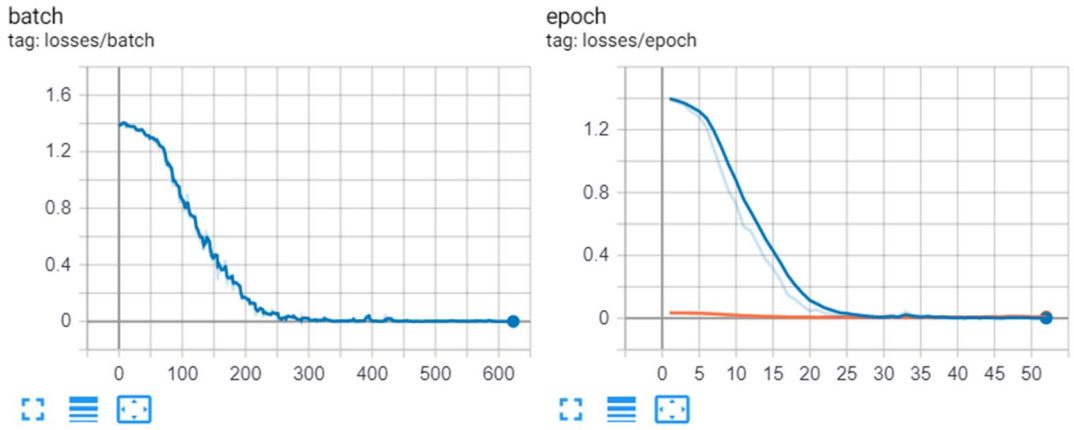
Il·lustració 6. Hflips a True i Random Dist a True



Il·lustració 7. Hflips a False i Random Dist a False



Il·lustració 8. Hflips a False i Random Dist a True



Il·lustració 9. Hflips a True i Random Dist a False

I per últim, en el cas dels Hflips i Random Dist, en ordre creixent de menys precís a més seria:

1. Hflips: True i Random Dist: True (il·lustració 6)
2. Hflips: False i Random Dist: False (il·lustració 7)
3. Hflips: False i Random dist: True (il·lustració 8)
4. Hflips: True i Random Dist: False (il·lustració 9)

És a dir que en el model Anna_net s'ha utilitzat el més òptim que és aquest últim, Hflips a True i Random Dist a False.