d

UNIVERSITAT DE VIC
UNIVERSITAT CENTRAL
DE CATALUNYA

Enginyeria Mecatrònica

Detecció d'ous en mal estat mitjançant Intel·ligència Artificial

Erick Brito Paz

Vic, Gener 2022

Treball Fi de Grau

Grau en Enginyeria Mecatrònica

Detecció d'ous en mal estat mitjançant Intel·ligència Artificial

Autor:

Erick Alexander Brito Paz

Tutor:

Moises Garín Escrivá

Universitat de Vic – Universitat Central de Catalunya

Facultat de Ciències i Tecnologia

Catalunya 2021-2022

Agraïments

A punt de escriure les ultimes línies d'aquest treball, i recordant a tota la meva vida universitària, voldria donar les gràcies a tots aquells docents que m'han impulsat d'una o altra forma a descobrir aquest camp i seguir explorant altres. Estic agraït de la oportunitat que el meu tutor Moises Garín m'ha donat per tal d'explorar i col·laborar en aquest projecte posant el meu gra de sorra.

També vull donar les gràcies a totes aquelles persones que directa o indirectament han fet possible que arribi aquest punt, en especial als meus pares i familiars.

Moltes gràcies.

Resum

Títul: Detecció d'ous en mal estat mitjançant Intel·ligència Artificial
Autor: Erick Alexander Brito
Tutor: Moisés Garin Escrivá (UVic)
Data: Gener de 2022
Paraules clau: *Visión Artificial, Xarxa Neuronal*

En aquest treball s'exposa de manera senzilla els diferents conceptes i etapes per tal de poder construir una xarxa neuronal bàsica amb ajuda de Tensorflow i Python. S'explica la necessitat i procedència de les primeres xarxes neuronals i la seva repercussió fins a l'actualitat en el nostre àmbit particular. Es fa un breu resum dels camps que intervenen i de la seva repercussió. Es recull ordenadament les diferents etapes que prioritzen l'objectiu d'aquest treball "Detecció i Classificació d'ous en mal estat" tal com el tractament de les imatges, com pot ser el mètode de segmentació per obtenir un ou individual i la posteriors tècniques d'ampliació d'un data set així com els criteris utilitzats per dividir les mostres en bons i dolents . Les diferents operacions que succeeixen dins d'una xarxa neuronal convolucional fins arribar a tenir un model capaç de classificar amb alta precisió l'estat d'un ou. S'exposa de manera clara i senzilla els diferents resultats experimentals obtinguts variant els diferents paràmetres de la xarxa neuronal convolucional per tal de visualitzar com s'escullen les mètriques optimes en el model final.

La intel·ligència artificial i l'ús concretament de les xarxes neuronals ofereix un ampli repertori de solucions a petits problemes quotidians i ens permeten oferir una oportunitat d'optimitzar i millorar els recursos d'una granja avícola la qual es la impulsora d'aquest treball no obstant amb les conclusions finals podem veure que encara hi ha un rang ampli de millora.

¿ Què som les persones sinó màquines molt evolucionades?

(Marvin Minsky)

Abstarct

Title: Detection of spoiled eggs by artificial vision
Author: Erick Alexander Brito
Supervisor: Moisés Garín Escrivá (Uvic)
Date: *January 2022*
Keywords: Artificial Vision, Neuronal Network

This final project degree explains in a simple way the different concepts and stages in order to be able to build a basic neural network with the help of Tensorflow and Python. It explains the need and origin of the first neural networks and their repercussions to date in our particular field. A brief summary of the fields involved and their repercussions is given. It collects in order the different stages that prioritize the objective of this project "Detection and Classification of eggs in bad condition with Artificial Intelligence" such as the treatment of the images, as it can be the method of segmentation to obtain an individual egg and the back techniques of extending a date set as well as the criteria used to divide the samples into good and bad. The different operations that take place within a convolutional neural network to the point of having a model capable of classifying the state of an egg with high precision. The different experimental results obtained by varying the different parameters of the convolutional neural network in order to visualize how the optimal metrics are chosen in the final model are presented in a clear and simple way.

Artificial intelligence and the specific use of neural networks offer a wide range of solutions to small everyday problems and allow us to offer an opportunity to optimize and improve the resources of a poultry farm which is the driving force behind this work however with the final conclusions we can see that there is still a wide range of improvement.

What are we people but highly evolved machines?

(Marvin Minsky)

Índex

Llista de Il·lustracions

Llista de Taules

Taula de Acrònims

ABREVIATURA	***SIGNIFICAT***
ML	Machine Learnig
DL	Deep Learnig
AI	Artificial Intelligence
CNN	Convolution neuronal network
GPU	Graphics processing unit
CPU	Central processing unit
TF	Tensor Flow
API	Aplication programming interface

1. Introducció

1.1 Context del projecte.

A Espanya el 2020 hi havia 1.340 granges de producció d'ous i 47,1 milions de gallines ponedores. El 78% de gallines espanyoles estan allotjades en gàbies condicionades, el 13% en granges a terra, el 8% són gallines camperes i l'1,4% ecològiques. Les granges espanyoles en sistema en gàbia són el 35% del total, les de terra el 19%, les camperes el 32% i les ecològiques el 14%.

La producció d'ous es reparteix per tota la geografia espanyola, encara que destaquen Castella la Manxa i Castella i Lleó com a principals regions productores, amb un 26% i un 16% de cens de ponedores. Els segueixen Aragó, el País Valencià i Catalunya. La producció d'ous el 2020 va ser de 1.256 milions de dotzenes, amb una facturació de 1.154 milions d'euros.

En els darrers 10 anys Espanya exporta entre un 15% i un 20% dels ous de taula que produeix. Les exportacions del gener al desembre del 2020 van sumar unes 138.934 tones d'ous i ovoproductes venudes al mercat intracomunitari i 43.090 es van vendre a països tercers.

La situació sanitària a Espanya i la competitivitat dels nostres preus han estat impulsors importants del comerç exterior d'ous.

Per mantenir aquesta competitivitat la qualitat d'un aliment es molt important i es defineix com el conjunt de característiques que determinen l'acceptació del consumidor per aquest aliment, i en el cas concret de l'ou, les principals propietats que cal controlar són *l'aspecte i la forma de la closca*, *color del rovell* i *aspecte de la clara*. La qualitat física de l'ou és un aspecte rellevant per a tots els implicats de la cadena de producció de l'ou, des dels productors, passant pels distribuïdors fins al consumidor final.

1.1.1 Qualitat d'un ou.

Avui en dia el sector avícola conta amb diverses innovacions tecnològiques que permeten augmentar la producció i reduint el temps, això implica una alta productivitat i per tant majors beneficis econòmics. Un altre aspecte important es la qualitat dels ous produïts on gracies a diferents sistemes intel·ligents es poden classificar durant la producció. A continuació es detallen els factors implicats en la qualitat d'un ou.

1.1.2 Factors que determinen la qualitat de la closca.

Les principals alteracions a la closca que comporten pèrdues econòmiques a la indústria de l'ou són: trencaments, fissures, deformació i microfractures.

- **Genètica:** Es tracta d'un aspecte on el productor té poca implicació. Les estirps comercials se seleccionen genèticament en base al % posada i la qualitat de l'ou, de manera que ja venen acompanyades de millores en la qualitat de la closca.
- **Edat dels animals:** La qualitat de la closca es redueix amb l'edat de la ponedora (Roberts i Ball, 2004), ja que en incrementar-se la mida de l'ou, el pes de la closca es manté constant, o fins i tot disminueix, reduint sensiblement el % de closca, la qual cosa condueix a una fragilitat més gran.
- **Nutrició:** Es tracta d'un dels aspectes més importants, sobre els quals pot actuar el productor.
- **Sistema de producció.**
- **Estres.**

1.1.3 Factors que determinen la qualitat interna de l'ou.

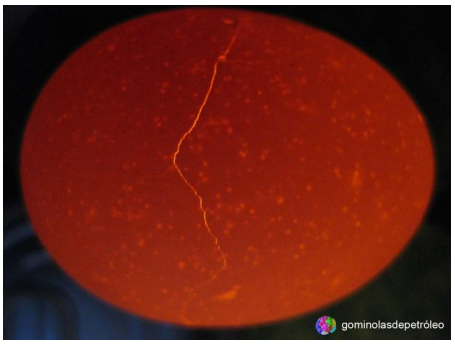
- **Gruix de l'àlbum:** Està influenciat per factors genètics, nutricionals (macro-ingredients, suplements i ració), sanitaris, l'edat de les ponedores i l'emmagatzematge de l'ou.

- **Qualitat de l'àlbum (unitats Haugh):**
  - **Temps i temperatura d'emmagatzematge**
  - **Edat de la gallina:** conforme envelleix, la viscositat de l'àlbum baixa.
  - **Nutrició**: L'increment en contingut proteic redueix les unitats Haugh, mentre que l'addició de vitamina C i E millora aquest paràmetre.
- **Qualitat de la membrana perivitel·lina**: varia en funció de l'edat.

Al llarg dels anys s'han desenvolupat diferents mètodes per quantificar els canvis associats a l'edat de la gallina, les condicions d'emmagatzematge i maneig, les alteracions genètiques, les instal·lacions i les condicions ambientals.

Des de 1949 es realitza el control de qualitat manual per Ovoscòpia [**Figura 1**], ja que es tracta d'una tècnica no invasiva que permet comprovar les característiques externes i internes de l'ou, podent detectar defectes del rovell, àlbum i closca. El mètode es basa en l'efecte duna font lluminosa concentrada en un ambient fosc, que permet determinar la mida/profunditat de la cambra d'aire; tanmateix, és una tècnica subjectiva, ja que els resultats depenen de la percepció del classificador.

Il·lustració 1: *Ovoscòpia, mètode de qualitat*



Il·lustració 2: *Ou amb una esquerda.*

1.1.4 Tècniques de detecció de imperfeccions als ous

En l'actualitat algunes de les classificadores d'ous estan equipades amb detecció automàtica d'esquerdes, però encara hi ha una part important del mercat que depèn que les persones inspeccionin manualment els ous. Se sap des de fa molt de temps que si els ous s'envien a través d'un transportador de rodets retro il·luminat i les persones que observen el flux d'ous es troben en un entorn relativament fosc ("cabina d'inspecció al trasllus"), és possible identificar-ne moltes de les esquerdes als ous. Als ous blancs també es poden observar taques de sang. Durant moltes dècades, aquesta va ser l'única opció per trobar anomalies a les estacions d'envasament d'ous. A mesura que augmentaven els volums, això es tornava cada vegada més tediós i ardu per als treballadors.

Després de molts estudis, es va calcular que un treballador no pot identificar més d'aproximadament el 50% de les esquerdes. Amb l'augment de la capacitat de les màquines al llarg dels anys, aquest nombre indubtablement està disminuint. És per això que al llarg dels anys, a partir de principis de la dècada de 1980 , tots els fabricants de màquines classificadores d'ous van desenvolupar la detecció automàtica d'esquerdes. Essencialment existeixen dos mètodes per detectar si un ou està trencat: mitjançant tècniques de visió artificial, emulant la tasca d'un operari i les tècniques acústiques. La tecnologia de la visió encara era immadura en aquells dies i l'única forma raonable d'identificar una esquerda a la closca d'un ou era "escoltant".

Les tècniques acústiques es basen en fer vibrar la closca. Una closca d'ou intacta vibrarà quan es dona un breu cop, com una copa de vi quan la toques amb el dit. Si hi ha una esquerda a la closca de l'ou, no importa com sigui de petita, l'energia del cop s'absorbirà, cosa que resultarà en un so més apagat. Igual que quan es trenca una copa de vi, no la sentiràs vibrar. Bàsicament, hi ha dos mètodes per crear l'impacte: un és fent que una sonda impacti activament l'ou i l'alternativa és fer que un ou impacti una sonda. Tots

dos tenen diferents avantatges i desavantatges. Una sonda activa necessita més tecnologia per crear l'impacte: parts mòbils que s'acosten a la closca de l'ou. Quan els ous es veuen obligats a rodar sobre un sensor, el sensor està fix i no es mou. Això fa que la tecnologia sigui més simple però també més vulnerable a embrutar-se, ja que l'equip sensible està muntat directament sota el flux d'ous. A més, l'efecte de "bolcada" depèn de la velocitat de la màquina i, per tant, també es veu influït per l'arrencada i la parada.

Respecte als sistemes que utilitzen tecnologia de visió. La visió és ideal per detectar fuites i esquerdes grans, però encara que els sistemes s'estan tornant cada cop més sofisticats, trobar esquerdes fines és un desafiament per a aquesta tecnologia ja que a vegades ni l'ull humà es capaç de detectar aquestes esquerdes tan petites i fines. La visió artificial presenta altres avantatges, com és ser un mètode sense contacte i que permet detectar altres defectes dels ous, com ara taques de excrements.

De tots els sistemes del mercat actual, els "sistemes de sonda d'accionament actiu" obtenen la millor puntuació en rendiment, per sobre del 90%, seguits pels sistemes de "bolcada" amb aprox. 70–80%. Només hi ha uns quants sistemes de visió al mercat que afirmen realitzar una detecció completa d'esquerdes i no obtenen una puntuació millor que el nivell de detecció del 50% dels treballadors encarregats de fer una inspecció.

2. Motivació

La motivació personal per dur a terme aquest projecte és la meva inquietud en aquest camp i l'àmplia aplicació i repercussió que dia a dia té en les nostres vides. Gràcies a aquest projecte, podré d'alguna manera ser capaç d'entendre de manera general el funcionament d'una Xarxa Neuronal i com enfocar-la a la solució d'un problema en particular. La realització d'aquest treball de fi de grau no només té com a finalitat donar solució a un problema en particular sinó també ser la guia per qualsevol persona que vulgui endinsar-se en aquest camp. D'aquesta manera aquest projecte és el punt de partida per tal d'algun dia aprofundir molt més, ja sigui, a nivell educatiu o professional.

La motivació principal d'aquest projecte es poden aconseguir desenvolupar una xarxa neuronal capaç de detectar ous en mal estat mitjançant la visió per ordinador en temps real. Per dur a terme aquest projecte és seguiran diferents processos, com la creació d'un dataset o el previ tractament de les imatges dels ous mitjançant **image processing**. Com es pot observar farem un recorregut ràpid per tots els camps relacionat amb la **I.A** i donar aquest treball de fi de grau un punt de partida generalista per qualsevol altre projecte i per tant la possibilitat de millorar-lo en qualsevol aspecte en un futur.

3. Objectius

El objectiu principal d'aquest projecte es crear un sistema intel·ligent capaç de detectar imperfeccions als ous mitjançant l'ajuda de Xarxes Neuronals Convolucionals. El podem dividir en dos parts:

Software:

- Segmentació i tractament de imatges de ous
- Creació i ampliació d'un data set
- Utilització de la llibreria de TensorFlow amb Keras
- Desenvolupament de una prototip de Xarxa Neuronal funcional
- Implementació i resultats amb noves imatges

Hardware:

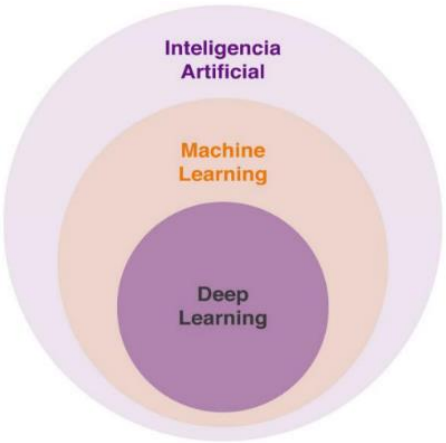
En aquesta part s'involucra tots aquells dispositius que ens permetran la implementació de la xarxa neuronal, com pot ser la càmera per captar les imatges en temps real o el sistema encastat encarregat del executar el programa de predicció.

4. Historia de l'art

Les noves tecnologies, i la intel·ligència artificial en particular, estan canviant de manera dràstica la naturalesa dels processos creatius. Els ordinadors tenen papers molt significatius en processos creadors com la música, l'arquitectura, les belles arts i la ciència. De fet, l'ordinador ja és un llenç, un pinzell, un instrument musical, etc. Tot i així, creiem que hem d'aspirar a relacions més ambicioses entre ordinadors i creativitat. En lloc de considerar l'ordinador com a eina d'ajuda als creadors humans, el podríem veure com una entitat creativa en si mateixa. Aquest punt de vista ha donat lloc a un nou subcamp de la intel·ligència artificial anomenat aprenentatge profund. Per compondre la evolució i el concepte d'intel·ligència artificial i com a partir d'aquí es pot arribar a una xarxa neuronal , farem un breu resum d'aquest conceptes.

4.1 La intel·ligència Artificial

La intel·ligència artificial fa referència a l'habilitat d'una màquina de presentar les mateixes capacitats cognitives que els éssers humans, com ara el raonament, l'aprenentatge, la creativitat i la capacitat de planejar. La IA permet que els sistemes tecnològics percebin el seu entorn, s'hi relacionin, resolguin problemes i actuïn amb una finalitat específica. La màquina rep dades (ja preparades o recopilades a través dels seus propis sensors, per exemple, una càmera), les processa i respon a ells. Els sistemes d'IA són capaços d'adaptar el seu comportament en certa manera, analitzar els efectes d'accions prèvies i de treballar de manera autònoma.



Il·lustració 3: *Descripció dels camps dins d'intel·ligència artificial*

4.2 Machine learning

El Machine Learning és un mètode d'anàlisi de dades que automatitza la construcció de models analítics. És una branca de la intel·ligència artificial basada en la idea que els sistemes poden aprendre de dades, identificar patrons i prendre decisions amb una mínima intervenció humana. Exemples de **ML** podria ser un sistema que podria predir si un estudiant reprovarà o aprovarà un examen aprenent de l'historial de resultats i atributs dels estudiants. Aquí, el sistema no està codificat amb una llista completa de totes les regles possibles que poden decidir si un estudiant aprovarà o reprovarà; per contra, el sistema aprèn per si mateix basant-se en els patrons que va aprendre de les dades històriques.

4.3 Deep learning

És un tipus de machine learning que entrena un ordinador perquè executi tasques com les que fem els éssers humans, com el reconeixement de la parla, la identificació d'imatges o fer prediccions. En lloc d'organitzar dades perquè s'executin mitjançant equacions redefinides, deep learning configura paràmetres bàsics sobre les dades i entrena l'ordinador perquè aprengui per compte propi reconeixent patrons mitjançant l'ús de moltes capes de processament.

4.4 Recursos per el Deep Learning

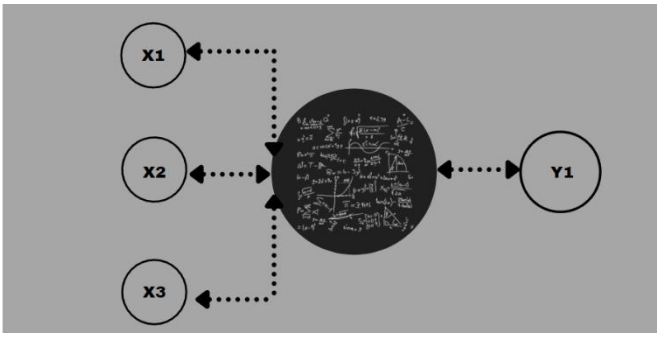
TensorFlow és, sens dubte, un dels marcs de DL més populars i àmpliament utilitzats a la fraternitat. De codi obert, va ser desenvolupat per Google i admet el desplegament en CPU, GPU i dispositius mòbils i perifèrics. Va ser llançat el novembre de 2015 i després va veure un gran augment en la seva adopció a la indústria.
TensorFlow és una biblioteca d'intel·ligència artificial de codi obert, que fa servir gràfics de flux de dades per construir models. Permet als desenvolupadors crear xarxes neuronals a gran escala amb moltes capes. És fa servir principalment per a: classificació, predicció.

Keras és una poderosa biblioteca de Deep learning que s'executa a sobre d'altres biblioteques d'aprenentatge automàtic de codi obert com TensorFlow i també és de codi obert. Per desenvolupar models d'aprenentatge profund. Keras adopta una estructura mínima a Python que fa que sigui més fàcil d'aprendre i ràpid d'escriure. És escalable i fàcil d'utilitzar. Els models d'aprenentatge que crea Keras són components discrets, el que significa que es poden combinar de moltes maneres. Keras també admet xarxes recurrents i xarxes de convolució. Les xarxes neuronals també estan escrites en Python i tenen una gran comunitat de suport.

Aquestes són llibreries que utilitzarem per poder construir la nostre xarxa neuronal, però abans definirem per sobre tots els detalls que la componen i que la defineixen com a tal.

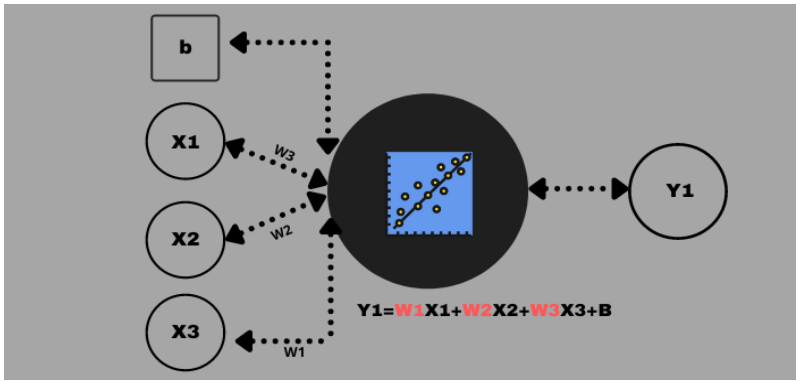
5. Xarxa Neuronal

Per poder començar a entendre que és una xarxa neuronal, primer hem d'entendre la unitat bàsica que la conforma. Fent referència al propi sistema nerviós de l'ésser humà la unitat bàsica és la neurona, com és pot veure a la **il·lustració 4**. Aquesta pot tenir diverses entrades **,**$X_1 X_2 X_3 X_n$**.** Aquestes dades d'entrada poden provenir d'una base de dades o de prèvies operacions fetes per altres neurones. Cada entrada té un pes($\boldsymbol{\omega_1, \omega_2, \omega_3, \omega_n}$) assignat segons la importància que té dins la neurona. Dins d'aquesta unitat de processament és du a terme una suma ponderada.



Il·lustració 4: *Representació bàsica de les connexions d'una sola neurona.*

Per explicar de manera senzilla aquesta operació, podem observar que aquesta equació de la **il·lustració 5** te la forma d'una equació de regressió lineal. Així podem dir que internament una neurona, en termes generals, fa una regressió lineal de les dades d'entrada donant com a resultat una sortida $\boldsymbol{y_1}$**.** Aquesta sortida no només depèn de les dades d'entrada i del pes associat, aquestes també depenen d'una variable independent, **b**, anomenada (**Bias**).



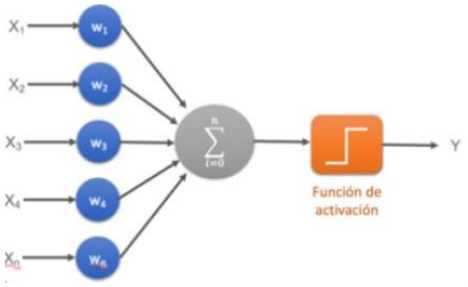
Il·lustració 5: *Representació bàsica de les operacions que intervenen dins d'una neurona.*

5.1 Classificació de xarxes neuronals artificials

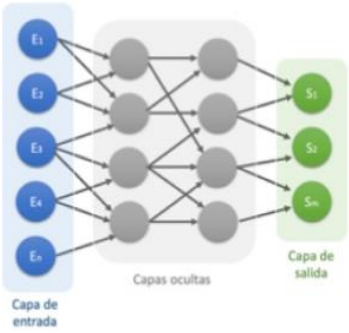
La primera classificació es pot fer a partir de la quantitat de capes que te la xarxa i la manera que té aquesta de processar les dades:

- **Xarxa Neuronal Monocapa – Perceptró Simple** : La xarxa neuronal monocapa es correspon amb la xarxa neuronal més simple, està composta per una capa de neurones que projecten les entrades a una capa de neurones de sortida on es fan els diferents càlculs.

- **Xarxa Neuronal Multicapa – Perceptró Multicapa** : La xarxa neuronal multicapa és una generalització de la xarxa neuronal monocapa, la diferència és que mentre la xarxa neuronal monocapa està composta per una capa de neurones d'entrada i una capa de neurones de sortida, aquesta disposa d'un conjunt de capes intermèdies (capes ocultes) entre la capa d'entrada i la de sortida.



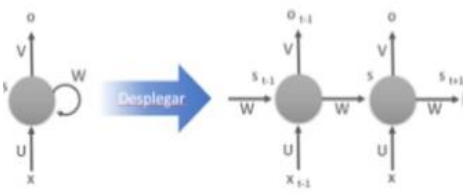
Il·lustració 6: *Perceptró monocapa*



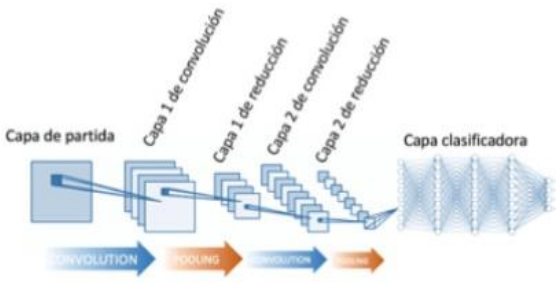
Il·lustració 7: *Perceptró multicapa*

- **Xarxa Neuronal Convolucional (CNN)** : Aquest tipus de xarxa és una variació d'un perceptró multicapa, però pel fet que la seva aplicació és realitzada en matrius bidimensionals, són molt efectives per a tasques de visió artificial, com en la classificació i segmentació d'imatges, entre d'altres aplicacions. Sobre aquest tipus aprofundirem més endavant donat que es el triat per construir el nostre classificador d'ous.

- **Xarxa Neuronal Recurrent (RNN):** Una capa de neurones recurrents es pot implementar de manera que, a cada instant de temps, cada neurona rep dues entrades, l'entrada corresponent de la capa anterior i alhora la sortida de l'instant anterior de la mateixa capa.



Il·lustració 9: *Xarxa neuronal recurrent*



Il·lustració 8: *Xarxa neuronal convolucional*

La segona classificació es pot fer segon el mètode d'aprenentatge:

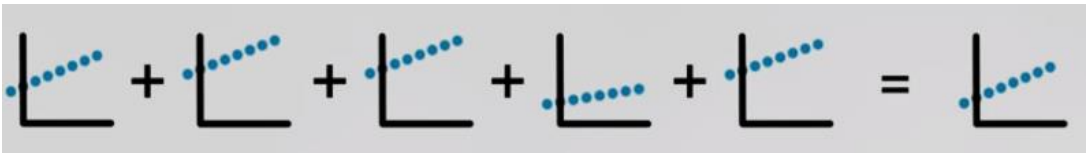
- **Aprenentatge supervisat:** Es caracteritza perquè el procés d'aprenentatge es fa mitjançant un entrenament controlat per un supervisor que determina la resposta que cal generar per a cada entrada. El supervisor controla la sortida i si aquesta no és correcta, modifica els pesos de les connexions, per tal que la sortida obtinguda s'aproximi a la desitjada.

- **Aprenentatge no supervisat:** Es caracteritza perquè no requereix cap influencia externa per modificar els pesos de les neurones. Aquest tipus d'aprenentatge cerca trobar les característiques, regularitats, correlacions o categories que es puguin establir entre les dades que es presentin com a entrada. La interpretació de les dades depèn de la seva estructura i de l'algoritme d'aprenentatge emprat. La sortida podia representar el grau de similitud entre les dades, un clustering o establiment de categories.

- **Aprenentatge per reforç:** Es considera un aprenentatge més lent que l'aprenentatge per correcció d'errors, en aquest cas no es disposa d'un conjunt complet de les dades exactes de sortida sinó que només s'indica si la dada és acceptable o no, amb això l'algorisme ajusta els pesos basant-se en un mecanisme de probabilitats.

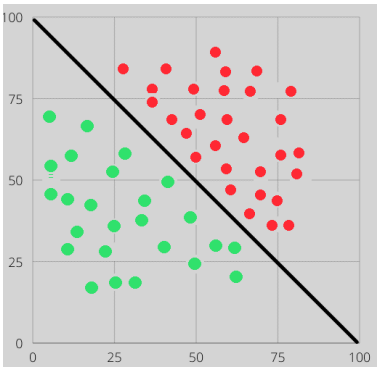
5.2 Perceptró Monocapa vs Perceptró Multicapa

Si analitzem la solució d'unir diverses neurones en una capa , tal com presenta el model perceptró monocapa, veurem que per tal de trobar solucions a problemes complexos tenim una limitació important i es la linealitat que té el nostre sistema.

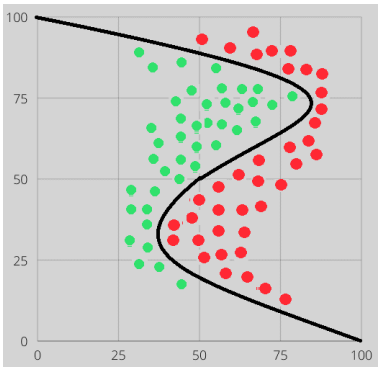


Il·lustració 10: *Representació d'una suma de diferents regressions lineals equivalent a la suma de diferents neurones en el model monocapa.*

En la **il·lustració 12** veiem dos grups definits de dades que és volen separar, vermells i verds, una possible solució es la recta definida amb negre. Però que passa si aquests punts estiguessin distribuïts de manera que cap recta sigui la solució optima al problema**.** Doncs que la solució del perceptró monocapa deixa de ser útil i passem a un nou model més complex, el perceptró multicapa. Que donaria solució a la **il·lustració 11.**



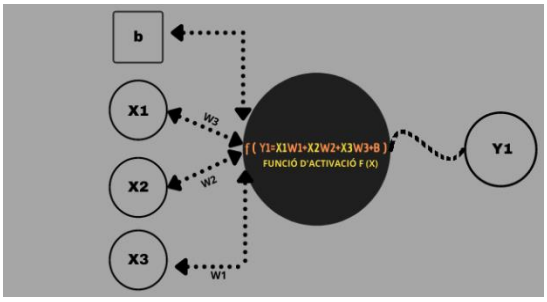
Il·lustració 12: *Representació de dades*



Il·lustració 11*: Representació de dades*

5.3 Funció d'activació

En aquest punt gracies a les funcions de activació que distorsionen la nostre sortida podem obtenir solucions no lineals com la corba de la **il·lustració 13.**



Il·lustració 13: *Representació d'una neurona amb la funció d'activació.*

Tant a les xarxes neuronals artificials com a les biològiques, una neurona no només transmet l'entrada que rep. Hi ha un pas addicional, una funció d'activació, que és anàloga a la taxa de potencial d'acció disparant al cervell. La funció d'activació utilitza la mateixa suma ponderada de l'entrada anterior $z = b + \sum_i w_i x_i$, i la transforma una vegada més com a sortida.

S'han proposat moltes funcions d'activació, però en descriurem només dues en detall: **Sigmoide** i **ReLu.**

La funció **sigmoide** és la funció d'activació més antiga i popular. Definida com:

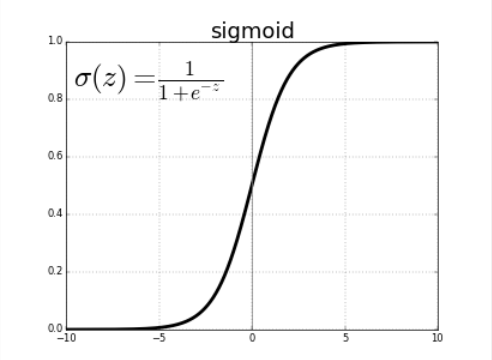
$$\sigma(x) = \frac{1}{1 + e^{-x}} \qquad \textbf{(Eq.1)}$$

e denota la constant d'Euler (o nombre natural), que és aproximadament igual a **2,71828**. Una neurona que utilitza la sigmoide com a funció d'activació s'anomena neurona sigmoide . Primer establim que la variable **z** equival a la nostra suma ponderada d'entrada i després la passem a través de la funció sigmoide.

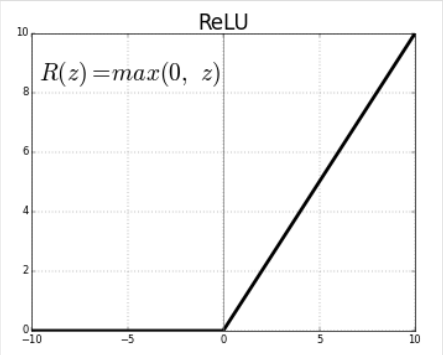
$$z = b + \sum_i w_i x_i \qquad \textbf{(Eq.2)}$$

$$\sigma(z) = \frac{1}{1 + e^{-z}} \qquad \textbf{(Eq.3)}$$

Encara que l'equació sembla complicada i arbitrària, en realitat té una forma força simple. La podem veure si tracem el valor de **σ(z)** com a funció de l'entrada **z**.



Il·lustració 14 : *Representació de la funció Sigmoid*



Il·lustració 15*: Representació de la funció ReLu*

Podem veure que **σ(z)** actua com una mena de funció **"aixafadora",** comprimint la nostra sortida a un rang de **0 a 1**. Al centre, on **z=0**, **σ(0)=1/(1+e^0)= 1/2**. Per a valors negatius grans de z, el terme e^{-z}al denominador creix exponencialment, i σ(z) s'aproxima a **0**. Al contrari, valors positius grans de z redueixen e^{-z}cap a 0, i σ(z) s'aproxima a **1**.

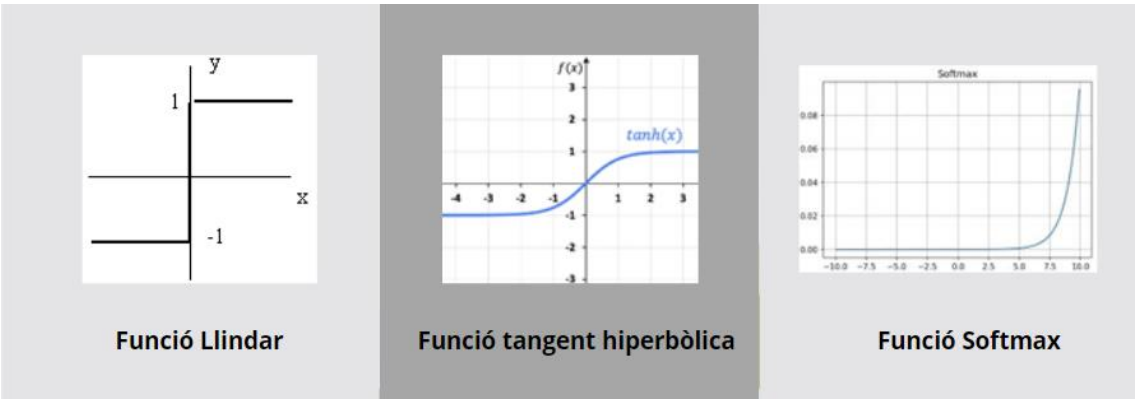
La funció sigmoide és contínuament diferenciable, i la derivada convenientment és σ'(z)=σ(z)(1−σ(z)). Aquest detall indica que hem de fer servir càlcul per entrenar una xarxa neuronal.

Les funcions sigmoides van ser la base de la majoria de les xarxes neuronals durant moltes dècades, encara que en anys recents han perdut popularitat.

En canvi, la majoria de les xarxes neuronals actuals usen un altre tipus de funció d'activació anomenada **rectified linear unit** o **ReLu [il·lustració 15]**. Malgrat el nom complicat, es defineix simplement com a **R(z)=max(0,z).**

En altres paraules, les **ReLu** permeten el pas de tots els valors positius sense canviar-los, però assigna tots els valors negatius a 0. Tot i que hi ha funcions d'activació encara més recents, la majoria de xarxes neuronals d'avui utilitzen ReLu o una de les seves variants.

Altres funcions d'activació:



Il·lustració 16: *Representació d'altres funcions d'activació*

5.4 Funció de pèrdua

La funció de cost o pèrdua intenta determinar l'error entre el valor estimat i el valor real, per tal d'optimitzar els paràmetres de la xarxa neuronal. Aquesta funció ens permet saber que tan bé classifica el nostre model.

Existeixen diferents funcions de cost segons la aplicació del nostre model:

- **Funcions de cost de regressió**: Els models de regressió s'utilitzen per predir una variable contínua, com ara el sou d'un empleat, el cost d'un cotxe, la probabilitat d'obtenir un préstec, etc. Per aquesta classe tenim diferents estimadors.

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}} \quad \textbf{(Eq.4)} \qquad MAE = \sum_{i=1}^{n} \frac{|\hat{y}_i - y_i|}{n} \quad \textbf{(Eq.5)} \qquad MASE = \frac{\sum_{i=1}^{n} |\hat{y}_i - y_i|}{\frac{n}{n-1}\sum_{i=2}^{n} |\hat{y}_n - y_{n-1}|} \quad \textbf{(Eq.6)}$$

- **Funcions de cost de classificació binària:** Les funcions de cost utilitzades en problemes de classificació no són les mateixes que les funcions de cost utilitzades en problemes de regressió. La funció de pèrdua de classificació popular es **cross-estropy.**

$$\mathcal{L}(\theta) = -\frac{1}{n}\sum_{i=1}^{n} y_i \log(p_i + (1-y)\log(1-pi)) \quad \textbf{(Eq.7)}$$

- **Funcions de cost multi-classe:** Els problemes de modelització predictiva que impliquen una categorització multi-classe són aquells en què les instàncies s'assignen a més de dues classes.

$$\mathcal{L}(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m} y_{ij} \log(p_{ij}) \quad \textbf{(Eq.8)}$$

6. Aprenentatge Autònom

Un cop sabem els components que conformen aquesta maquinària el següent pas es saber com funciona per tal de ser capaços de replicar-la, millorar-la o ajustar-la a les nostres necessitats. Per aquesta comanda necessitarem parla de dos processos claus que passen dins de les neurones en el moment que estan entrenant. Aquests processos són **Forward-propagation** i **Backward-propagation.** Per tal de comprendre de manera breu farem descripcions generals sense entrar-hi massa en les matemàtiques que hi ha darrere aquests processos.

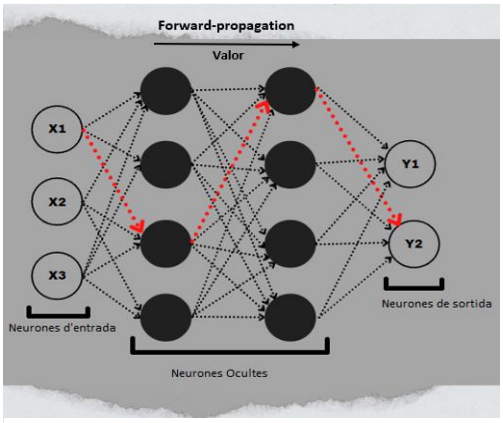
- **Forward-propagation**

La propagació cap endavant és el conjunt de processos matemàtics des que introduïm les nostres dades a la xarxa neuronal, fins que la xarxa neuronal dona un resultat. Les operacions es van fent d'esquerra a dreta, és a dir de la capa d'entrada, van a la primera capa oculta, després van a la segona capa oculta i finalment van a la darrera capa que ens dona el resultat. Aquesta direcció establerta pel tractament de les dades s'anomena Forward-propagation, tal com veiem a la **il·lustració 18.**

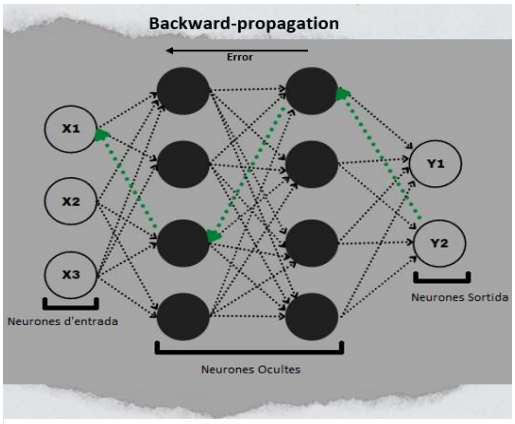
- **Backward-propagation**

La propagació enrere detecta les ponderacions correctes que cal aplicar als nodes d'una xarxa neuronal mitjançant la comparació de les sortides actuals de la xarxa amb els resultats correctes o desitjats. Com hem dit en apartats anteriors cada neurona té un pes que li dona la seva importància dins de la xarxa i determinant en el resultat d'una predicció, aquest pes en una primera iteració s'assigna un valor aleatori i amb el procés de backward-propagation ajustem aquest pesos per tal de minimitzar l'error en cada iteració i obtenir un resultat òptim (**il·lustració 17**).

La diferència entre el resultat desitjat i el resultat actual es calcula mitjançant la **funció de pèrdua o cost**. En altres paraules, la funció de pèrdua ens indica el grau de precisió que té la nostra xarxa neuronal en fer prediccions per a una entrada determinada.



Il·lustració 18: *Representació del procés de Forward-propagation*



Il·lustració 17: *Representació del procés de Backward-propagation*

Podem fer un resum dels diversos processos que passa dins d'una xarxa neuronal:

- La xarxa neuronal dona un **resultat** gracies al procés de **Forward-propagation**.
- Amb aquest resultat s'aplica un **funció d'error**. Aquesta funció calcula la quantitat d'**error** que hi ha entre el **resultat predit** i el **resultat real.**

$$\delta_1^{(1)} = \omega_1 * \delta_1^{(2)} * J'(a_{11}) \quad \textbf{(Eq.9)}$$

$\boldsymbol{\delta_1^{(1)}}$: Error de la neurona actual.

ω : pes de la neurona.

$\boldsymbol{\delta_1^{(2)}}$: Error de l'anterior neurona

$\boldsymbol{J'(a_{11})}$: Producte Hadamart.

- Seguidament comença el procés de **backward-propagation,** que ens permet enviar l'error cap a endarrere per tal d'actualitzar el pesos de cada neurona i per tant millorar la predicció de la nostre xarxa.

$$\omega = \omega - \alpha * \delta * \emptyset \quad \textbf{(Eq. 10)}$$

ω : pes de la neurona

α : ràtio d'aprenentatge

δ : error

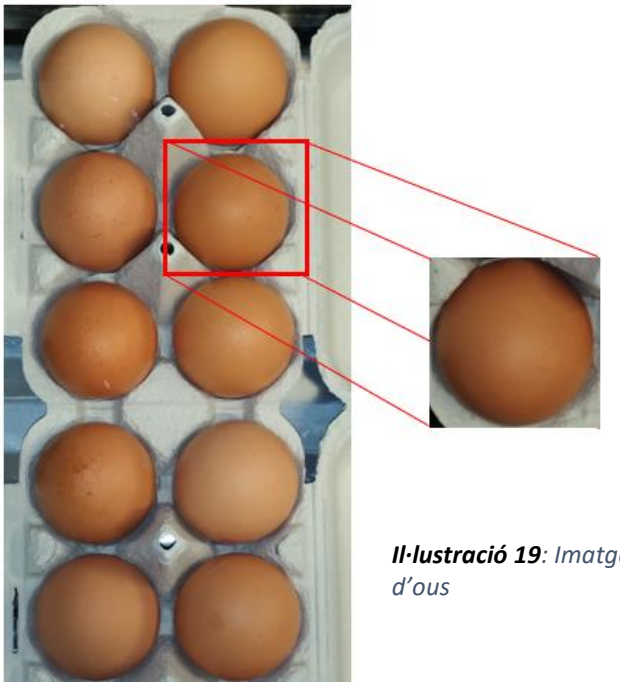
Φ : input

7. Desenvolupament

7.1 Dataset

Una de les parts més importants a l'hora de modelitzar un classificador amb l'ajuda de Xarxes Neuronals és tenir un ampli dataset. Un Dataset no és més que un conjunt de dades tabulades a qualsevol sistema d'emmagatzematge de dades estructurades. El terme fa referència a una única base de dades d'origen, la qual es pot relacionar amb altres, cada columna del Dataset representa una variable i cada fila correspon a qualsevol dada que estiguem tractant, això s'aplica a l'hora de tenir dades numèriques, d'altra banda una dataset també pot ser només un conjunt de fotos com ous, animals, objectes, etc..

En els següents apartats podrem veure com a partir d'una imatge d'un conjunt d'ous **[Figura 19]** podem obtenir una imatge individual **[Figura 20]** i d'aquesta manera poder construir el nostre dataset **[Figura 20]** .



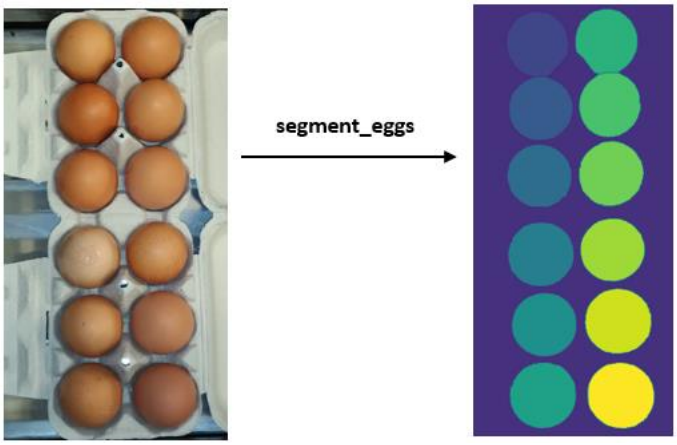
***Il·lustració 19**: Imatge d'un conjunt d'ous*

***Il·lustració 20:** Imatges de diferents ous extretes a partir d'un conjunt amb diverses característiques .*

7.2 Processat d'imatge

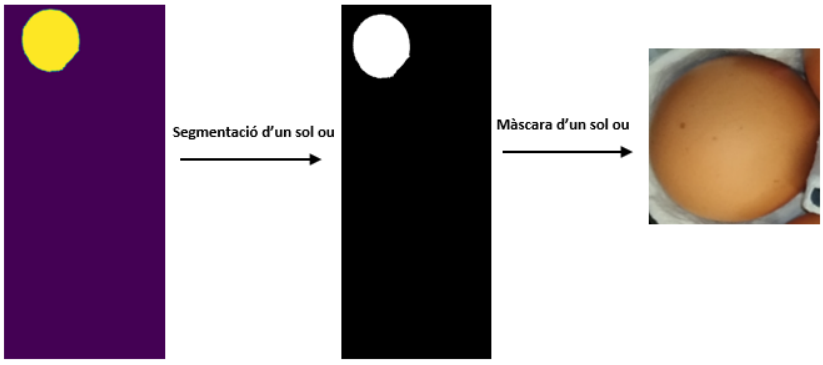
En aquest apartat s'explica de manera més clara com a partir d'una imatge d'un conjunt d'ous **[Figura 19]** podem obtenir una imatge única d'un ou **[Figura20].** Per tal de generalitzar no entrarem de manera densa en el codi que proporciona aquests resultats sinó en l'idea que hi ha darrera del procés.

Si partim de la imatge que ens proporciona la granja del conjunt d'ous, la nostra primera tasca es poder identificar de manera individual cada ou. Per aquest procés mitjançant K-means i els centres relatius de cada ou podem etiquetar-los. Cada color es una etiqueta (1,2,3) diferent que ens permet identificar en la imatge els píxels corresponents a cada ou. El mètode '**segment_eggs'** desenvolupat al codi, dur a terme aquest procés.



Il·lustració 21: *Segmentació d'un conjunt d'us mitjançant k-means*

Després podem separa cada ou mitjançant l'etiqueta i crear una mascara **[Figura 22]** per poder obtenir la imatge final **,** ja que es aquesta la que processa la Xarxa Neuronal.



Il·lustració 22: *Extracció d'un ou a partir d'un conjunt*

7.3 Classificació dels ous

En el apartat anterior parlem de la separació de cada ou a partir d'un conjunt . Arribat aquest punt el nostre objectiu es identificar aquells ous que, segons el criteri de qualitat establerta per la granja, son bons o dolents. Aquesta tasca l'hem de fer nosaltres ja que la Xarxa Neuronal prèviament ha de saber quines característiques pertanyen a un ou en "bon estat" i quines a un ou en "mal estat". A partir de una imatge d'un conjunt d'ous [**Figura 18**] i a la elaboració d'un registre d'ous dolents podem classificar-los.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Nombre archivo	filas	columnas	h1	h2	h3	h4	h5	h6	h7	h8	h9	h10	h11	h12
2	IMG_20210309_095911.jpg	6	2	0	0	0	0	0	0	0	1	0	0	0	0
3	IMG_20210309_095918.jpg	6	2	0	0	0	0	0	0	0	1	0	0	0	0
4	IMG_20210309_100017.jpg	6	2	1	1	0	0	0	0	0	0	0	0	0	0
5	IMG_20210309_100119.jpg	6	2	0	0	0	0	0	1	0	0	0	0	0	0
6	IMG_20210309_100133.jpg	6	2	0	0	0	0	0	1	0	0	0	0	0	0
7	IMG_20210309_100205.jpg	6	2	0	0	1	0	0	0	0	0	0	0	0	0
8	IMG_20210309_100440.jpg	6	2	0	0	1	0	0	0	0	0	0	0	0	0
9	IMG_20210309_100624.jpg	6	2	0	0	0	0	0	1	0	0	0	0	0	0
10	IMG_20210309_100818.jpg	6	2	0	1	0	0	0	0	0	0	0	0	0	0
11	IMG_20210309_101102.jpg	6	2	0	0	0	0	0	0	0	0	0	0	0	1
12	IMG_20210309_101313.jpg	6	2	0	0	0	1	0	0	0	0	0	0	0	0

Il·lustració 23: *Imatge d'Excel*

A la taula [**Il·lustració 23**] podem observar que, a la primera columna consta el nom de la imatge, a la segona i tercera consta el nombre de files i columnes que pot tenir aquest conjunt d'ous, seguidament per cada posició dins del conjunt posem un "0" si es un ou bo i un "1" si es dolent. D'aquesta manera a l'hora que processem cada conjunt d'ous podem fer la classificació. Així podem tenir dos directoris per cada tipus. D'aquesta manera quan obtenim la imatge final de l'ou **[Figura 23]** ja sabem a quin directori pertany.

Bons

Dolents

Nota:

Quan parlem de "processat" fem referència al conjunt de tècniques que utilitzem per tal de dur aquesta tasca. En el nostre cas, tenim un script que identifica i etiqueta cada a ou del conjunt total (egg_detector.py) i al mateix temps els classifica gràcies a un altre script (Excel_data.py) que verifica que tipus d'ou es. Cada script està degudament comentat per tal d'entendre cada línia de codi.

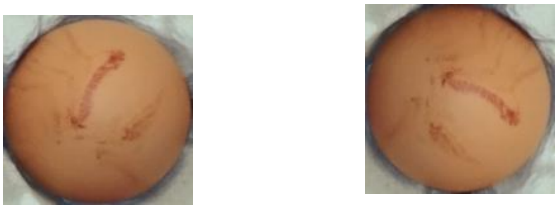
7.4 Data Augmentation

Aquest concepte de "data augmentation" fa referència als diferents mètodes que tenim per ampliar el nostre conjunt de dades de manera artificial. Pot passar, com és el nostre cas, que el nostre dataset no contingui imatges suficients per a començar a entrenar la nostra Xarxa Neuronal, per aquesta raó a partir de les imatges existents i aplicant mètodes com rotació o desplaçaments podem obtenir moltes més. D'altra banda aquestes modificacions permeten que la Xarxa Neuronal sigui robusta aquest tipus de canvis i pugui aprendre millor.

- **Flip:** Aquest mètode ens permet obtenir una rotació amb efecte mirall

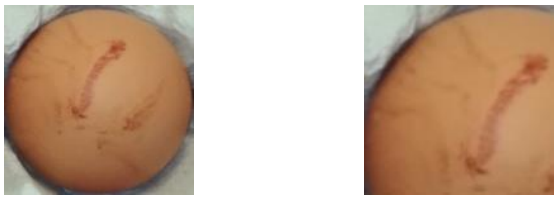
Il·lustració 24: *Imatge original (esquerra) i imatge després d'un flip (dreta)*

- **Rotation:** Aquest mètode ens permet rotar la imatge a diferents graus.



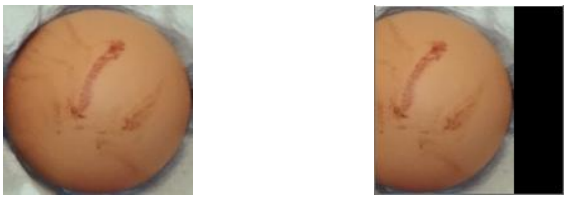
Il·lustració 25*: Imatge original (esquerra) i imatge després d'una rotació (dreta)*

- **Crop:** Aquest mètode ens permet obtenir una mostra aleatòria de la imatge original



Il·lustració 26: *Imatge original (esquerra) i imatge després d'un crop (dreta)*

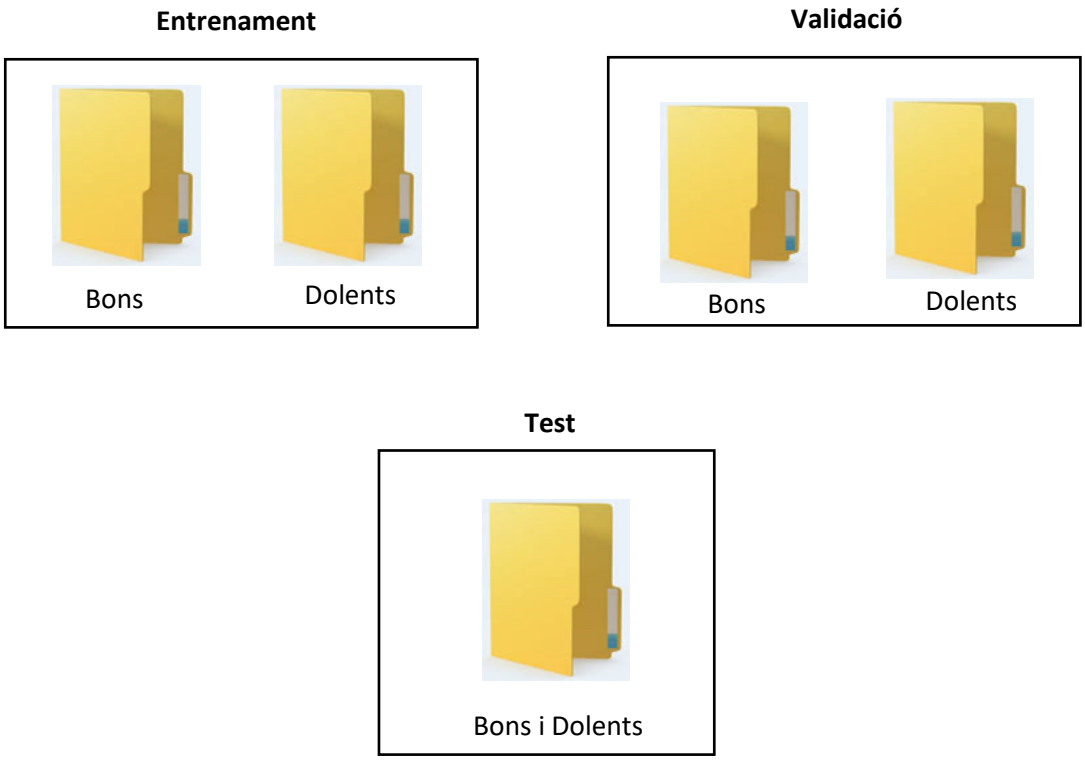
- **Shift:** Es un mètode que ens permet desplaçar els píxels de la imatge.



Il·lustració 27: *Imatge original (esquerra) i imatge després d'un shift horitzontal (dreta)*

7.5 Estructura del data set

Arribat aquest punt es important saber com hem de estructura el nostre data set per tal de entrenar a la Xarxa Neuronal. Primer hem de entendre que una part de les dades han de servir d'entrenament 60-70% , un altre part de validació 20-30% i finalment, per tal de comprovar que tan bona es la predicció de la xarxa, imatges de testing 10%.



Cal mencionar que no es l'única forma d'estructura un dataset ja que segons el llenguatge i les llibreries o Framworks que s'utilitzi pot ser més senzill o mes laboriós. En el nostre cas utilitzem TensorFlow que ens proporciona diferents mètodes que ajuden a simplificar aquest procés amb l'estructura mencionada.

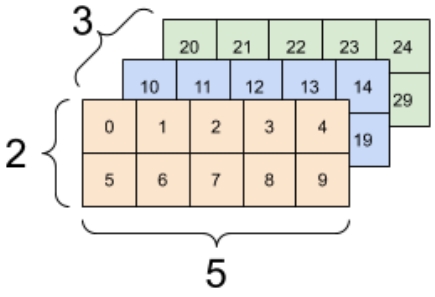
8. Model Convolucional

En capítols anteriors hem parlat sobre que era una Xarxa Neuronal i com funciona de manera teòrica. En aquest capítol agafarem aquestes idees i veurem de manera estructurada que és una Xarxa Neuronal convolucional i l'estructura de codi que hem utilitzat.

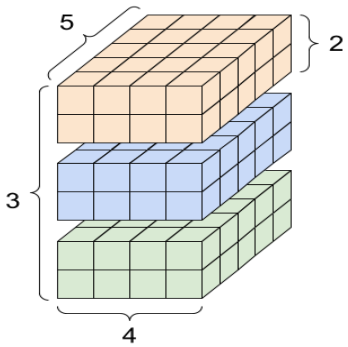
Les xarxes neuronals convolucionals són molt similars a les xarxes neuronals que descrivim en els capítols anteriors. Estan formades per neurones que tenen paràmetres en forma de pesos i biaixos que es poden modificar a mesura que la xarxa entrena. Però un tret diferencial de la **CNN** és que fan la suposició explícita que les entrades són imatges, cosa que ens permet codificar certes propietats a l'arquitectura per reconèixer elements concrets a les imatges.

8.1 Dades d'entrada

Les dades d'entrada poden ser de diversos tipus. Essencialment, el model en **TensorFlow** entén les dades com a "tensors". Els tensors no són més que una forma genèrica per a vectors, o en termes d'enginyeria informàtica, una simple matriu n-dimensional. Les dades de qualsevol forma és representen finalment com una matriu numèrica homogènia. Per tant, si les dades són tabulars, serà un tensor bidimensional on cada columna representa una mostra d'entrenament i tota la taula/matriu serà m mostres. Si parlem d'imatges la cosa és una mica més complicada, ja que per tal de definir-la haurem d'optar per un tensor **3-D** per escala de grisos i **4-D** en color [R, G, B].



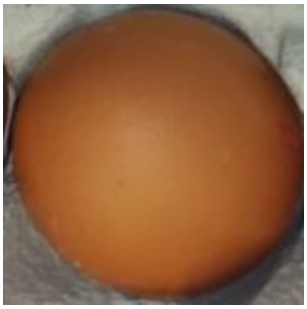
Il·lustració 29: *Tensor 3D*



Il·lustració 28: *Tensor 4D*

En el cas del tensor 3-D les 3 mostres tindrien una mida de 5X2 píxels. En el cas del tensor 4-D tindríem 3 mostres de 5X4 píxels amb 2 canals. Per poder saber una mica més sobre tensors, la pàgina web **TensorFlow** ofereix una guia més extensa. Altres aspectes que hem de tenir present és la mida de les imatges i si seran processades en color o en escala de grisos.

Són aspectes importants que poden influir de manera decisiva a l'hora d'extreure les característiques de cada imatge i, per tant, guanyar o perdre informació.



***Il·lustració 32**: Imatge 150x150*

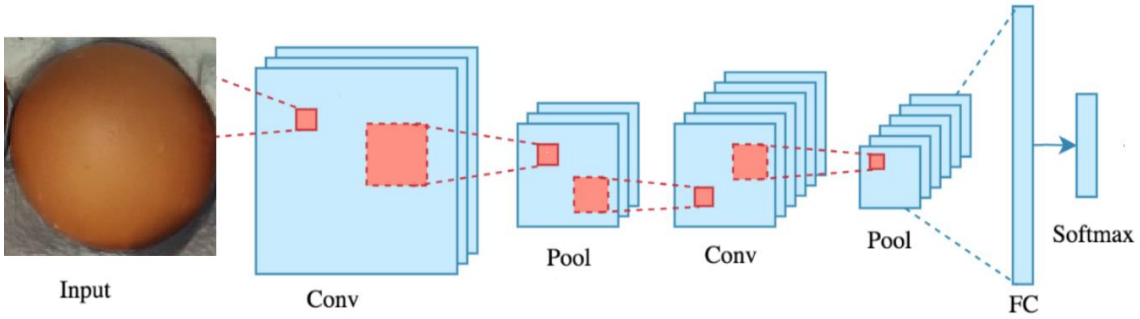
***Il·lustració 31:** Imatge 100x100*

***Il·lustració 30:** Imatge 50x50*

Com es pot observar en les il·lustracions 30-32 la resolució és un aspecte important i hem de ser capaços de triar aquella que sigui òptima entre tenir com menys píxels millor, per tal que el càlcul i el processament de cada imatge sigui òptim, i no perdre cap característica important.

8.2 Processament de les imatges en una Xarxa Neuronal Convolucional

Un cop hem triat les característiques òptimes mencionades en l'apartat anterior, la xarxa procedeix a fer els seus propis càlculs per tal d'extreure característiques dels ous.

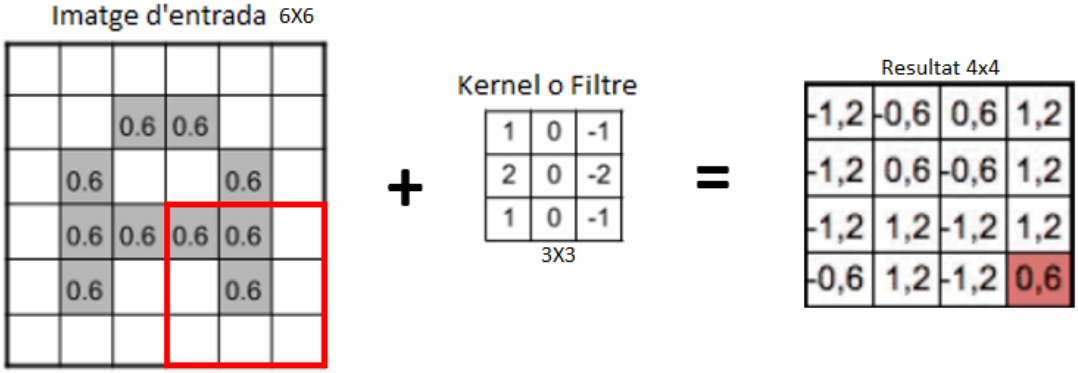


***Il·lustració 33**: Processament d'una imatges dins d'una xarxa Neuronal*

La **Il·lustració 33** mostra les diferents etapes que s'aplica a una imatge d'ou. Cada processament així com les vegades que s'aplica no és estàndard, és a dir, els requeriments de cada Xarxa Neuronal i l'objectiu d'aquesta poden ser diferents dels nostres i, per tant, es poden prescindir d'algunes operacions si es requereix. Seguidament, explicarem cada etapa i la seva importància.

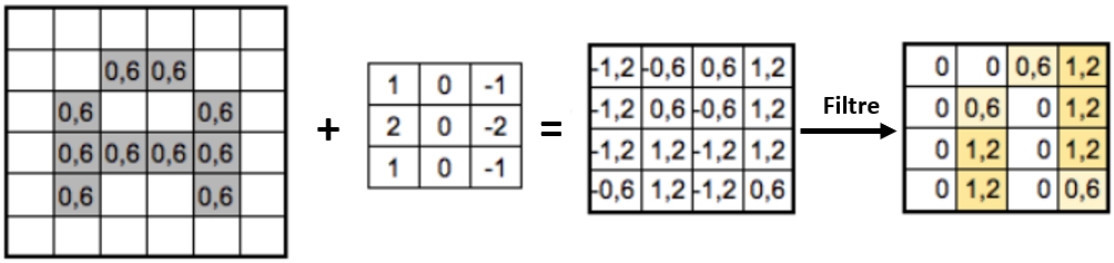
- **Convolucions**

La primera operació que es du a terme és la convolució, aquesta consisteixen a prendre "grups de píxels propers" de la imatge d'entrada i anar operant matemàticament (producte escalar) contra una petita matriu que es diu kernel. Aquest Kernel normalment és de 3X3, però es pot modificar segons la mida de la imatge d'entrada. El kernel recorre cada píxel de la imatge i aplica una operació, el resultat després de recórrer total la imatge és una altra matriu.



Il·lustració 34: *Convolució d'una imatge*

Per exemple, la nostra imatge d'ou amb una mida de 150X150 ,és a dir un total de 22.500 píxels que és igual al nombre de neurones d'entrada, si l'operem amb un filtre o kernel de 3x3 obtindríem una imatge de sortida de 148x148. La realitat és que no s'aplica només un kernel sinó un conjunt de filtres, el nombre de filtres estàndard és de 32, per tant, obtindríem 32 noves imatges de 150x150. Cada imatge nova guarda unes característiques de la imatge d'entrada. A més, es sobtat que després d'aplicar el kernel la matriu resultant tingui la mateixa mida que la original, això és degut a un procés anomenat **padding** que incorpora píxels blancs per tal de tenir la mida original.

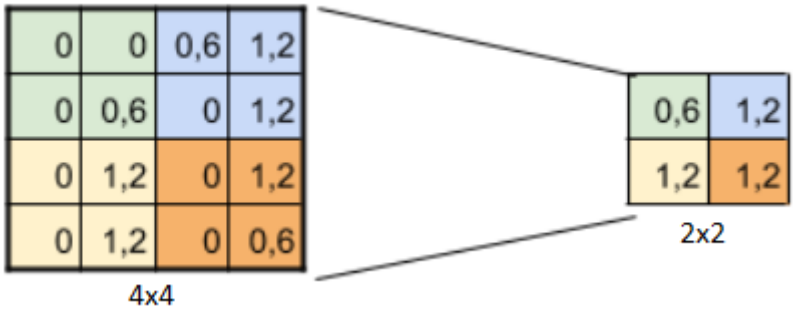


Il·lustració 35: *Convolució i detecció de característiques mitjançant un filtre.*

Aquestes 32 imatges noves son filtrades amb una funció d'activació **[Figura 39]**, això ens permet netejar la imatge i ressaltar les particularitats que el filtre a detectat. Per tant la nostre **primera capa oculta** tindrà 720.000 neurones. Els paràmetres com poden ser la mida del kernel o les vegades que apliquem aquest, no tenen un punt de partida fixe i com hem dit abans solen ser estàndards a l'hora de programar la nostre capa.

- **Max-pooling**

Com hem vist en l'apartat anterior la convolució ens multiplica les neurones, hem passat de tenir 22.500 neurones a 720.000 en una sola operació. Si apliquem una altre convolució per tal d'extreure mes característiques el nombre de neurones seria massa gran per dur a terme qualsevol operació. Arribat aquest punt necessitem reduir aquest nombre sense eliminar informació important. Per tant apliquem la operació de Max-pooling, que es tornar a filtrar la imatge, però aquesta vegada amb un kernel de 2x2, com a resultat guardem el píxel més gran (per això es diu 'Max') i reduïm les imatges a la meitat.



Il·lustració 36: *Max-pooling*

Si tornem al nostre exemple dels ous, amb la primera convolució hem obtingut 32 imatges de 150x150. Amb el max-pooling aquestes 32 imatges tindrien una nova mida de 75x75, això donaria una segona capa oculta de 180.000 neurones, set vegades menys que la capa anterior. Segons les nostres necessitats podem tornar aplicar una segona convolució amb les dades resultants conjuntament amb un max-pooling, aquesta teoria la posem en practica amb Python i TensorFlow.

Entrada de la imatge

```
model.add(Conv2D(32,(3,3),input_shape=input_shape))
model.add(Activation('relu')) #simgoid
model.add(MaxPooling2D(pool_size=(2,2)))
```

1. Convolució + Max-pooling

```
model.add(Conv2D(32,(3,3)))
model.add(Activation('relu')) #simgoid
model.add(MaxPooling2D(pool_size=(2,2)))
```

2. Convolució + Max-pooling

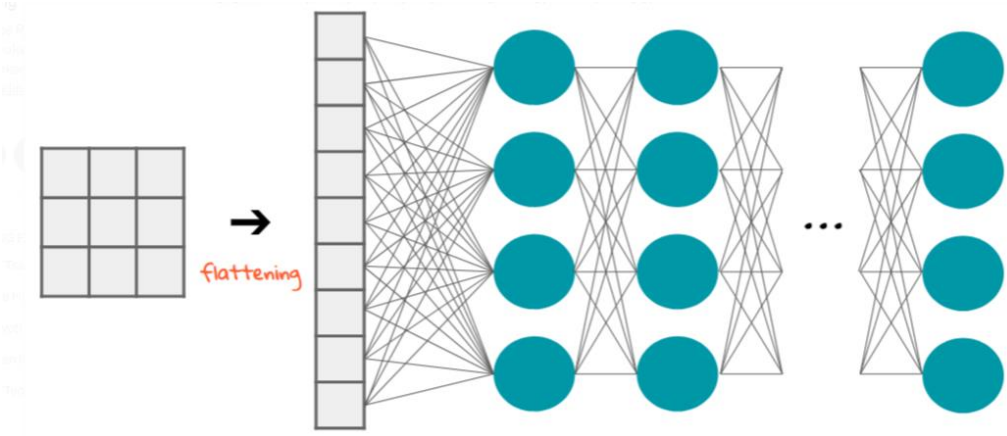
```
model.add(Conv2D(64, (3,3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
```

3. Convolució + Max-pooling

El codi d'exemple es una extracció del programa principal per tal de veure com aquesta teoria s'aplica de manera senzilla amb poques línies de codi. En aquest cas en les tres capes utilitzem la funció **ReLu** per tal de filtrar però podem utilitzar altres com la **Sigmoid**. Aquest conjunt de capes s'anomena **convolution layers.**

- **Flattening**

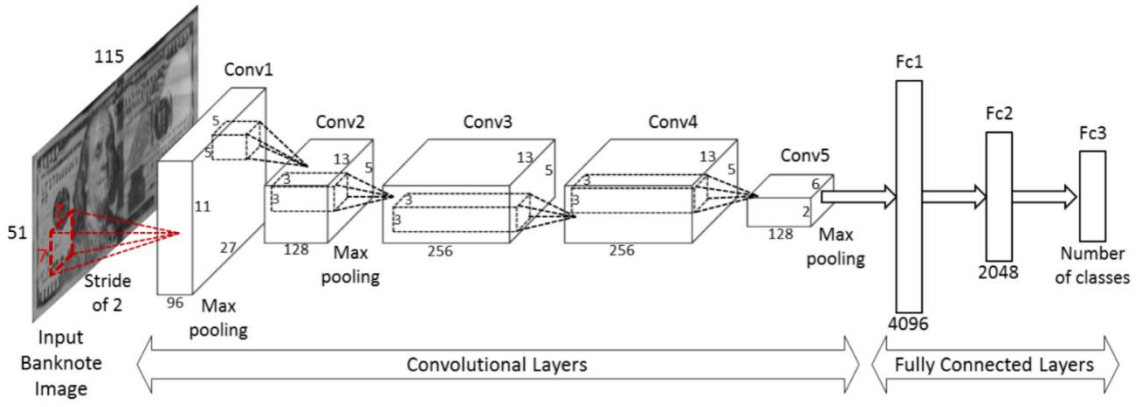
La següent operació que apliquem, després de fer les convolucions, es passar les matrius multidimensionals a un vector en una sola dimensió. Aquest tipus de capes s'anomenen **fully-connected layers** i son el pas previ d'una xarxa neuronal abans de donar un predicció o fer una classificació.



Il·lustració 37: *Procés de Flattening*

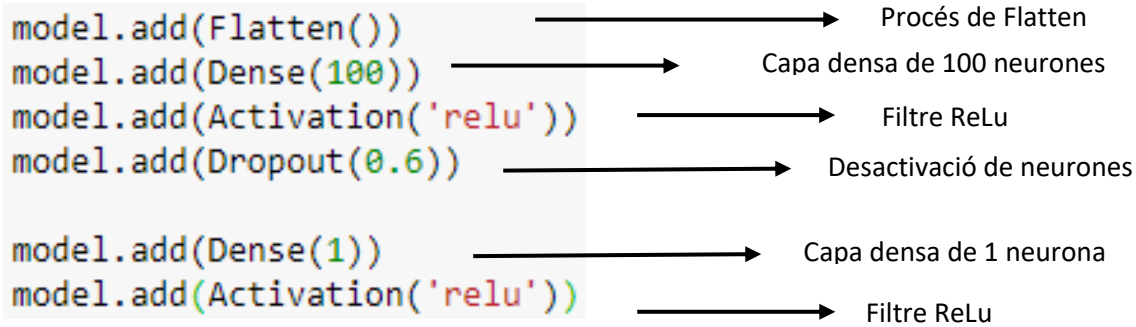
- **Dropout**

Dropout és un mètode que desactiva un nombre de neurones d'una xarxa neuronal de forma aleatòria. A cada iteració de la xarxa neuronal dropout desactivarà diferents neurones, les neurones desactivades no es tenen en compte per al **Forward-propagation** ni per al **Backward-propagation** el que obliga les neurones properes a no dependre tant de les neurones desactivades. Aquest mètode ajuda a reduir l'**overfitting** ja que les neurones properes solen aprendre patrons que es relacionen i aquestes relacions poden arribar a formar un patró molt específic amb les dades d'entrenament, amb dropout aquesta dependència entre neurones és menor a tota la xarxa neuronal, de aquesta manera les neurones necessiten treballar millor de forma solitària i no dependre tant de les relacions amb les neurones veïnes



Il·lustració 38: *Procés complet de les operacions dins d'una xarxa neuronal*

Si ens fixem en el tros de codi que segueix després de la convolució, podem observar que amb l'operació **Flatten** convertim les 32 imatges de mida 75x75 en un sol vector de longitud 180.000, això si només haguéssim fet una sola convolució. Per tal de reduir aquest nombre implementem una penúltima **capa densa** de 100 neurones que es filtra per la nostra funció d'activació **ReLu**. Per últim per evitar l'overfitting li diem que durant l'entrenament el 60% de les neurones, de manera aleatòria, es desactivin i deixem com a sortida 1 neurona que la filtrem amb la mateixa funció d'activació que abans. Notem que nosaltres nomes volem un resultat "bo" o "dolent".



Il·lustració 39: *Imatge del codi principal de les diferents capes ocultes*

La capa densa amb una solo neurona correspon a l'ultima neurona la qual ens donarà el resultat final de totes les operacions.

8.3 Optimitzadors

En essència, l'objectiu de l'entrenament de xarxes neuronals és minimitzar la **funció de cost [5 - 5.4]** i trobar els pesos adequats associats a cada neurona de la xarxa. El descobriment d'aquests pesos es duu a terme mitjançant un algorisme numèric anomenat backpropagation **[6]** ja explicat anteriorment.

L'optimitzador és l'encarregat de generar pesos cada cop millors, la seva importància és crucial. El seu funcionament essencial es basa a calcular el gradient de la **funció de cost** per cada pes de la xarxa. Com que volem minimitzar l'error, modificarem cada pes en la direcció (negativa) del gradient.

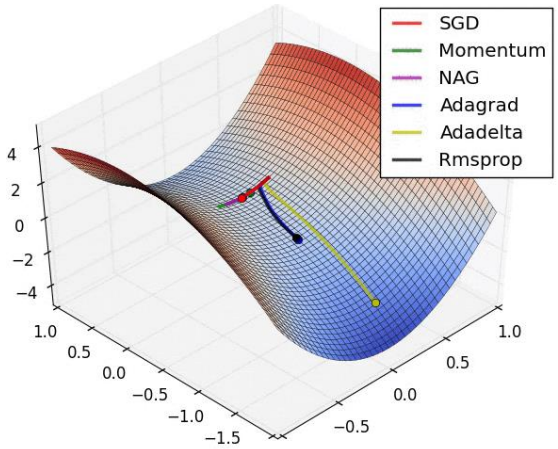
A continuació, presentarem una visió general dels principals optimitzadors:

- **Stochastic Gradient Descent (SGD):** SGD realitza una iteració amb cada mostra d'entrenament (és a dir, després de passar cada mostra d'entrenament, calcula la pèrdua i actualitza el pes). Com que els pesos s'actualitzen amb massa freqüència, la corba de pèrdua global seria molt sorollosa. Tanmateix, l'optimització és relativament ràpida en comparació amb altres.

- **Adam:** significa Adaptive Moment Estimation, és, amb diferència, l'optimitzador més popular i utilitzat a DL. En la majoria dels casos, podem triar a cegues l'optimitzador Adam i oblidar-nos de les alternatives d'optimització. Aquesta tècnica d'optimització calcula una taxa d'aprenentatge adaptativa per a cada paràmetre. Defineix el moment i la variància del gradient de la pèrdua i aprofita un efecte combinat per actualitzar els paràmetres de pes. L'impuls i la variància junts ajuden a suavitzar la corba d'aprenentatge i millorar eficaçment el procés d'aprenentatge

Existeixen altres optimitzadors que depenen del model de DL que utilitzem poden ser útils. No entrarem massa en detall en cadascun donat que l'objectiu és tenir una visió amplia de la seva existència i el paper que tenen una xarxa neuronal:

- Adagrad
- Adadelta
- RMSprop
- Adamax
- Nadam



Il·lustració 40: *Representació de la eficiència dels diferents optimitzadors*

Per poder continuar amb el nostre exemple particular d'ous veurem que optimitzador em triat i altres paràmetres.

```
optimizer=optimizers.Adam(learning_rate=0.001)#Learnin rate !
model.compile(optimizer=optimizer,
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

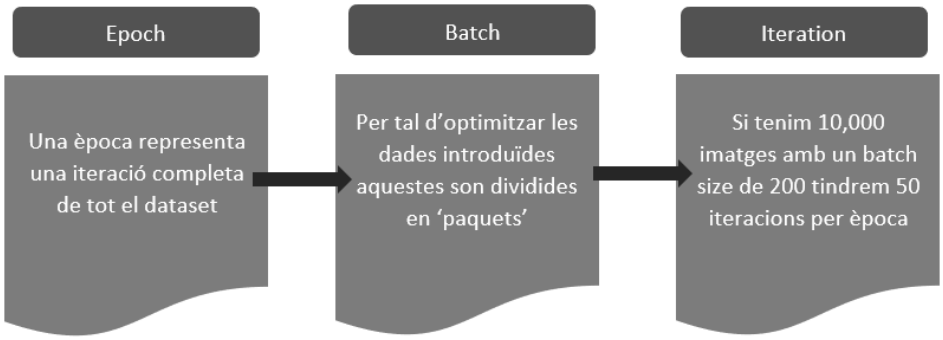
Il·lustració 41: *Imatge del codi principal corresponent a l'optimització*

Com hem mencionat abans, el nostre problema enfoca una classificació binaria (1 si es bo i 0 si es dolent) per aquest tipus de càlcul veiem que l'optimitzador Adam pot ser la millor opció degut a la seva polivalència. Si ens fixem hi ha un paràmetre anomenat **learning rate,** aquest ens indica el percentatge amb el que els pesos s'actualitzen a la xarxa. Aquest percentatge va del 0 al 1 i tampoc tenim cap mètode per calcular-lo, per aquesta raó haurem de fer proba i error. El paràmetre **loss** fa referencia a la funció de pèrdua explicada a l'apartat 5.4. **Metrics** es la llista de mètriques que ha d'avaluar el model durant l'entrenament i les proves, com per exemple la precisió en la predicció.

9. Entrenament i Predicció

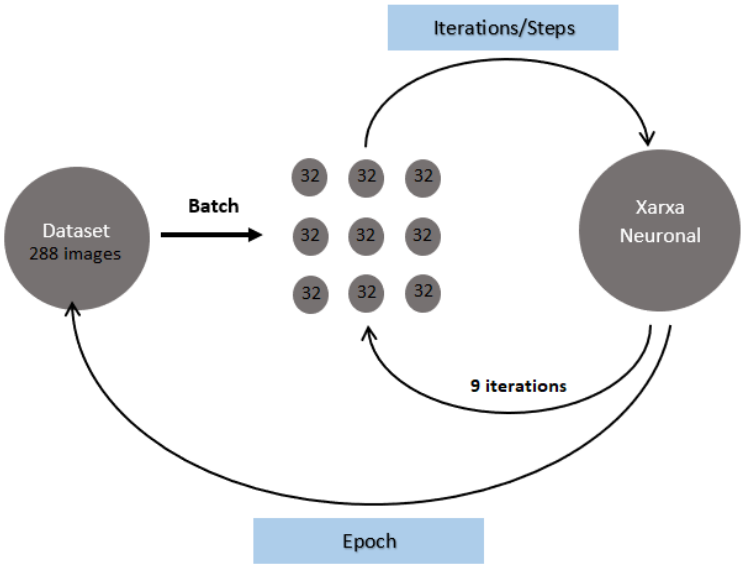
Un cop configurem un model, tenim totes les peces necessàries. Ara podem seguir endavant i entrenar el model amb les dades. Durant la formació, sempre és una bona pràctica proporcionar un conjunt de dades de validació per avaluar si el model funciona com es desitja després de cada època. El model aprofita les dades d'entrenament per entrenar-se i aprendre els patrons, i al final de cada època, utilitzarà les dades de validació no vistes per fer prediccions i calcular mètriques. El rendiment del conjunt de dades de validació és una bona indicació del rendiment general.

Avanç de continuar definirem que es una època dins d'una xarxa neuronal i altres conceptes directament relacionats que tenen una gran importància durant l'entrenament.



Il·lustració 42: *Agrupació de les mostres*

Aquest paràmetres es poden modelar segons la necessitats de la xarxa neuronal, per tant, no hi ha cap numero "òptim" que es pugui generalitzar a totes les xarxes. Doncs la manera correcte de trobar aquest paràmetre serà amb proba i error.



Il·lustració 43: *Esquema de fluid del tractament de les mostres dins d'una xarxa*

9.1 Paràmetres de entrenament

Amb el conceptes anteriors podem interpretar el codi que ens permet entrenar el model.

```
history=model.fit(
    train_generator,
    steps_per_epoch=train_sample//batch_size,
    epochs=25,
    shuffle= True,
    validation_data=validation_generator,
    validation_steps=validation_sample//batch_size)
```

Il·lustració 44: *Imatge del codi principal corresponent a l'entrenament de la xarxa neuronal*

Per poder començar a entrenar hem de definir els paràmetres que anteriorment hem parlat. Podem observa que li hem de passar les dades d'entrenament, en el nostre cas son les imatges de ous bons i dolents agrupades en **train_generator,** seguidament hem de definir les **iteracions** i ho podem fer dividint el nostre total de mostres entre el **batch size** triat. En el nostre cas tenim 214 imatges d'entrenament i un batch size de 32,per tant a cada època tindrem 6 iteracions. La segona part es afegir **validation_data**, aquests paràmetres son mostres distintes a avaluar el rendiment de la xarxa. De la mateixa manera per tal d'optimitzar aquesta tasca les mostres de validació també son agrupades. Doncs tenim que les nostres mostres de validació, 272 imatges (ous bons i dolents) amb un batch size de 32, per tant tindrem 8 iteracions.

```
Epoch 1/25
6/6 [==============================] - 5s 658ms/step - loss: 0.7330 - accuracy: 0.5275 - val_loss: 0.
7017 - val_accuracy: 0.5117
Epoch 2/25
6/6 [==============================] - 3s 536ms/step - loss: 0.6816 - accuracy: 0.5625 - val_loss: 0.
6663 - val_accuracy: 0.4922
Epoch 3/25
6/6 [==============================] - 3s 561ms/step - loss: 0.6164 - accuracy: 0.6615 - val_loss: 0.
5372 - val_accuracy: 0.8242
Epoch 4/25
6/6 [==============================] - 4s 663ms/step - loss: 0.5291 - accuracy: 0.7418 - val_loss: 0.
4927 - val_accuracy: 0.7773
Epoch 5/25
6/6 [==============================] - 4s 616ms/step - loss: 0.4410 - accuracy: 0.7917 - val_loss: 0.
6117 - val_accuracy: 0.6641
Epoch 6/25
6/6 [==============================] - 5s 838ms/step - loss: 0.3617 - accuracy: 0.8462 - val_loss: 0.
2978 - val_accuracy: 0.9141
Epoch 7/25
6/6 [==============================] - 6s 1s/step - loss: 0.3780 - accuracy: 0.8407 - val_loss: 0.368
8 - val_accuracy: 0.8359
```

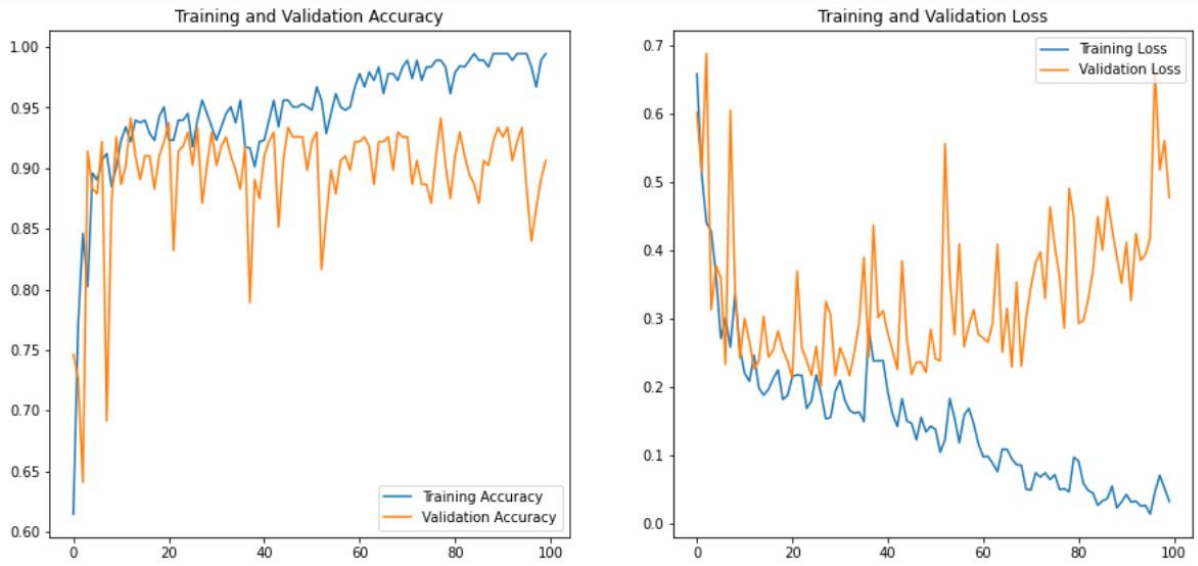
Il·lustració 45: *Feedback de la xarxa neuronal durant l'entrenament*

9.2 Resultats de l'entrenament

Després de cada entrenament podem graficar les variables que la xarxa ens proporciona després de cada època, com són **loss** i **accuracy** amb les dades d'entrenament i les de validació, com són **val_loss** i **val_accuracy** tal i com veiem a la **il·lustració 46.** La **Taula 1** ens proporciona les característiques de les mostres durant l'entrenament. El model de xarxa neuronal utilitzat es el mateix que hem anat explicat durant el capítol de desenvolupament.

Mostres	Format
Mida imatges	75x75
Format d'imatge	RGB
Batch size	32
Epochs	100
Mostres d'entrenament	214
Mostres de validació	242
Optimitzador	Adam
Funció de pèrdua	Binary crossentropy
Learning rate	0.001

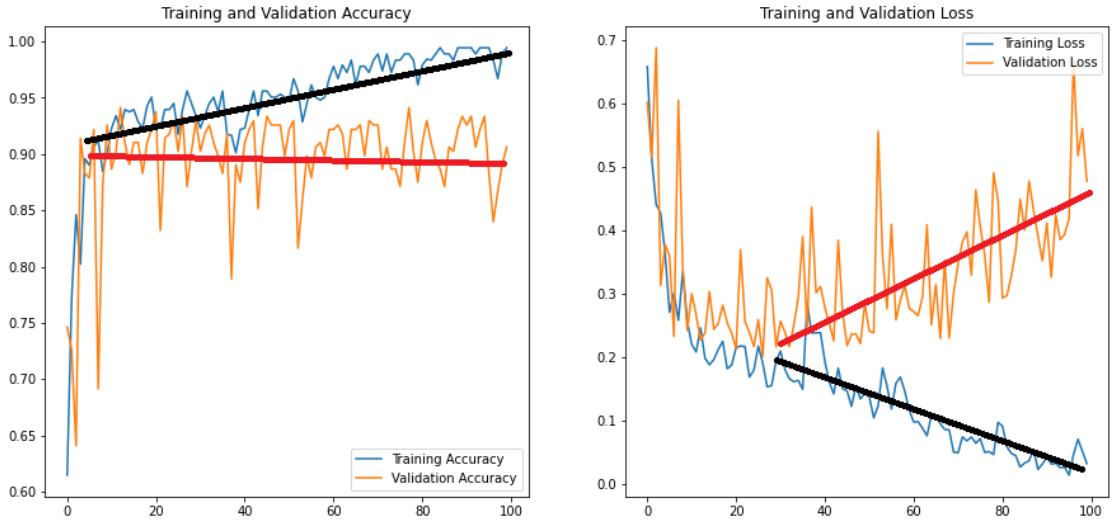
Taula 2: *Característiques de les mostres*



Il·lustració 46*: Gràfiques d'entrenament obtingudes del codi principal.*

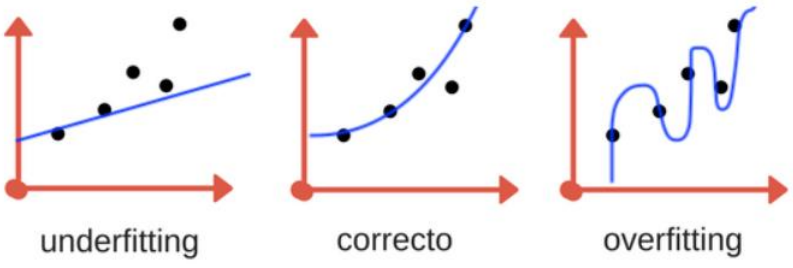
9.3 Overfitting & Underfitting

Quan entrenem el nostre model intentem fer encaixar -fit en anglès- les dades d'entrada entre ells amb la sortida. Potser es pot traduir overfitting com a “sobreajust” i underfitting com a “subajust” i fan referència a la fallada del nostre model en generalitzar -encaixar- el coneixement que pretenem que adquireixin. Són conceptes importants, ja que, el nostre objectius es obtenir un model que pugui generalitzar a l’hora de predir un ou bo o dolent. Si ens fixem en les gràfiques que hem obtingut podem veure una gran dispersió de les dades d’entrenament respecte a les dades de validació.



Il·lustració 47: *Anàlisi dels resultat obtinguts del codi principal*

En aquest punt podem veure que la precisió de les mostres d’entrenament augmenta, ja que amb aquestes aprèn la xarxa, però amb les mostres de validació baixa la precisió, ja que són dades que mai ha vist i, per tant, és incapaç de deduir si es tracta d’un ou bo o dolent. Amb la funció de pèrdua podem confirmar aquesta teoria observem que la línia d’aprenentatge al principi baixa i seguidament començar a pujar. Arribat aquest punt la xarxa ha memoritzat les dades traduint-se en un overfitting, això ocasionaria que amb la predicció d’una mostra que mai ha vist sigui incapaç de classificar-la. El mateix resultat obtindríem amb el underfitting donat que quan li passem una mostra amb unes lleugeres diferencies no pot generalitzar degut a poques imatges d’entrenament.



Il·lustració 48: *Overfitting, Underfitting*

9.4 Anàlisis dels resultats

En aquests gràfics podem observar la relació que hi ha entre els diversos valors de la resolució de la imatge, per tal de veure fins quins llindars la xarxa neuronal es capaç d'entrenar de manera òptima. Tan mateix observem que amb la primera resolució de proba, amb un mida de 75x75, obtenim resultats molt bons donat que durant l'entrenament i la validació les corbes augmenten, en el cas de la precisió i disminueix en el cas de la pèrdua ambdós equitativament, punt molt important.

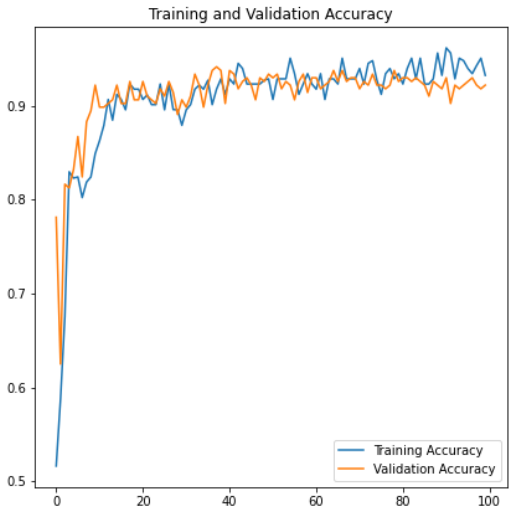
Aquests resultats són importants donat que d'aquesta manera podem veure que no tenim problemes de overfitting o underfitting .

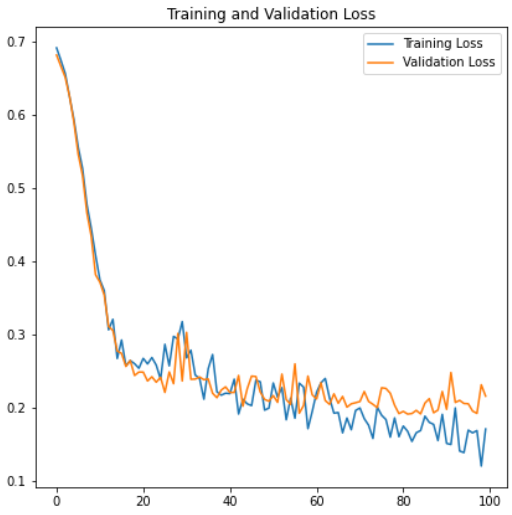
Per tal de trobar una resolució òptima, s'exposen les diferents gràfiques per observar els diferents resultats segons la resolució **[il·lustració 49-52]**. Es pot veure que tant una resolució massa petita com és 25x25 com una resolució gran de 150x150 els resultats empitjoren degut a que en el primer cas tenim underfitting i en el segon cas overfitting.

Mida Imatge	Gray/RGB	Batch size	Epochs	Learning rate	Imatge
75x75	RGB	32	100	0.0001	Il·lus. 49
25x25	RGB	32	100	0.0001	Il·lus. 50
100x100	RGB	32	100	0.0001	Il·lus. 52
150x150	RGB	32	100	0.0001	Il·lus. 56

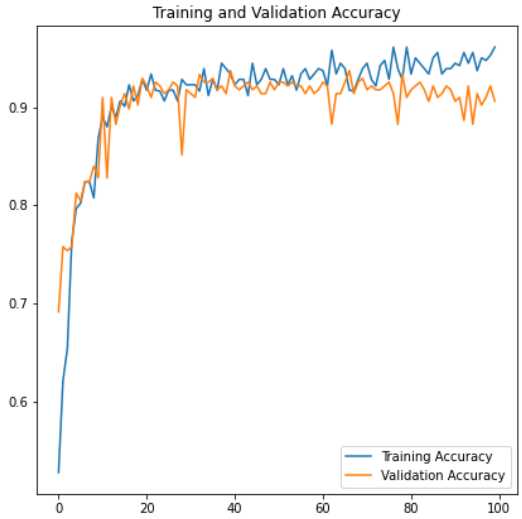
Taula 3: *Configuració de les imatges d'entrada*

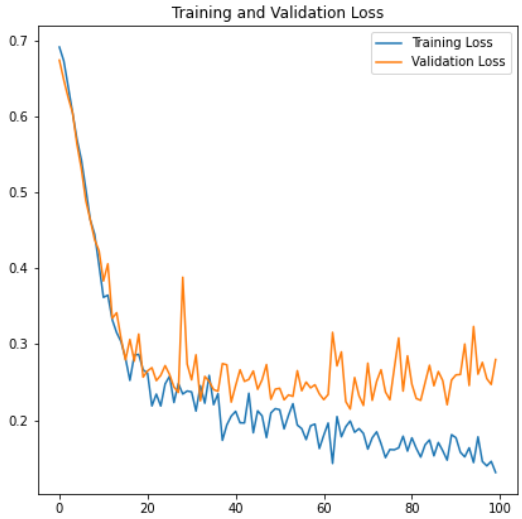
Il·lustració 49: *Imatge 75x75*



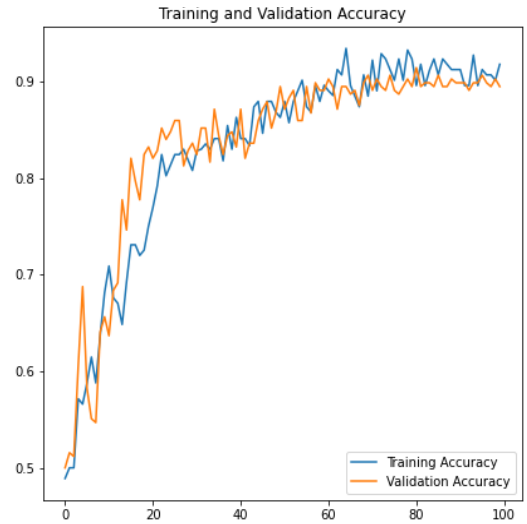


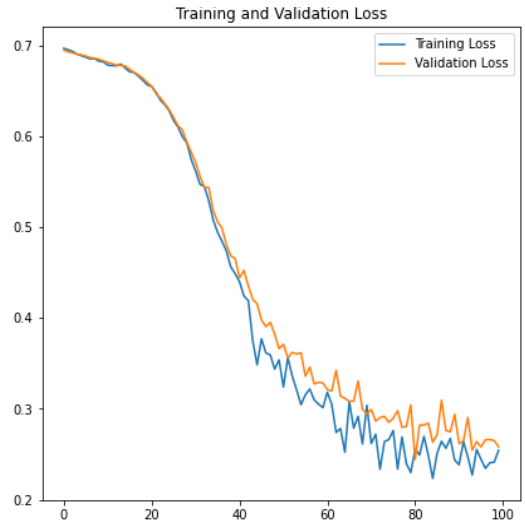
Il·lustració 52: *Imatge 25x25*





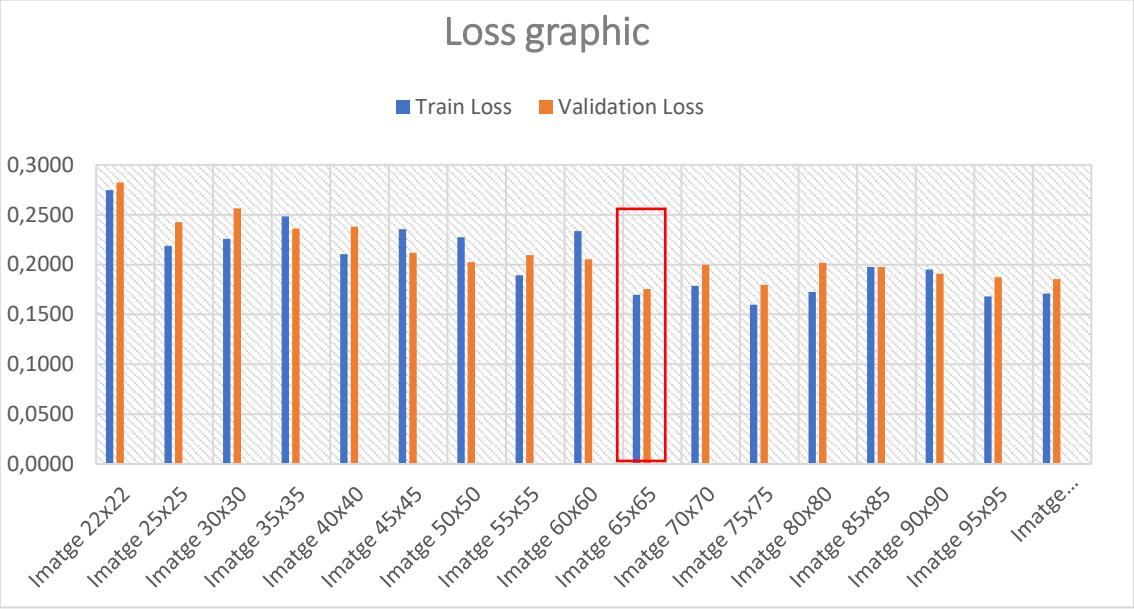
Il·lustració 51*: Imatge 150x150*



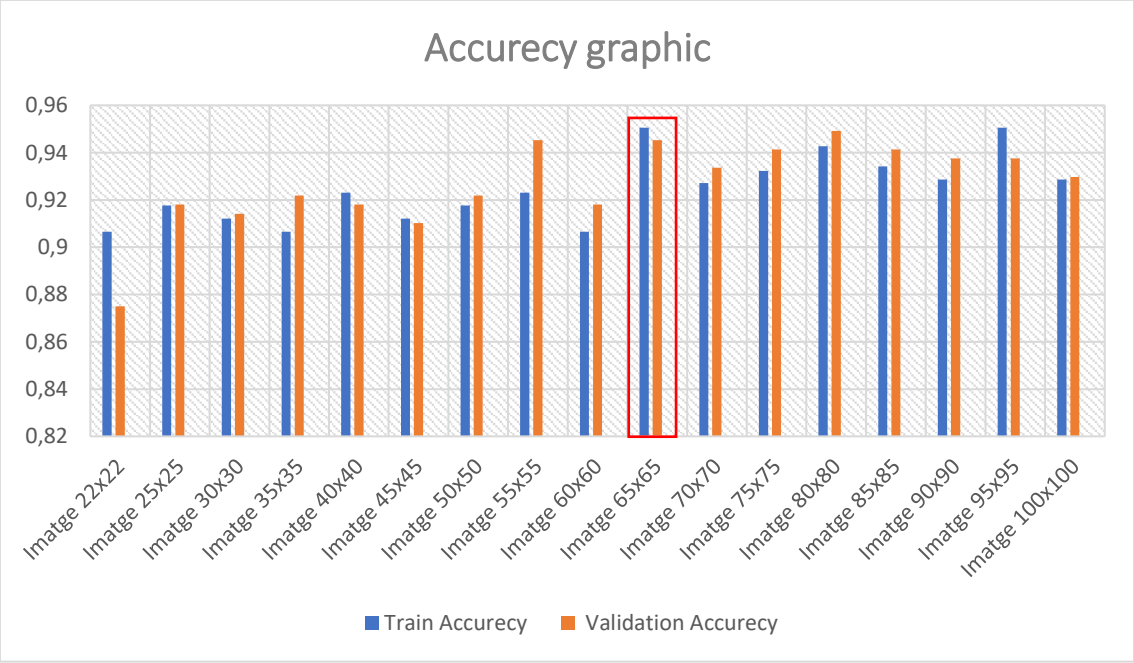


Il·lustració 50: *Imatge 100x100*

Per tal d'aconseguir la millor resolució he fet dos gràfiques amb 17 mides diferents amb els factors de **precisió** i **pèrdua,** tant de l'entrenament com de la validació, per tenir una visió més general dels resultats obtinguts. Com a conclusió podem veure que amb una resolució de 65x65 obtenim els millors resultats amb una alta precisió i una baixa pèrdua. **[il·lustració 53-54]**



***Il·lustració 53**: Gràfica de diverses resolucions amb la mètrica de pèrdua*



***Il·lustració 54:** Gràfica de diverses resolucions amb la mètrica de precisió*

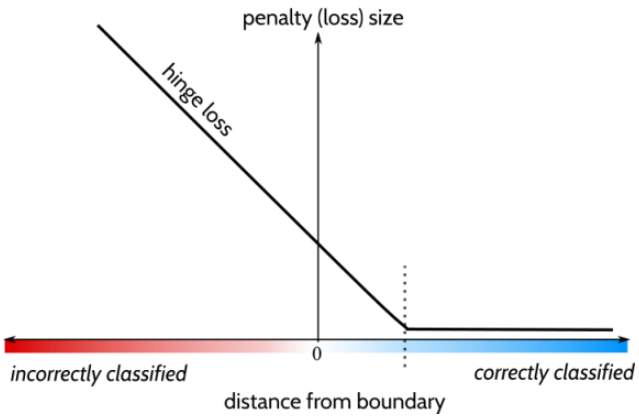
En aquest punt , un altre factor que pot ser clau, tal com mencionem en altres capítols, es el nombre de capes, aquest no es fixe per tant amb intuïció i els resultats de les gràfiques em d'ajustar la millor configuració, donat que poques capes o massa capes ocultes poden donar un model dolent.

T.A : Train Accuracy % / **T.L=** Train Loss %
V.A= Validation Accuracy % / **V.L=** Validation Loss %

Mida	Capa 1	Capa 2	Capa 3	Capa 4	T.A	T.L	V.A	V.L	Num
65x65	32	0			0.9427	0.1709	0.9141	0.2732	**1**
65x65	32	32	0		0.9396	0.1625	0.9336	0.2155	**2**
65x65	32	32	32	0	0.9176	0.2381	0.9258	0.2164	**3**
65x65	32	64	64	0	0.9341	0.1808	0.9336	0.1738	**4**
65x65	32	64	128	0	0.9341	0.1787	0.9453	0.1773	**5**
65x65	32	32	64	128	0.9231	0.1894	0.9336	0.1760	**6**
65x65	32	64	64	128	0.9341	0.2059	0.9297	0.1895	**7**

Taula 4: *Configuració capes ocultes*

Recordem que per tal de agafar el millor model hem de saber identificar una bona gràfica de precisió i de pèrdua. Així mentre més baix sigui la funció de pèrdua millor serà la classificació i per tant la xarxa generalitzar millor.



Il·lustració 55: *Gràfica de pèrdua*

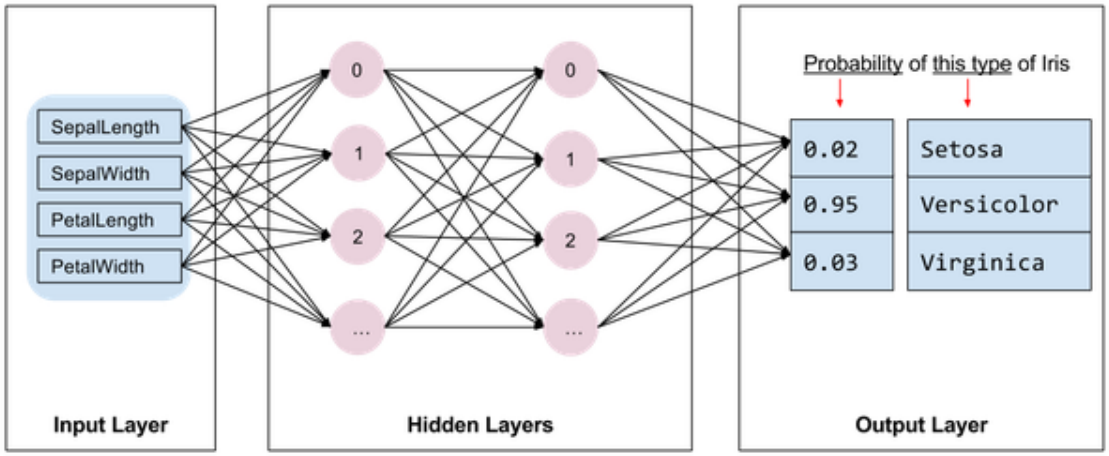
Si ens fixem en les mètriques **T.L** i **V.L** de cada configuració ens fixem que les millors són 4,5 i 6 degut a que ambdós mètriques són relativament baixes, el qual ens indica que la xarxa en comparació amb altres models té una tassa d'aprenentatge alta. Altres mètriques que tenim en compte són **T.A** i **V.A**, la qual ens indica el percentatge de precisió a l'hora de classificar una imatge, per tant hem de triar la mètrica més alta. Veiem que per una imatge de mida 65x65 la millor configuració de capes es la numero **5**.

9.5 Predicció

L'últim punt del model es la predicció. En aquest procés hem de contrastar els resultats obtinguts amb el model triat amb anterioritat, per això, fem ús de les mostres de test. Recordem que aquestes mostres no han sigut processades per la xarxa durant l'entrenament.

Com hem dit l'objectiu d'aquesta etapa es contrastar si les mètriques donades durant l'entrenament i prèviament analitzades concorden amb la realitat, només d'aquesta manera podrem saber si el model es funcional. Com es pot veure un cop la xarxa a fet la seva predicció "Good" o "Bad", amb el nom de la imatge podem veure si la predicció es correcta o no.

Com es pot veure en la il·lustració 56 el resultat d'una xarxa es un numero que ens indica la probabilitat en la predicció. En el nostre cas al tenir només dos possibles sortides ho podem simplificar a "0" bo i "1" dolent. El nombre total d'imatges que li hem passat són 59, la predicció ha sigut bastant robusta donant un èxit del 100%, cal mencionar que algunes mostres tenien esquerdes visibles, d'altres sang o estaven un mica trencats, característiques bastant visibles



Il·lustració 56: *Output d'una xarxa neuronal*

```
Bueno_1176.jpg: Good 0.0
Bueno_1177.jpg: Good 0.0
Bueno_1178.jpg: Good 0.0
Bueno_1179.jpg: Good 0.0
Bueno_118.jpg: Good 0.0
Malo_1639.jpg: Bad 1.0
Malo_1640.jpg: Bad 1.0
Malo_1641.jpg: Bad 1.0
```

Il·lustració 57: *Resultats de la predicció*

Bueno_1176 **Bueno_1179** **Bueno_1639**

Il·lustració 58: *Mostres de test*

10. Conclusions

L'objectiu d'aquest treball de fi de grau era obtenir un sistema de detecció d'ous amb ajuda de la intel·ligència artificial, en concret amb xarxes neuronal convolucionals, capaç de fer una bona classificació dels ous. Per tal de fer això aquest treball recull de manera ordenada tots aquells punts importants des de que s'inicia el projecte fins obtenir un model funcional.

El primer punt important es l'obtenció de les imatges i el seu tractament, punt en el qual hi ha més treball degut a la importància de segmentar molt bé la imatge per obtenir un sol ou a partir d'un conjunt. Aquesta part del projecte ha estat la que més temps ens ha ocupat degut a que hi ha diversos factor com la lluminositat que poden variar els resultats i es difícil poder generalitzar, per aquest motiu, una conclusió clara que vaig poder arribar durant aquesta etapa va ser que les condicions en l'àrea on es volia implementar el sistema havien d'estar controlades degut a que una petita variació, en la posició de la càmera o, com hem dit, de la lluminositat podria inhabilitar el sistema.

Un cop segmentades les imatges, havíem de començar a crear un data set ampli per tenir imatges suficients a l'hora d'entrenar la xarxa neuronal. Aquest punt es important degut a que , i com es comenta durant el projecte, s'utilitza les mateixes imatges transformades amb diverses tècniques. Pot ser per un desenvolupament posterior en la millorar del meu model, valdria molt més obtenir mostres directament de la granja on tindríem diferents característiques.

El segon punt important era triar quin tipus de xarxa neuronal utilitzar en aquest projecte i quin tipus de estructura tindria. Després d'una breu recerca, es va triar el model convolucional, un factor que va ser determinant en l'elecció d'aquest model va ser l'amplia informació que hi havia, sobre tot, en TensorFlow i Keras, llibreries que simplificaven molt la recerca

A partir d'aquí, la intenció com es comenta al principi, es deixar cada etapa d'aquest projecte documentada per tal d'entendre la complexitat que hi ha, personalment ha sigut molt important ja que només així he après realment el funcionament d'aquest model i endinsar-me totalment en aquest projecte.

Amb els coneixements adquirits tinc clar que encara hi ha un ampli rang de possibilitats per tal de millorar el model amb els diferents paràmetres. Aquest projecte m'ha donat la possibilitat de donar un solució a un problema real i l'excusa perfecte per aprendre molt més sobre d'intel·ligència artificial. Crec que les meves ganes i la ajuda del meu tutor he pogut obtenir uns resultats molt bons.

L'única part del projecte en la que m'hagués agradat continuar treballant es en el hardware i tenir la oportunitat de veure el sistema implementat en la granja.

11. Bibliografia

[1] *" Intel·ligència Artificial "* | https://builtin.com/artificial-intelligence/

[2] "Machine Learning" | https://www.ibm.com/es-es/cloud/learn/machine-learning/

[3] *"Deep Learning "* | https://machinelearningmastery.com/what-is-deep-learning/

[4] *"Deep Learning I "* | https://www.ibm.com/cloud/learn/deep-learning/

[5] *" Informació avicultura Espanya "* | https://avicultura.info/determinacion-de-calidad-del-huevo/

[6] *"Informació sistemes de detecció de fissures en ous"* | https://www.thepoultrysite.com/articles/a-better-way-to-spot-eggshell-cracks/

[7] *"Qualitat d'un ou"* | https://avicultura.info/determinacion-de-calidad-del-huevo/

[8] *"TensorFlow vs Keras"* | https://towardsdatascience.com/tensorflow-vs-keras-d51f2d68fdfc

[9] *" Xarxes Neuronal Convolucionals"* | https://cnvrg.io/cnntensorflow/?gclid=CjwKCAiAxJSPBhAoEiwAeO_fPeNhMPokgNv0DDeCAWwTc0NXfaVAXJMhhui6-G6uNpcfT90koQz2xoC73YQAvD_BwE/

[10] *" Xarxes Neuronal Convolucionals I" | https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/*

[11] "Optimizadors" | https://ichi.pro/es/descripcion-general-de-diferentes-optimizadores-para-redes-neuronales-247308806799555

[12] *"Informació Back-propagation i Forward-propagation"* | https://rubialesalberto.medium.com/redes-neuronales-propagaci%C3%B3n-hacia-adelante-y-propagaci%C3%B3n-hacia-atr%C3%A1s-4745c0fb6286

[13] *" Informació sobre les Xarxes Neuronals"* | https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/

[14] *" Data augmentation "* | https://nanonets.com/blog/data-augmentation-how-to-use-deep-learning-when-you-have-limited-data-part-2/

[15] *"Data augmentation TensorFlow"* | https://www.tensorflow.org/tutorials/images/data_augmentation/

[16] *"Max-pooling"* | https://www.machinecurve.com/index.php/2020/01/30/what-are-max-pooling-average-pooling-global-max-pooling-and-global-average-pooling/

[17] *" Informació del procés de Flattenig"* |https://www.quora.com/What-is-the-meaning-of-flattening-step-in-a-convolutional-neural-network/

[18] *" Funció de pèrdua"* | https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/

[19] *" Overfitting i Underfitting"* | https://www.aprendemachinelearning.com/que-es-overfitting-y-underfitting-y-como-solucionarlo/

[20] *" Informació API TensorFlow"* | https://www.tensorflow.org/api_docs/python/tf/keras/Model/

[21] *"Informació API Keras"* | https://keras.io/api/layers/convolution_layers/