UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Politècnica Superior d'Enginyeria
de Manresa

TREBALL FINAL DE GRAU

IMPLEMENTACIÓ D'UN SISTEMA INTEL·LIGENT DE VISIÓ ARTIFICIAL PER UN PROCÉS DE CONTROL DE QUALITAT INDUSTRIAL MITJANÇANT XARXES NEURONALS

Grau en Enginyeria Electrònica Industrial i Automàtica

Curs 20/21

Autora: Anna Marbà Mas

Director: Víctor Barcons Xixons

6 de Juliol del 2021, Manresa

"Llamamos "caos" al orden que todavía no comprendemos."

- Edward Lorenz, matemàtic

Aquest treball ha tingut un gran impacte tant en la meva formació acadèmica com en la personal, ha sigut un període de molta dedicació i implicació i és per aquest motiu pel qual vull expressar el meu profund agraïment a tots els que han format part d'alguna manera en aquest camí.

En primer lloc, m'agradaria agrair al meu tutor Víctor Barcons de l'EPSEM, per guiar-me en tot moment i mostrar-se sempre disposat a solucionar tots els meus dubtes. Així com també al professorat de l'EPSEM per l'aprenentatge rebut durant aquests anys.

Gràcies a l'empresa Airtificial Intelligent Robots per proporcionar-me el material, als meus companys de feina i en especial a l'Álvaro per la seva predisposició a ajudar-me.

Com no podia ser d'altra manera, vull agrair a la meva família i en particular a la meva mare Lourdes pel seu suport incondicional. També als meus amics i companys amb els quals he compartit aquesta etapa, i en especial al Carlos per recolzar-me sempre.

Finalment, una cordial menció als grups d'investigació i canals de divulgació que posen a disposició els seus treballs i estudis de manera pública i contribueixen a crear aquesta gran comunitat de recursos.

Moltes gràcies.

Resum

Aquest treball s'emmarca en l'adopció de la Intel·ligència Artificial en l'entorn industrial i es pretén mostrar les xarxes neuronals artificials com a mitjà per fer front a possibles mancances o limitacions actuals i l'automatització de processos que no aportin valor afegit.

Per l'elaboració d'aquest treball s'ha realitzat un profund estudi i recerca, en concret, en les xarxes neuronals artificials com a l'algoritme empleat en Aprenentatge Automàtic. Simulen el comportament de les neurones biològiques i són capaces d'aprendre a trobar els patrons que resideixen en el conjunt de dades d'entrada i elaborar-ne un model.

Es va detectar el procés de control de qualitat com un procés crític en el sector industrial per garantir la qualitat del producte i augmentar el rendiment productiu total. Per aquest motiu, s'ha implementat un sistema intel·ligent basat en el disseny i entrenament d'un algoritme amb xarxes neuronals convolucionals que ha sigut capaç d'aprendre correctament a classificar imatges de peces defectuoses i no defectuoses resultants d'un procés productiu. L'algoritme s'ha desenvolupat amb les llibreries *TensorFlow* i *Keras* d'Aprenentatge Automàtic. Per tal d'aconseguir un model de classificació òptim, es va realitzar un anàlisi de resultats obtinguts per diferents valors d'hiperparàmetres. Es va obtenir una precisió del 97.5% per la classificació de les imatges de validació per la qual cosa s'ha conclòs que les xarxes neuronals artificials poden presentar-se com una alternativa rendible a la programació convencional per la seva flexibilitat i adaptació a entorns amb molta variabilitat.

Abstract

This project refers to the adoption of Artificial Intelligence in the industrial environment and aims to show the artificial neural networks as a means to deal with possible weaknesses or current limitations and the automation of processes that do not provide added value.

To elaborate this project, a deep study and research has been carried out, in particular, in artificial neural networks as the algorithm used in Machine Learning. They simulate the behaviour of biological neurons and are able to learn to find the patterns that reside in the input dataset and develop a model.

The quality inspection process was detected as a critical process in the industrial sector to ensure product quality and increase overall production efficiency and performance. For this reason, an intelligent system has been implemented based on the design and training of an algorithm with convolutional neural networks that has been able to correctly learn to classify images of defective and non-defective parts resulting from a production process. The algorithm was developed with the *TensorFlow* and *Keras* machine learning libraries. In order to achieve an optimal classification model, an analysis of results obtained by different hyperparameter values was performed. An accuracy of 97.5% was obtained for the classification of validation images so it has been concluded that artificial neural networks can be a cost-effective alternative to conventional programming due to their flexibility and adaptation to environments with a lot of variability.

ÍNDEX DE CONTINGUT

1. INTRODUCCIÓ 12

1.1. Objectius 13
1.2. Abast 13
1.3. Motivació 14

2. ANTECEDENTS 16

2.1. Referents històrics 17
2.2. Machine Learning en la indústria tecnològica 18
2.3. Anàlisi del mercat 20
2.4. Evolució dels controls de qualitat industrials 22

3. MARC TEÒRIC: XARXES NEURONALS ARTIFICIALS 25

3.1. Tipus de xarxes neuronals 26

3.1.1. El perceptró 26
3.1.2. Xarxes neuronals multicapa 28
3.1.3. Xarxes neuronals convolucionals 29

3.2. Paradigma d'aprenentatge supervisat 35

3.2.1. Descens del gradient 35
3.2.2. Backpropagation 36
3.2.3. Hiperparàmetres d'entrenament 37
3.2.4. El problema del *overfitting* 41

4. DISSENY I IMPLEMENTACIÓ DEL SISTEMA INTEL·LIGENT 42

4.1. Tecnologies i eines utilitzades 43

4.1.1. Python 43
4.1.2. Tensorflow i Keras 44
4.1.3. Pycharm 45
4.1.4. Adobe Photoshop 45
4.1.5. Hardware 46

4.2. Estudi del dataset *47*

4.2.1. Interès d'aplicar *Machine Learning* 49

4.3. Preprocessament de les imatges 50
4.4. Disseny i construcció del model 53

4.4.1. Arquitectura de la CNN 54
4.4.2. Optimització d'hiperparàmetres 56

4.4.3. Programació de la CNN 57

4.5. Entrenament del model 62

5. AVALUACIÓ I ANÀLISI DE RESULTATS 63

5.1. Correlació entre els hiperparàmetres i el rendiment final 67
5.2. Elecció del model 70
5.3. Prediccions 72

6. LÍNIES FUTURES 75

6.1. Millores 75
6.2. Integració del model a la línia de producció 77

7. CONCLUSIONS 79

8. REFERÈNCIES BIBLIOGRÀFIQUES 81

ÍNDEX DE FIGURES:

Figura 1: Esquema abast de la implementació del sistema intel·ligent. 13

Figura 2: Diferències a nivell conceptual entre programari convencional i Machine Learning. 16

Figura 3: Duel d'escacs entre una màquina i Kasparov. 17

Figura 4: Robots magatzem automatitzat de Amazon. 18

Figura 5: Projecte Rob-aKademl. 19

Figura 6: Gràfic estat de l'adopció de la I.A. en empreses al 2021 21

Figura 7: Inspectora de qualitat de peces en una fàbrica de màquines de cosir, 1977 22

Figura 8: Sistema de visió artificial per control de qualitat amb interfície de monitorització. 23

Figura 9: Peces resultants d'un procés de foneria. 24

Figura 10: Representació aproximada d'una neurona biològica..... 25

Figura 11: Perceptró. 26

Figura 12: Funcions d'activació. 27

Figura 13: Regressió lineal simple.....27

Figura 14: Regressió lineal múltiple 27

Figura 15: Xarxa neuronal multicapa. 28

Figura 16: Model porta AND, OR, XOR. 29

Figura 17: Representació abstracta d'una CNN. 29

Figura 18: Representació d'una imatge en escala de grisos com una matriu de valors. 30

Figura 19: Resultat d'una convolució. 31

Figura 20: Mapes de característiques. 32

Figura 21: Operació max pooling 2x2 sobre una imatge de 4x4 píxels. 33

Figura 22: Exemple de l'estructura d'una xarxa neuronal convolucional. 33

Figura 23: Funció de cost. Avaluació de la inclinació. 35

Figura 24: Funció de cost. Apropant-se al mínim de la funció fent ús del descens del gradient..... 36

Figura 25: Error en funció de la predicció en probabilitat quan l'etiqueta real és 1. 39

Figura 26: Exemple molt senzill de la seqüència d'aprenentatge d'una xarxa neuronal. 40

Figura 27: Overfitting i underfitting. 41

Figura 28: Gràfic variació de l'error en el temps per les dades d'entrenament I dades de prova... 41

Figura 29: Diagrama conceptual per l'obtenció del model de classificació. 42

Figura 30: Logotip Python. 44

Figura 31: Logotip TensorFlow.....44

Figura 32: Logotip Keras. 44

Figura 33: Logotip PyCharm. 45

Figura 34: Logotip Adobe Photoshop..... 45

Figura 35: Interfície gràfica Photoshop. 46

Figura 36: Peça OK..... 48

Figura 37: Peça NOK..... 48

Figura 38: Exemple de criteris per considerar una peça com a NOK. 49

Figura 39: Peça NOK 300x300 amb defectes provocats..... 52

Figura 40: Peça NOK 300x300 píxels amb defectes provocats 52

Figura 41: Captura de pantalla de l'estructura de carpetes en local 53

Figura 42: Arquitectura de la xarxa neuronal convolucional en qüestió 55

Figura 43: Captura de pantalla de l'IDE de PyCharm 58

Figura 44: Captura de pantalla de l'IDE PyCharm 59

Figura 45: Relació entre els dos fitxers de programa main.py i predict.py..... 61

Figura 46: Procés d'entrenament en curs 62

Figura 47: Efecte d'un valor massa petit de lr per convergir en un mínim de la funció de cost..... 65

Figura 48: Efecte d'un valor massa gran de lr per convergir en un mínim de la funció de cost. 66

Figura 49: Gràfic Impacte dels hiperparàmetres en la pèrdua. 68

Figura 50: Gràfic Impacte dels hiperparàmetres en la precisió. 70

Figura 51:Gràfic de les mètriques de rendiment per lr=0.001 i batch_size=4. 71

Figura 52: Captura de pantalla dels resultats de la predicció. 72

Figura 53: Segmentació d'imatges amb k-means..... 76

Figura 54: Diagrama conceptual fases integració del model de classificació en producció. 77

ÍNDEX DE TAULES:

Taula 1: Comparativa entre xarxa neuronal multicapa i convolucional..... 30

Taula 2: Estructura de la xarxa neuronal convolucional. 54

Taula 3: Espai de cerca en quadrícula per cada combinació d'hiperparàmetres..... 57

Taula 4: Resultats obtinguts per cada combinació d'hiperparàmetres. 64

Taula 5: Resultats obtinguts per més èpoques d'entrenament..... 67

Taula 6: Mètriques de rendiment per lr=0.001 i batch_size=4..... 71

Taula 7: Resultats de la predicció..... 72

Taula 8: Prediccions de classificació i la seva etiqueta real. 74

1. INTRODUCCIÓ

En els últims anys s'han produït grans avenços tecnològics i descobriments de noves tècniques i algoritmes[1] que han portat a la indústria cap a una nova època, denominada per l'economista alemany *Klaus Schwab* com la Quarta Revolució Industrial. (1) Aquesta revolució presenta nombrosos avantatges i oportunitats en la profunda transformació de l'economia, la societat i el mercat laboral però també comporta reptes d'adaptació per enfrontar la transició d'època. El seu impacte promet ser significatiu en diferents àmbits com la medicina, la seguretat, la comunicació, la robòtica, el transport...

El treball final de grau porta com a títol "*Implementació d'un sistema intel·ligent de visió artificial per un procés de control de qualitat industrial mitjançant xarxes neuronals*" i s'emmarca en l'adopció de la I.A. en l'entorn industrial.

En aquest treball es pretén mostrar les xarxes neuronals artificials i l'Aprenentatge Automàtic com a mitjà per implantar millores a mancances o limitacions actuals en la indústria gràcies als beneficis que ofereixen. En aquest sentit, s'ha realitzat un previ anàlisi del mercat actual amb la finalitat de determinar les necessitats reals en la indústria. Es va constatar una necessitat en el sector de la indústria tecnològica pel que fa a la millora del rendiment productiu mitjançant l'automatització de processos. En concret, es va detectar el procés de control de qualitat com un procés crític per garantir la qualitat del producte a client.

Arribats a aquest punt, la finalitat d'aquest treball és presentar les xarxes neuronals artificials com una alternativa rendible i innovadora per la seva capacitat d'aprendre a trobar les característiques que diferencien les peces defectuoses de les no defectuoses i l'elaboració d'un model de classificació. En primer lloc, les xarxes neuronals poden ser una solució especialment en els casos en els quals les dades d'entrada presenten molta variabilitat i resulta molt difícil o impossible generalitzar-ho i programar-ho explícitament mitjançant lògica condicional. En segon lloc, suposaria una reducció de temps i costos gràcies a l'automatització del procés d'extracció de característiques.

Per tal de comprovar-ne el rendiment, s'ha escollit implementar un sistema intel·ligent mitjançant xarxes neuronals convolucionals en el context de l'automatització d'un procés de control de qualitat industrial.

---

[1] *Algoritme: en programació es tracta d'una seqüència de instruccions lògiques per dur a terme una sèrie de processos a fi de donar resposta a determinats problemes.*

1.1. Objectius

Els objectius principals d'aquest treball són els següents:

- Recercar i aprendre sobre el gran camp de la I.A.
- Ampliar coneixements sobre visió artificial, programació i automatització.
- Enumerar possibles aplicacions de *Machine Learning* i llistar els seus avantatges en l'àmbit industrial.
- Realitzar un estudi teòric de les xarxes neuronals artificials a nivell d'arquitectura, tipologies i paradigma d'aprenentatge.
- Dissenyar i entrenar un algoritme d'Aprenentatge Automàtic capaç de classificar peces defectuoses i no defectuoses d'un determinat procés productiu mitjançant xarxes neuronals.
- Avaluar el rendiment d'aquest model per diferents hiperparàmetres.

1.2. Abast

Per tal de concretar l'abast d'aquesta implementació, s'ha elaborat el diagrama de la *Figura 1* i s'ha definit l'ús que es fa del terme "sistema intel·ligent". Encara que sovint s'utilitza aquest terme englobant també els dispositius que el formen com els sensors i actuadors, en aquest treball es fa referència a sistema intel·ligent com a un programa de computació capaç de processar informació de l'entorn, obtenir-ne coneixement i donar una resposta. D'aquesta manera, l'entrada al sistema intel·ligent és la imatge capturada d'un sensor (una càmera de visió artificial) i la sortida és el resultat de la classificació de la imatge.

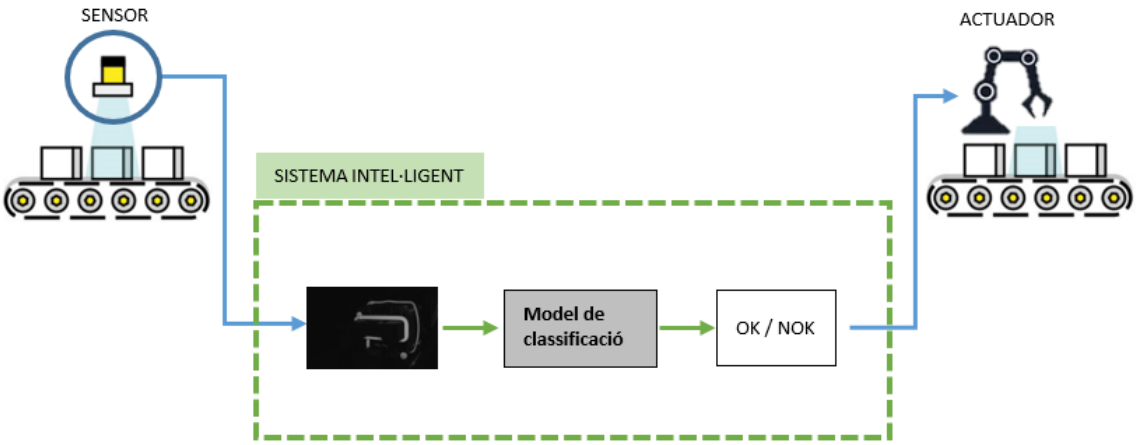


Figura 1: Esquema abast de la implementació del sistema intel·ligent. Font: elaboració pròpia

El treball s'estructura en un primer apartat *2. Antecedents* el qual ofereix una visió de l'estat actual pel que fa a l'adopció de la I.A. i *Machine Learning* en la indústria i correspon a recerca i anàlisis previs que han permès formular els objectius i finalitat del treball. Aquest apartat es considera molt important per introduir el lector en el marc d'aquest treball.

En segon lloc, en l'apartat *3. Marc teòric: Xarxes neuronals artificials* hi consten els fonaments teòrics amb els quals es basa la part pràctica, fent un especial èmfasi en les xarxes neuronals convolucionals. La informació s'ha extret d'articles web, llibres, canals de divulgació i altres recursos degudament referenciats en l'apartat *8. Referències Bibliogràfiques* al final del treball.

Finalment, la part pràctica d'aplicació consta d'un apartat *4. Disseny i implementació del sistema intel·ligent* on es dissenya l'algoritme mitjançant xarxes neuronals i s'entrena amb diferents imatges etiquetades de peces resultants d'un procés productiu per tal d'obtenir el model de classificació. En l'apartat *5. Avaluació i anàlisi de resultats* s'escull la combinació d'hiperparàmetres que ofereixi millor rendiment. En l'apartat *6. Línies futures* es realitza un llistat de millores en implementacions futures i una possible definició de la integració d'aquest model a la línia de producció per l'automatització del control de qualitat mitjançant visió artificial ja que la implementació a producció desafortunadament queda fora del meu propi abast.

1.3. Motivació

La idea d'aquest treball va sorgir el quadrimestre passat amb la proposta del tutor en la temàtica I.A., *Machine Learning* i xarxes neuronals. Fou llavors que vaig realitzar un projecte en el qual vaig començar a cercar en aquesta temàtica i implementar xarxes neuronals senzilles per diferents aplicacions (classificació de moviments i classificació de colors amb una placa *Arduino*) seguint instruccions i manuals de repositoris web.

Per altra banda, l'elecció d'aquest tema ha sigut en gran part pel meu interès des de sempre cap als robots intel·ligents, l'avenç de la tecnologia i l'automatització de processos productius en la indústria. Haver cursat el grau en Enginyeria Electrònica Industrial i Automàtica ha fet despertar la meva passió vers tot aquest món. El fet que les xarxes neuronals artificials es basin en el funcionament de les neurones biològiques i la creació d'una ment artificial em sembla increïblement fascinant.

El treball presenta una forta component teòrica degut al fet de tractar-se de temes que no s'havien desenvolupat durant la meva formació acadèmica a la universitat i la necessitat de realitzar una extensa recerca i estudi previs al desenvolupament de la part pràctica. Tanmateix, es sol veure les xarxes neuronals com una caixa negra des de la ignorància pel que fa al seu funcionament intern;

per tal de no limitar-me a això, s'intenta desvelar i entendre els processos i les operacions que s'hi realitzen que de fet, ho fa molt més interessant.

Pel que fa a la part pràctica del treball, fou l'empresa *Airtificial Intelligent Robots*, companyia líder en robòtica col·laborativa i desenvolupament d'estructures intel·ligents, en la qual realitzo pràctiques acadèmiques, que em va proporcionar generosament el *dataset* per entrenar la xarxa neuronal. Treballar amb aquestes imatges suposa molt valor personal ja que son imatges de peces resultants d'un procés de producció real i que de fet, he presenciat i he vist funcionar. Per aquest motiu estic molt agraïda a l'empresa per haver contemplat aquesta possibilitat.

2. ANTECEDENTS

Intel·ligència Artificial (I.A.) es defineix com la subdisciplina del camp de la Informàtica dedicada al desenvolupament d'algoritmes que permetin la creació de màquines que puguin imitar comportament intel·ligents com els següents:

- Robòtica: capacitat d'una màquina de moure's i adaptar-se a l'entorn.
- N.L.P. (*Natural Language Processing*): capacitat d'entendre el llenguatge.
- Veu: conversió de text a veu i de veu a text.
- Visió artificial (o visió per computador): detectar i classificar objectes en imatges o vídeos.

En la definició de I.A. cal remarcar la frase "imitar comportaments intel·ligents" ja que això no vol dir que aquests comportaments siguin en essència comportaments cognitius[2]. *Machine Learning* o Aprenentatge Automàtic és la rama del camp de la I.A. que busca dotar a les màquines de capacitat d'aprenentatge, entenent aprenentatge com la generalització del coneixement a partir d'un conjunt d'experiències. Així doncs, les màquines poden imitar comportaments intel·ligents perquè l'humà les hagi programat explícitament o, molt més interessant, perquè el propi sistema hagi après a realitzar-les. (2)

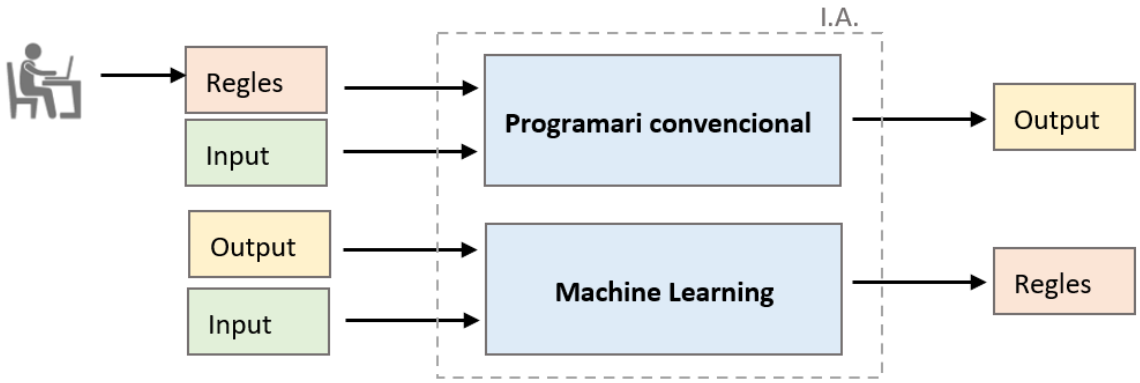


Figura 2: Diferències a nivell conceptual entre programari convencional i Machine Learning. Font: elaboració pròpia

Deep Learning o Aprenentatge Profund és una família d'algoritmes de *Machine Learning* i se'n fa referència sovint quan s'ha de tractar informació complexa i voluminosa (*Big Data*[3]) que requereixi de xarxes neuronals amb un alt número de capes i amb una arquitectura profunda.

---

[2] *Cognició: fa referència a l'aptitud dels éssers de processar informació a partir de la percepció i el coneixement adquirit per valorar i considerar certs aspectes en detriment d'altres.*

[3] *Big Data: Dades massives és el nom que reben els conjunts de dades, els procediments i les aplicacions informàtiques, que, pel seu volum, la seva naturalesa diversa i la velocitat a la qual han de ser processades, ultrapassen la capacitat dels sistemes informàtics habituals.*

A grans trets, en *Machine Learning* es poden realitzar tasques de classificació o regressió i la diferència està en el tipus de resultat que s'obté. En el primer cas, el resultat és una classe; per exemple, classificació de imatges de diferents tipologies de peça. Contràriament, en el segon cas, el resultat és un valor numèric; per exemple, predir el número de ventes d'una empresa en funció de diferents paràmetres com el número de clients o l'històric de ventes. Existeixen diferents tècniques i algoritmes que cobreixen aquestes dues tasques: arbres de decisió, regressió logística, regressió lineal, *k-means*, xarxes neuronals...

Les xarxes neuronals han demostrat ser més efectives per classificar *datasets*[4] d'imatges més complexos (3). Per aquest motiu, el marc teòric d'aquest treball es centra en tasques de classificació mitjançant xarxes neuronals i aprenentatge supervisat, el qual es caracteritza per un aprenentatge basat en un modelatge de dades prèviament etiquetades.

2.1. Referents històrics

Els primers referents històrics de la Intel·ligència Artificial es remunten a l'any 1950 quan *Alan Turing*, considerat un dels pares de la I.A., publica un article on es pregunta si les màquines poden pensar. (4)

En l'any 1997, una supercomputadora creada per *IBM* va vèncer el campió mundial del món dels escacs i fou aquest el punt d'inflexió on es va començar a sentir parlar de I.A. fora dels àmbits acadèmics i d'investigació.

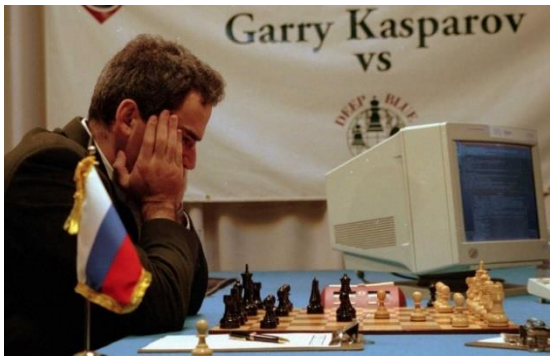


Figura 3: Duel d'escacs entre una màquina i Kasparov. Font: (5)

Les xarxes neuronals artificials van donar un gran pas quan a finals dels anys cinquanta l'informàtic *Frank Rosenblat* va dissenyar la primera xarxa neuronal artificial: el perceptró.

[4] *Dataset*: *conjunt d'imatges o conjunt de dades*

No obstant, a finals del segle XX va aparèixer una reacció motivada per la decepció de la comunitat degut a promeses poc realistes per part dels desenvolupadors i expectatives massa altes que va provocar l'inici d'un període de reducció de fons i interès en la investigació de la I.A. el qual es coneix com a "Hivern de la I.A.".

Després de més de 15 anys, la publicació d'un treball per part de *David E. Rumelhart, Geoffrey E. Hinton* i *Ronald J.Williams* i l'avenç en computació va fer tornar l'interès en les xarxes neuronals amb la presentació d'un nou algoritme d'entrenament. (6)

En el nostre dia a dia trobem I.A. per exemple en els motors de recerca de internet que estudien les nostres preferències per oferir-nos suggeriments o en assistents virtuals. De fet, la seva importància no recau únicament en la innovació i la comoditat sinó que realment ha permès avançar en molts camps com en el llançament d'eines per realitzar intervencions mèdiques a distància o sistemes capaços de reconèixer patrons patològics i diagnosticar malalties a través d'Aprenentatge Automàtic.

2.2. *Machine Learning* en la indústria tecnològica

Són múltiples les àrees en les quals es pot aplicar *Machine Learning* en una empresa industrial tecnològica. A continuació se'n llisten alguns exemples:

- En **robòtica autònoma** amb capacitat d'analitzar l'entorn on es mou i saber detectar obstacles. Per exemple, Amazon és una de les multinacionals que ha automatitzat els seus centres de logística en magatzems intel·ligents amb la introducció de robots capaços de calcular rutes per agafar paquets evitant col·lidir amb altres robots. (7)

Figura 4: Robots magatzem automatitzat de Amazon. Font: (40)

- En **inspeccions de qualitat directes** al producte mitjançant visió artificial.

- **Monitorització intel·ligent** de maquinària en producció amb capacitat d'aprendre la variabilitat que pot presentar la màquina mitjançant la generalització dels resultats d'auditories per la seva posterior validació.

- **Control de la maquinària de forma remota** mitjançant visió artificial.

- En **manteniment predictiu** mitjançant algoritmes d'aprenentatge que processen la informació de l'estat i les variables de màquines o estructures adquirida pels sensors i són capaços d'identificar valors anormals amb la finalitat de detectar possibles anomalies abans que es produeixin i perjudiquin el procés productiu.

- **Robots en producció** que aprenen a realitzar les tasques de muntatge. L'institut de *Fabricació IFF* i *Fraunhofer IPA* estan desenvolupant tecnologies per simplificar la programació del robot i facilitar les tasques de muntatge en el seu projecte *Rob-aKademl.* (8) Aquest projecte es basa en un bessó digital en un entorn de simulació física en el qual s'ensenya els moviments al robot per diferents processos d'assemblatge de manera que el robot explora els moviments que li provoquen, els planifica i els optimitza de manera que al final els va aprenent. A continuació es transfereix aquest aprenentatge a la realitat.

Figura 5: Projecte Rob-aKademl. Font: (41)

- En **ciberseguretat** desenvolupant algoritmes intel·ligents per l'anàlisi de malware o detecció i prevenció d'intrusos. Altres aspectes de la seguretat serien reconeixement facial a l'entrada de l'empresa i detecció del número de persones en una sala per verificar que no superi el límit sanitari COVID-19.

- En un departament d'**operacions** per facilitar l'anàlisi de dades i obtenir els gràfics d'interès automàticament en funció de l'anàlisi que se'n vulgui fer i les dades d'entrada.

- Realitzar **prediccions d'indicadors d'empresa** utilitzant dades històriques com la predicció de la demanda dels seus productes per la setmana següent o el valor econòmic en un determinat any.

Avantatges

Per una part, aplicar tecnologies de I.A. en una indústria permet l'automatització de processos, rutines o tasques que abans es realitzaven manualment o no aportaven valor afegit, la qual cosa implica els següents beneficis:

- Augment de la productivitat i de la eficiència global del sistema productiu (*OEE*). Major control del procés, el temps de cicle es redueix a un temps constant, la màquina pot operar les 24 hores del dia.
- Millora de la qualitat, fiabilitat i precisió. Al reduir-se la interacció humana, la possibilitat d'errors humans i la subjectivitat s'elimina i es pot mantenir un nivell d'homogeneïtat de procés.
- Major nivell de seguretat ja que l'operari no hauria de manipular ni situar-se en punts propers a entorns perillosos.

En particular, aplicar tecnologies de *Machine Learning* en una indústria presenta nombrosos avantatges com els que s'exposen a continuació:

- Ofereixen flexibilitat per adaptar-se a la variabilitat de les dades d'entrada i esquiven la rigidesa que poden presentar els sistemes de control basats en màquines d'estats o condicionals lògics sense perdre la robustesa.
- Increment de la competitivitat si s'aconsegueix produir més a menor cost amb alts nivells de productivitat, eficiència i qualitat.
- Facilita la presa de decisions de forma intel·ligent en casos en els quals s'aplica *Machine Learning* per anàlisi de dades o prediccions.

Per altra banda, se sent a dir que la introducció massiva de I.A. provocarà la pèrdua de molts llocs de treball. Segons un informe de l'institut *McKinsey Global*, prop de 375 milions de treballadors hauran de canviar de categoria ocupacional al 2030 degut a la disrupció provocada per la digitalització i robotització. (9) No obstant, això no s'hauria de veure com un punt negatiu sinó que per l'arribada de noves tecnologies sorgiran noves professions que crearan llocs de treball i els operaris que realitzaven aquestes tasques repetitives es podran dedicar a tasques que requereixin més atenció humana i siguin de més valor afegit.

2.3. Anàlisi del mercat

En aquest apartat es realitza un anàlisi a nivell quantitatiu i qualitatiu mitjançant dades i informes de companyies analistes i la pròpia percepció de l'estat actual del mercat per tal de realitzar una possible definició de necessitats actuals en la indústria tecnològica.

La companyia *Gartner* estima que el número d'empreses que utilitzen I.A. en el seu negoci va créixer un 270% entre 2015 i 2019. (10) Es preveu que en els pròxims anys continuï creixent a un ritme exponencial impulsat per les 4 tecnologies líders: *Deep Learning, Machine Learning*, *NLP* i visió per computador.

Tot i així, en la velocitat d'adopció de la I.A. hi influeixen diferents factors que poden actuar com a reptes per algunes empreses. En un informe elaborat per l'empresa *O'Reilly* al Febrer del 2021 (11) s'esmenten els següents:

1. Les barreres de més impacte són la falta de personal capacitat i la falta de dades de qualitat per al modelatge. No és sorprenent que la demanda d'experiència especialment en coneixements d'Aprenentatge Automàtic i científics de dades, hagi superat la oferta. No obstant, en els últims anys han sorgit noves titulacions i formacions especialitzades en aquest àmbit per la qual cosa es preveu que aquest aspecte no suposi un obstacle en els propers anys.
2. Algunes empreses es mostren reticents a la incorporació de la tecnologia per no tenir una percepció de valor de la I.A. en l'empresa, no trobar un cas d'ús o dificultat de la visualització del retorn de la inversió.
3. Altres obstacles de caràcter tècnic que no es consideren tant rellevants serien la infraestructura de desenvolupament, majoritàriament per les prestacions necessàries per executar models de *Machine Learning*, i l'ajust d'hiperparàmetres.

Per fer front els anteriors reptes, en l'estratègia d'adopció cal remarcar la importància de la formació no només del personal tècnic sinó també dels líders de negoci de manera que valorin el seu potencial de negoci i puguin analitzar les necessitats reals del mercat així com també els casos on la I.A. pugui aportar major valor. (12)

En línia amb l'estudi d'*O'Reilly*, es demostra que la majoria d'empreses que utilitzen I.A. corresponen a indústries tecnològiques. D'aquest subconjunt i d'acord amb el gràfic de la *Figura 6*, el 35% aplica I.A. de forma estandarditzada en producció i la resta té projectes en fase d'avaluació o considerava el seu ús.

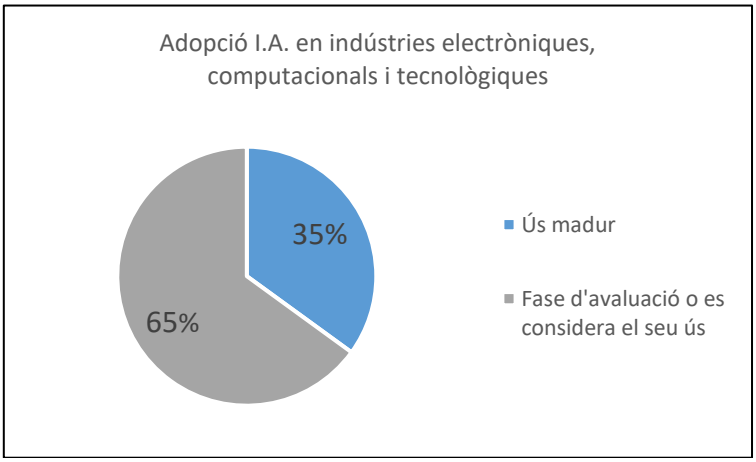


Figura 6: Gràfic estat de l'adopció de la I.A. en empreses al 2021. Font: (11)

Aquestes xifres reflecteixen el gran impacte tecnològic que està revolucionant l'entorn empresarial i mostren l'aparició d'una clara necessitat de transformació digital en tots els sectors industrials pels beneficis que aporta l'anàlisi i automatització intel·ligent. De fet, la filosofia en moltes empreses és la millora contínua basada en la revisió continuada dels processos operatius, reducció de costos i altres factors que en conjunt permetin l'augment del rendiment i la competitivitat. És per aquest motiu que es busca la introducció i innovació d'eines de treball i tecnologies més eficients.

En particular, en la indústria tecnològica de producte, la visió per computador és una de les tecnologies líders de la I.A. que ha experimentat una major evolució principalment per controls de qualitat automatitzats i robots amb visió artificial. En qualsevol línia de producció existeix un procés d'inspecció de qualitat del producte, més sofisticat o senzill però sempre n'hi ha un. Degut a que la satisfacció i retenció del client s'ha convertit en un dels objectius claus en les empreses, (13) s'ha detectat aquest procés com a crític.

2.4. Evolució dels controls de qualitat industrials

El control de qualitat industrial és el conjunt d'accions per les quals es detecten errors o mancances durant el procés de producció. L'objectiu és garantir que els productes finals compleixin els requisits mínims de qualitat imposats per l'empresa i sol·licitats pel client. A continuació s'exposa de forma breu l'evolució d'aquestes inspeccions:

- **Manual:**

Un operari s'encarrega de la supervisió del producte resultant d'un procés mitjançant la vista o amb eines per realitzar mesures com el peu de rei, voltímetre... Actualment moltes inspeccions encara es realitzen de forma manual.

Figura 7: Inspectora de qualitat de peces en una fàbrica de màquines de cosir, 1977. Font: (14)

A partir de l'any 1950, els controls de qualitat es van començar a automatitzar mitjançant visió artificial per les següents tasques:

- Sistemes d'inspecció visual per tal d'assegurar l'homogeneïtat dels productes, per exemple per garantir que les ampolles continguin la mateixa quantitat de líquid, identificar peces faltants en els productes o comprovar el correcte posicionament.
- Detecció de la presència de defectes en un producte resultants d'un error en un procés.
- Mesura de dimensions (profunditat, llargada, distància) , geometria, alineació i centratge de peces de forma immediata i amb una precisió micromètrica. Un exemple seria en la fabricació de cargols mitjançant metrologia 2D[5].

La unió dels dos conceptes visió artificial i control de qualitat dona com a resultat processos productius més efectius i eficients. S'eliminen les unitats defectuoses abans d'arribar al final de línia de producció la qual cosa es tradueix en un estalvi en materials, reducció de costos i energia per manipular la peça defectuosa en els processos posteriors. Tanmateix, un sistema de visió artificial amb prou resolució de càmera pot inspeccionar aquells detalls que podrien ser menyspreables per l'ull humà. A l'eliminar el contacte físic entre el sistema de mesura i la peça, s'evita el dany a la peça.

- **Visió artificial i programari convencional**:

S'assigna valors numèrics a tots els píxels d'una imatge en escala de grisos i es comparen amb uns patrons, característiques i marges prèviament programats. En la *Figura 8* es mostra una línia de producció de peces on es realitza un control de qualitat consistent en la mesura del radi de l'eix de la peça. A l'esquerra apareix la interfície de monitorització que mostra la imatge capturada per la càmera, indica el valor de la mesura obtinguda i notifica el resultat de la inspecció.

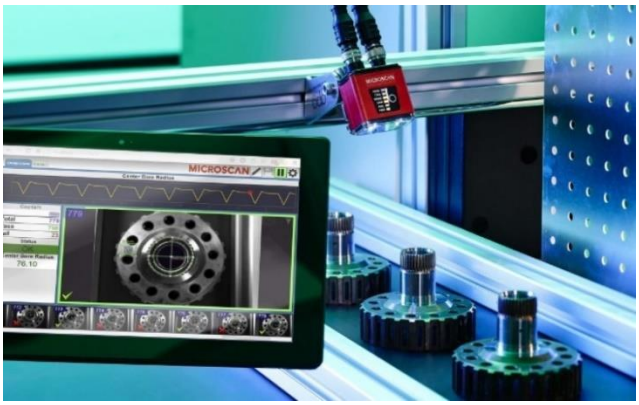


Figura 8: Sistema de visió artificial per control de qualitat amb interfície de monitorització. Font: (15)

[5] *Metrologia 2D: sistema de control dimensional per assegurar toleràncies de producció.*

- **Visió artificial i *software* d'Aprenentatge Automàtic:**

Presenta la capacitat d'aprendre patrons i característiques a partir d'un conjunt d'exemples de la peça sense programar-les explícitament la qual cosa és especialment interessant en peces que presentin molta variabilitat com les peces resultants d'un procés de soldadura o fabricació de circuits integrats.

Per exemple, en la *Figura 9* es mostren unes peces resultants d'un procés de foneria en el qual es poden generar molts tipus de defectes com picadures, encongiment, defectes del motlle, rebaves...

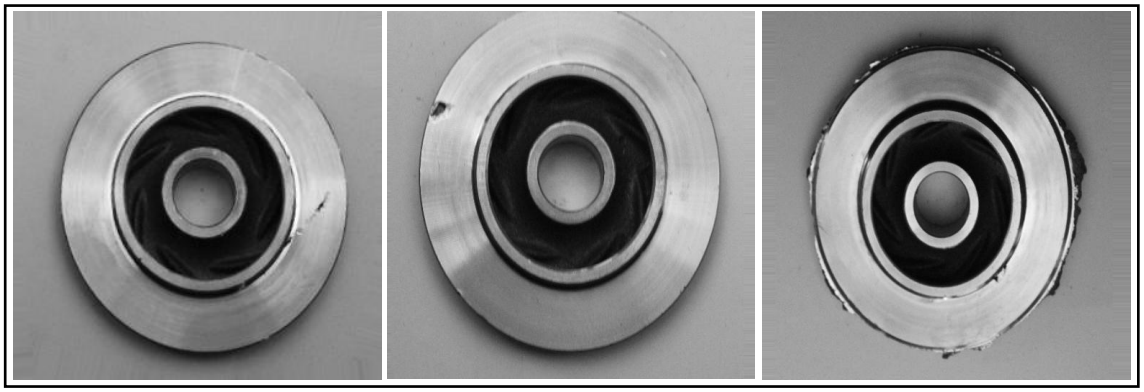


Figura 9: Peces resultants d'un procés de foneria. Font: (16)

La introducció de *software* d'Aprenentatge Automàtic suposaria l'automatització de les següents tasques:

- Tasca d'extracció de característiques per part d'un enginyer de *software* el qual, abans de començar a programar, ha de realitzar el pas previ de caracterització del procés productiu (en certa manera elabora un model) per tal d'identificar els criteris i característiques que diferencien una peça OK de una NOK.

- Tasca de classificació manual per part d'un operari especialitzat en el procés productiu en els casos en els quals és impossible o molt complicat programar la classificació mitjançant lògica condicional.

Mentre que la visió humana es coneix per ser la millor en la interpretació qualitativa d'escenes complexes no estructurades, la visió artificial destaca en les mesures quantitatives d'escenes estructurades per la seva velocitat, precisió i reproductibilitat. No obstant, amb la introducció de tècniques d'Aprenentatge Automàtic, seria capaç d'analitzar escenes cada vegada més complexes.

3. MARC TEÒRIC: XARXES NEURONALS ARTIFICIALS

Les xarxes neuronals artificials són models computacionals inspirats en el comportament de les neurones biològiques fent la seva analogia a operacions matemàtiques. Es tracta d'un algoritme de *Machine Learning* per presentar un paradigma[6] d'aprenentatge i processament automàtic.

S'estima que el cervell humà conté més de cent mil milions de neurones. Una xarxa neuronal biològica està composta per un grup de neurones connectades químicament i el seu objectiu principal és desenvolupar operacions de síntesi i processament d'informació relacionades amb els sistemes biològics. Consten de les dendrites que reben els impulsos d'entrada, el cos de la neurona on es processen aquests estímuls i l'axó que porta la sortida de la neurona a les dendrites de la següent neurona. Les connexions entre neurones anomenades sinapsis tenen diferents pesos assignats de manera que poden convertir-la en una connexió excitadora o inhibidora. (17)

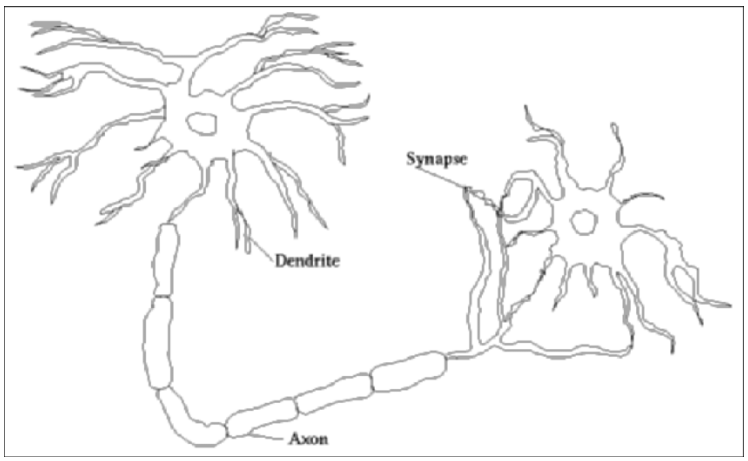


Figura 10: Representació aproximada d'una neurona biològica. Font: (17)

Vivim en un univers complex i ple de soroll; no obstant, la intel·ligència humana ha aconseguit entendre aquesta realitat caòtica en una cerca de les analogies i semblances que s'hi amaguen. En les primeres pàgines d'aquest treball es fa referència a la següent frase del matemàtic Edward Lorenz: "*Llamamos "caos" al orden que todavía no comprendemos*". (18) De fet, detectar patrons i explotar la capacitat d'observar el món de forma simplificada ha permès el desenvolupament de la nostra espècie. (2) La realitat s'ha reconstruït a través de models[7] en totes els àmbits i disciplines: mapes geogràfics, equacions físiques, fórmules, partitures... És per aquest motiu que la cita de Lorenz té una certa relació amb la tasca que desenvolupen les xarxes neuronals artificials les quals elaboren models simplificats d'una realitat que volem entendre.

---

[6] *Paradigma d'aprenentatge: Mecanismes o algoritmes pels quals les xarxes neuronals aprenen.*

[7] *Model: Representació conceptual d'un fenomen complicat o no gaire conegut que permet donar-li una explicació matemàtica per analogia. Estableix la relació entre les variables d'entrada i les de sortida. Un model ha de buscar l'equilibri entre aproximar-se a representar exactament la realitat i ser simple per poder-lo utilitzar.*

Segons la topologia de la xarxa, les xarxes neuronals es classifiquen en: perceptró, xarxa neuronal multicapa, xarxa neuronal convolucional... (2)

3.1. Tipus de xarxes neuronals

3.1.1. El perceptró

La neurona és la unitat bàsica de processament. El càlcul que realitza internament és la suma ponderada de cada variable d'entrada (*x1, x2, xn*) amb els seus corresponents pesos (*w1, w2, wn*) i afegint-t'hi la suma del terme independent associat a aquella neurona anomenat *bias* (*b*). Els pesos són els valors que definiran la intensitat amb la qual cada variable d'entrada afecta a la neurona. Addicionalment, al resultat d'aquesta suma se l'hi aplica una funció d'activació.

$$y = f_{activation}(w_1 \cdot x_1 + w_2 \cdot x_2 + w_n \cdot x_n + b)$$

$$y = f_{activation}(b + \sum_{i}^{n} w_i \cdot x_i)$$

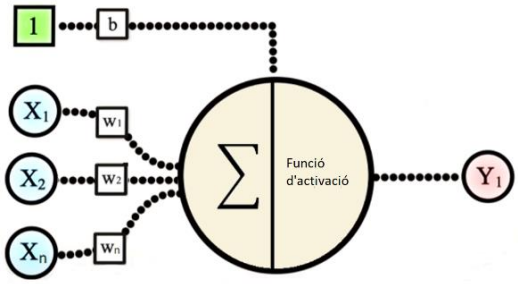


Figura 11: Perceptró. Font: (2)

La funció d'activació es comporta com un filtre que escala el resultat de la suma ponderada per tal d'aconseguir un valor comprès dins d'un rang d'interès. Com que són funcions no lineals ajuden a modelar funcions corbes augmentant la capacitat d'expressió de la xarxa neuronal. De forma simplificada, aquesta funció seria binària (un esglaó), la neurona s'activa o no. D'entre les funcions d'activació més utilitzades es troba la funció sigmoide la qual comprimeix el resultat de la suma ponderada a un rang d'entre 0 i 1, especialment útil per expressar probabilitat. La funció tangent hiperbòlica es tracta d'una funció sigmoide amb diferent rang i la funció *ReLu* (*Rectified Linear Unit*) permet el pas dels resultats positius i assigna tots els valors negatius a zero.

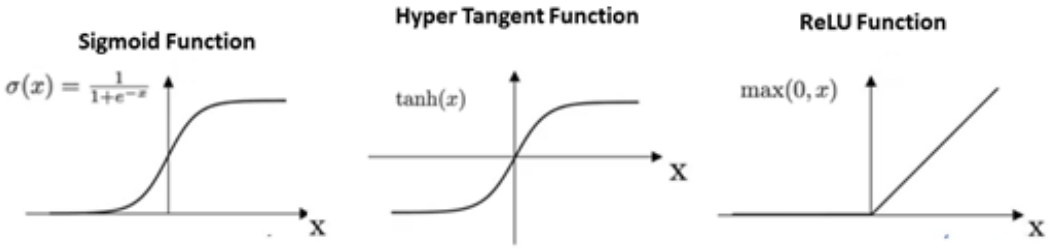


Figura 12: Funcions d'activació. Font: (19)

Significat geomètric:

Per una sola dimensió, una sola variable d'entrada x_1, es realitzaria un model de regressió lineal simple. La tasca de la neurona seria la de trobar la recta que millor s'ajusti a les dades bidimensionals, és a dir, trobar els paràmetres w_1 i b:

$$y = w_1 \cdot x_1 + b$$

No obstant, un fenomen sol estar afectat per més d'una variables d'entrada. Cada una representa una dimensió i es pot entendre com una característica de la realitat que es vol modelar. En aquest cas, la neurona realitzaria un model de regressió lineal múltiple i la seva tasca seria trobar l'hiperplà que millor s'ajusti al núvol de dades tridimensionals o N-dimensionals. Un exemple seria trobar el model que relacioni el preu d'un ordinador portàtil amb les seves prestacions.

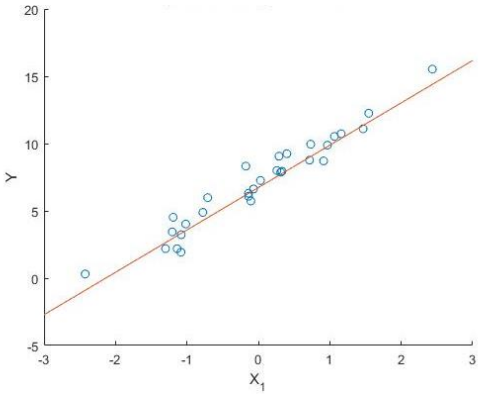


Figura 13: Regressió lineal simple. Font: (20)

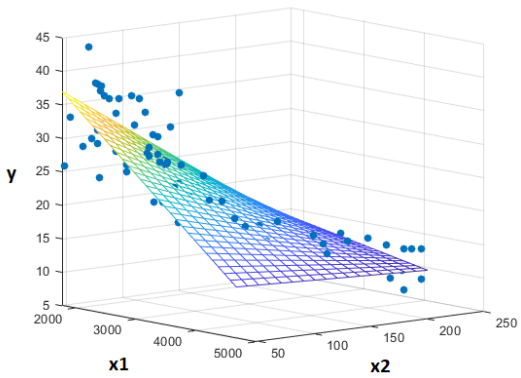


Figura 14: Regressió lineal múltiple. Font: (20)

3.1.2. Xarxes neuronals multicapa

La unió de diverses neurones seqüencialment forma xarxes neuronals. Es solen conèixer com a "*fully-connected*" ja que cada neurona de la capa anterior es connecta a totes les neurones de la capa posterior de manera que la sortida d'una neurona és l'entrada de la següent i s'aconsegueix aprendre de forma jerarquitzada.

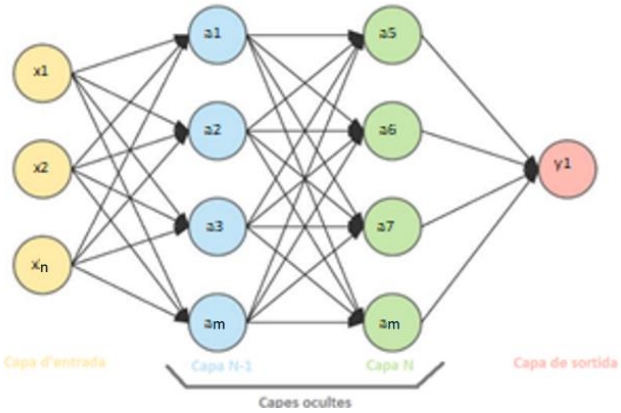


Figura 15: Xarxa neuronal multicapa. Font: elaboració pròpia

La *Figura 15* mostra una xarxa neuronal amb una capa d'entrada de 3 neurones, dues capes ocultes de 4 neurones i una única neurona a la capa de sortida. En aquest cas les operacions que s'hi realitzen són en essència un conjunt de sumes ponderades que es poden representar de forma vectorial:

$$Y = f_{activation}(W \cdot X + b)$$

$$\begin{aligned} Nomenclatura: \quad & X \text{ és la matriu d'entrada} \\ & W \text{ és la matriu de paràmetres} \\ & b \text{ és la matriu de bias} \\ & Y \text{ és la matriu de sortida} \end{aligned}$$

Significat geomètric:

En qualsevol problema de classificació, la tasca de les xarxes neuronals serà la de trobar els valors dels paràmetres de la neurona que tracin una frontera entre els grups que es volen classificar. La necessitat de combinar neurones ve per les limitacions del perceptró. Per exemple, una sola neurona pot classificar i modelar informació d'una porta AND i OR, tanmateix, no pot modelar una porta XOR ja que no pot separar amb una sola línia un núvol de punts distribuïts d'aquesta manera. La solució és afegir una altra neurona en la mateixa capa per obtenir una altra recta de separació.

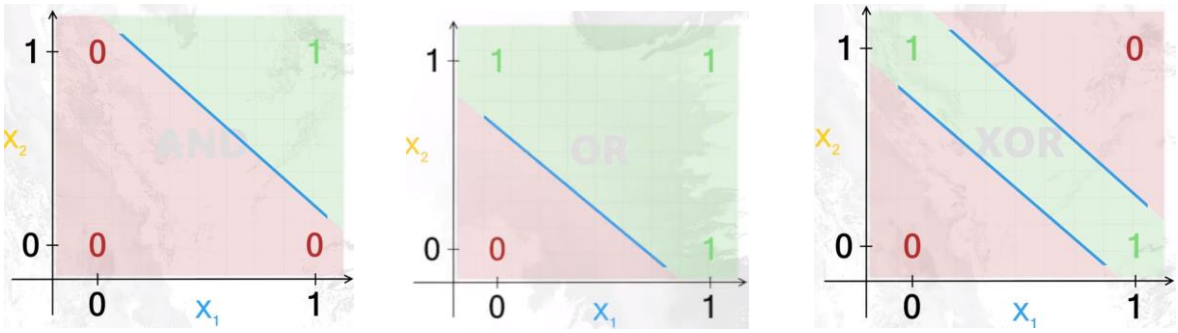


Figura 16: Model porta AND, OR, XOR. Font: (2)

3.1.3. Xarxes neuronals convolucionals

Les xarxes neuronals convolucionals són una variant de les xarxes neuronals multicapa dissenyades per emular el comportament del còrtex visual del cervell humà i es solen conèixer com a CNN de l'anglès *Convolutional Neural Networks*. Destaquen en aplicacions de processament d'imatges i, per tant, en aplicacions de visió artificial. La importància d'aquestes xarxes ve per la seva capacitat de desxifrar els patrons més complexos en grans *datasets* d'imatges. (2)

Es poden entendre les capes d'una CNN com neurones organitzades en 3 dimensions (amplada, altura i profunditat). Normalment les CNN es representen com un embut per la seva estructura de profunditat on la imatge inicial es va comprimint espacialment, la seva resolució disminueix, a la vegada que augmenta el seu gruix degut a l'augment de mapes de característiques.

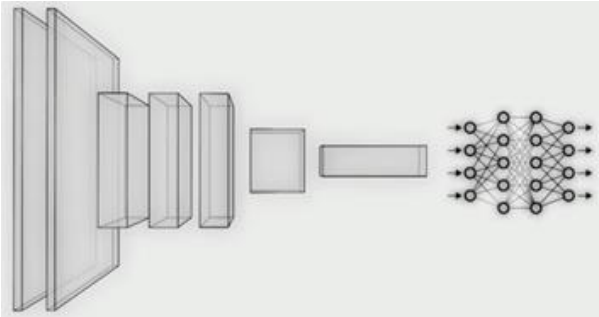


Figura 17: Representació abstracta d'una CNN. Font: (2)

Les CNN mitiguen els problemes de l'arquitectura multicapa i ofereixen nombrosos avantatges per al reconeixement d'imatges. En la *Taula 1* es mostra la comparativa entre les dues tipologies. (21)

	Xarxa neuronal multicapa	**Xarxa neuronal convolucional**
Connexions entre capes	*Fully-connected*	Connectivitat local entre neurones de capes adjacents (cada neurona està connectada només a una petita regió del volum d'entrada).
Eficiència computacional	Baixa, a cada capa els paràmetres incrementen de forma exponencial.	Elevada. - S'estalvia moltes connexions i paràmetres. - Utilitza els mateixos pesos de filtre per totes les convolucions d'un mapa de característiques.
Dades d'entrada: píxels de la imatge	- No se li dona importància a la posició de cada píxel dins la imatge. - Problema de la dimensionalitat: a cada capa el número de pesos i neurones augmenta i resulta impossible resoldre-ho computacionalment amb imatges de major resolució.	Té en l'estructura espacial. (De fet, el valor d'un píxel està molt lligat al valor dels píxels veïns i és el fet que permet que sorgeixen estructures i formes.)
Paràmetres que ha d'aprendre a ajustar la xarxa neuronal	Pesos i *bias*.	Pesos dels filtres *kernel*.

Taula 1: Comparativa entre xarxa neuronal multicapa i convolucional. Font: elaboració pròpia

Operacions que es realitzen en una CNN

Una imatge es pot representar com una matriu de píxels amb valors entre 0 i 255 en 3 canals (RGB) per imatges de color o en 1 canal per imatges en escala de grisos. El número total de neurones a la capa d'entrada serà el número total de píxels de la imatge. Per exemple, per una imatge en escala de grisos amb 28x28 píxels d'altura i amplada, el número de neurones d'entrada seria 784.

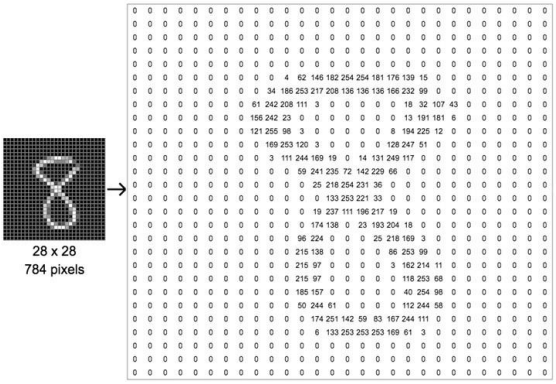


Figura 18: Representació d'una imatge en escala de grisos com una matriu de valors. Font: (22)

Abans d'alimentar la xarxa s'ha de realitzar una normalització d'aquests valors mitjançant la divisió del valor de cada píxel entre 255, amb el qual s'obté un valor entre 0 i 1. Sovint és convenient omplir de zeros el voltant de la matriu d'entrada per controlar la mida espacial de la matriu de sortida de la primera capa convolucional amb la tècnica *same padding*.

En la majoria de CNN es realitzen dues operacions principals, una operació matemàtica anomenada convolució[8] i una segona operació anomenada *pooling*. (21)

- **Convolució:**

Aplica diferents filtres anomenats *kernel* per escanejar la imatge en grups de píxels adjacents per tal d'obtenir imatges transformades anomenades mapes de característiques. L'operació matemàtica consisteix en un producte escalar amb els valors dels paràmetres dels filtres i el valor dels píxels. Cada un d'aquests filtres serà un detector d'una característica en concret depenent dels seus paràmetres. En la *Figura 19* es mostra un exemple d'un filtre 3x3 que, pels valors dels seus pesos, s'activa quan troba diferències de contrast i serveix per detectar vores verticals. D'aquesta manera s'obté una imatge on cada píxel blanc indicarà la presència de la característica.

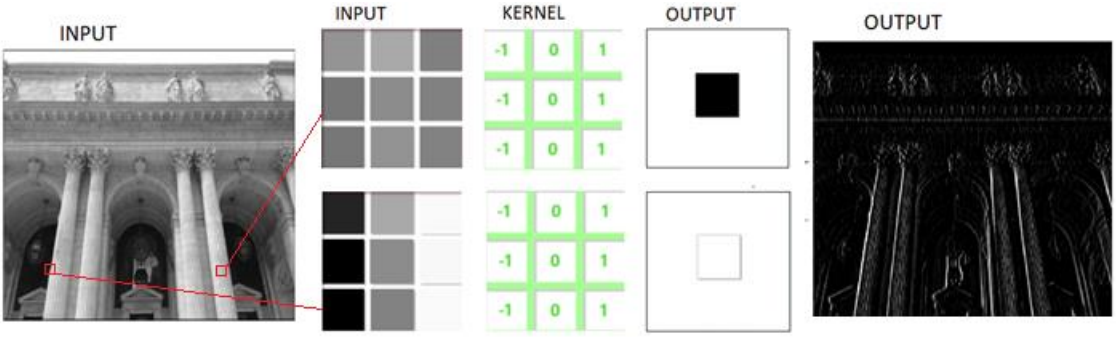


Figura 19: Resultat d'una convolució. Font: (23)

El resultat d'aplicar *n1* filtres *kernel* en una primera convolució seria *n1* matrius o mapes de característiques. En línia amb l'exemple anterior del *dataset* numèric, la sortida d'aquesta primera capa convolucional serà *n1* matrius de 28x28x1 píxels cada una aplicant la tècnica *same padding*. Agafant un valor de *n1* igual a 32 filtres, correspon a 25088 neurones en aquesta capa convolucional oculta. Des d'aquest punt de vista, cada neurona realitza una operació de convolució amb filtre *kernel* i dona com a resultat un valor (un píxel).

[8] *Convolució: operació matemàtica que transforma dues funcions en una tercera funció que representa la magnitud de superposició de les dues funcions originals.*

Sovint, després de l'operació de convolució s'aplica una funció de no linealitat com la funció d'activació *ReLu* amb la finalitat de convertir els valors negatius en zeros (píxel de color negre) i accentuar més l'activació d'una característica en un píxel.

Després d'aquest primer nivell de profunditat s'han detectat les primeres característiques, a mesura que s'augmenta en profunditat i s'afegeixen més capes de convolució s'aconsegueixen formes més complexes. A cada capa les operacions es fan més potents, es converteix una regió de 9 píxels en un sol píxel d'informació i cada vegada s'accedeix a més informació espacial de la imatge inicial. En realitat, una operació de convolució només pot detectar patrons senzills com vores, superfícies planes, textures, canvis de contrast. Tanmateix, el fet de realitzar deteccions sobre deteccions de les capes anteriors permet construir patrons cada vegada més complexos tal i com es mostra en la *Figura 20*.

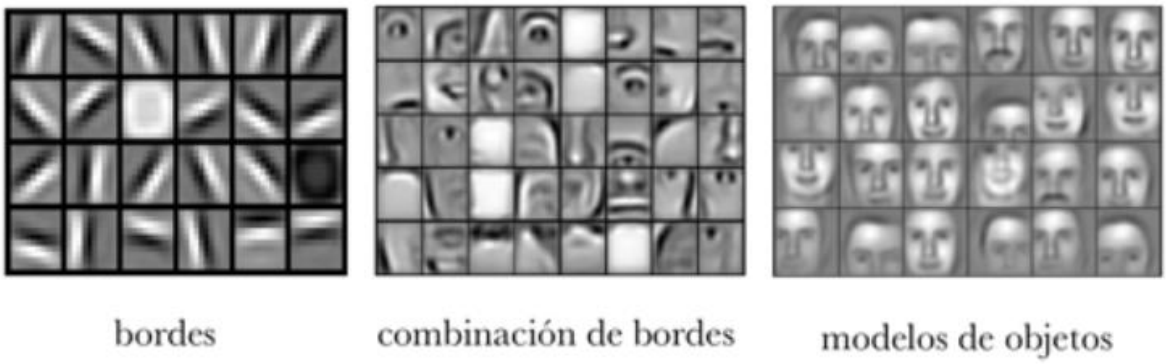


Figura 20: Mapes de característiques. Font: (24)

- ***Pooling*:**

Si es realitza una segona convolució, el número de neurones de la segona capa se n'aniria pels núvols i això implica major processament. Per evitar-ho se sol introduir una capa de *pooling* després de cada capa de convolució (*max pooling* o *mean pooling*) amb la finalitat de reduir la mida dels mapes de característiques realitzant una operació de mostreig sobre els valors d'una regió contigua. Això es realitza de tal manera que prevalguin les característiques més importants que ha detectat cada filtre.

Continuant amb l'exemple anterior i suposant que s'escull *max pooling* de mida 2x2, per cada mapa de característiques es fan grups de 2x2 píxels contigus i es transforma en un píxel resultant del valor més alt. En aquest cas, utilitzant 2x2 la imatge resultant és reduïda a la meitat i passarà de 28x28 píxels a 14x14 píxels. Finalment quedaran 32 imatges de 14x14 píxels, passant de 25088 neurones a 6272 que segueixen emmagatzemant la informació més important.

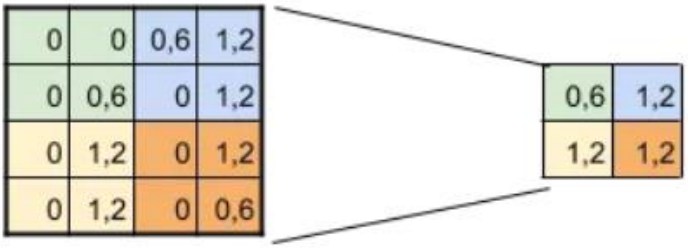


Figura 21: Operació max pooling 2x2 sobre una imatge de 4x4 píxels. Font: (21)

A partir d'aquí, es realitzen més convolucions i *pooling*. Si es realitza una segona convolució de la sortida del resultat *pooling*, el número de filtres a aplicar pot ser diferent de *n1*. Agafant *n2* filtres *kernels*, a la sortida de la capa *max pooling* s'obtindria un conjunt de *n2* mapes de característiques de 7x7 píxels. Si es realitza una tercera convolució, a la sortida de la capa *max pooling* s'obtindran *n3* mapes de característiques de 3x3 píxels. Normalment els valors de *nx* creixen en potències de 2 a cada capa.

Finalment, la sortida de l'última capa d'operacions es connecta a una xarxa neuronal *fully-connected,* normalment amb una sola capa oculta, per realitzar una classificació dels mapes de característiques. Degut a que l'última capa oculta de *max pooling* es coneix com a tridimensional per prendre la forma 7x7x *n2*, s'aplana per convertir-se en una capa de neurones plana.

A continuació es mostra la xarxa neuronal convolucional corresponent als exemples anteriors:

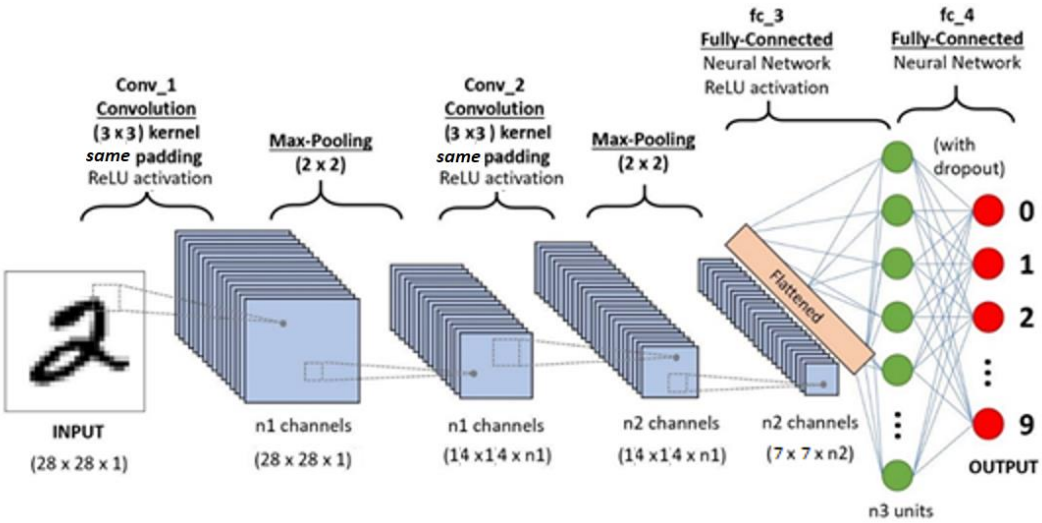


Figura 22: Exemple de l'estructura d'una xarxa neuronal convolucional. Font: (25)

La configuració de l'última capa de la xarxa neuronal *fully-connected* depèn del problema de classificació:

- **Més de 2 classes** i l'entrada només pot pertànyer a una sola classe (sortides excloents):
  Funció d'activació: *softmax*.
  Número de neurones: igual al número de classes.
  Per exemple, si es vol classificar una imatge del *dataset* numèric MNIST, el número de classes són 10 (una per cada dígit). El resultat serà un vector de 10 valors amb la probabilitat que la imatge correspongui a cada una de les classes. Posteriorment, es pot programar per tal que tingui un format conegut com a *one-hot-encoding*, un bit per cada classe de manera que es tingui un sol bit alt (1) i tots els altres baixos (0).

- **2 classes:**
  Funció d'activació: sigmoide.
  Número de neurones: 1 neurona.
  Un sol valor de sortida que indica la probabilitat (entre 0 i 1) que la imatge correspongui a una classe.

3.2. Paradigma d'aprenentatge supervisat

Es tracta de trobar la combinació ideal de paràmetres per obtenir el model que generalitzi millor el conjunt de dades d'entrada. És la pròpia xarxa neuronal qui aprèn a ajustar aquests paràmetres mitjançant un algoritme. El procés pel qual es van reajustant els paràmetres del model s'anomena optimització o entrenament.

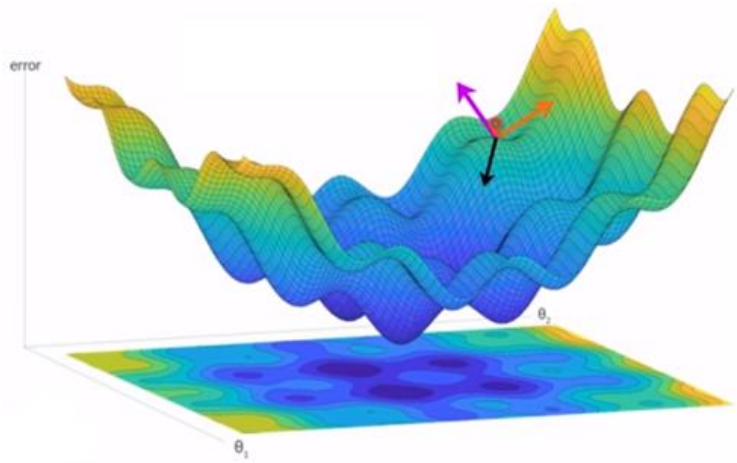
Es definirà una funció que representi quin és l'error per cada una de les combinacions possibles dels paràmetres, s'anomenarà funció de cost. Com que hi ha més d'un paràmetre de model, la funció serà multidimensional. L'objectiu serà trobar el seu mínim global i per fer-ho s'utilitzarà el mètode del Descens del Gradient.

3.2.1. Descens del gradient

El Descens del Gradient és un mètode iteratiu que no dona directament els paràmetres que es necessiten sinó que a poc a poc s'apropará al mínim error. La metodologia que es segueix és la següent: (2)

1r: Avaluar la inclinació per trobar el major pendent en la funció de cost:

Quan s'inicialitza la xarxa neuronal, els valors dels paràmetres son aleatoris de manera que se situa en un punt aleatori de la funció de cost. Es realitzen les derivades parcials de l'error en funció de cada paràmetre en aquest punt. El vector que conté les pendents per cada una de les dimensions de la funció és el vector Gradient i el descens d'aquest és:



$$\begin{bmatrix} \frac{\partial error}{\partial \theta_1} \\ \frac{\partial error}{\partial \theta_2} \end{bmatrix} = \nabla f$$

Figura 23: Funció de cost. Avaluació de la inclinació. Font: (2)

2n: Avançar una distància en aquella direcció (α):

$$\theta := \theta - lr \cdot \nabla f$$

$$\begin{aligned} Nomenclatura:\ & \theta\ és\ un\ paràmetre\ (pes\ o\ bias) \\ & lr\ és\ el\ ràtio\ d'aprenentatge \\ & \nabla f\ és\ el\ vector\ gradient \end{aligned}$$

3r: Tornar a repetir el procés en la nova posició:

Iterar el procés fins arribar a una zona on moure's ja no suposi una variació notable del cost (pendent sigui aproximadament zero).

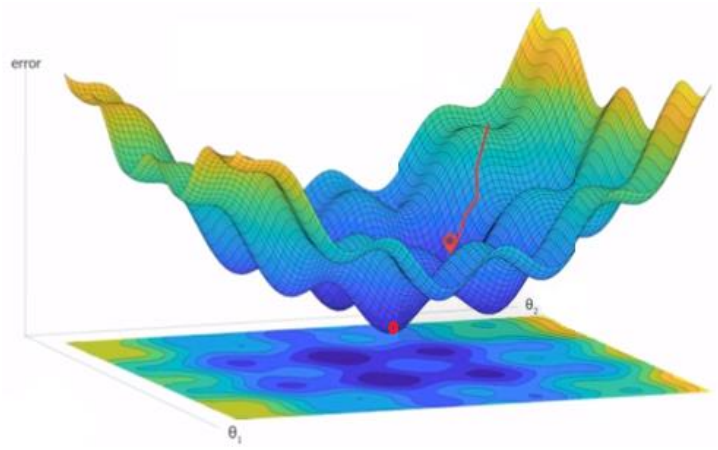


Figura 24: Funció de cost. Apropant-se al mínim de la funció fent ús del descens del gradient. Font: (2)

3.2.2. Backpropagation

Per emprar el mètode del descens del gradient es necessiten les derivades parcials pero no es coneix a priori la forma de la funció de cost pel conjunt de dades d'entrada degut a la impossibilitat que la xarxa neuronal arribi a provar tots els hiperparàmetres possibles i a tabular el seu corresponent error generat. *Backpropagation* és un mètode que s'empra per a calcular el vector gradient fent ús de la lògica de retropropagació de l'error cap enrere. Aquest algoritme considera la xarxa neuronal com una cadena de responsabilitat per saber quina part de responsabilitat té cada neurona en el resultat final, la qual cosa té sentit perquè l'error de les capes anteriors depèn directament de l'error de les capes posteriors.

3.2.3. Hiperparàmetres d'entrenament

Els hiperparàmetres d'un model fan referència als valors de les configuracions durant el procés d'entrenament. És crucial trobar els valors òptims dels hiperparàmetres per aconseguir un model que minimitzi la funció de cost de la forma més rendible. Aquests valors es determinen per part d'un científic de dades, a diferència dels paràmetres que aprèn la xarxa neuronal a partir de les dades.

A continuació es mostren els principals hiperparàmetres en detall:

Ràtio d'aprenentatge o learning rate (lr)

És la longitud del pas a cada iteració i es pot entendre com la velocitat de descens per una hipersuperfície. Té sempre un valor positiu, sovint en l'interval entre 0 i 1. Es recomana un valor estàndard d'entre 0,001 i 0,01 tot i que dependrà de les curvatures i plegaments de cada funció de cost que no es poden determinar a priori.

La problemàtica d'utilitzar un valor de *lr* fix resideix en el fet que si no es determina el valor exacte de longitud, el punt mai acabarà d'arribar al mínim de la funció de cost (no convergirà). La solució a aquesta problemàtica és usar un *lr* adaptatiu que tingui capacitat d'ajustar-se a cada iteració de manera que s'estableixi un *lr* incial per les primeres iteracions i a continuació, el valor va disminuint de forma uniforme quan s'apropi al mínim per garantir la convergència.

Mida del lot (batch_size)

Número de mostres per les quals s'actualitzen els paràmetres del model. Un conjunt d'imatges es pot dividir en un o més lots. Per exemple, suposant que el *dataset* d'entrenament conté 900 imatges i s'estableix un valor de *batch_size* igual a 100, l'algoritme pren les primeres 100 mostres i entrena la xarxa. A continuació, pren les segones 100 mostres i torna a entrenar la xarxa, així successivament fins que s'hagin propagat totes les imatges.

Quan el lot té la mida d'1 imatge, l'algoritme d'aprenentatge s'anomena descens del gradient estocàstic; quan el lot té la mateixa mida que el *dataset* s'anomena descens del gradient per lots i quan el lot té una mida inferior a la mida del *dataset* s'anomena descens de gradient de mini-*batch*.

Per la majoria de casos és habitual i recomanable utilitzar el descens de gradient en mini lots amb valors potència de dos com 32, 64... (26)

Èpoques (epochs)

Número de vegades que l'algoritme s'entrena amb el conjunt de dades d'entrenament, es pot establir en un valor enter entre 1 i infinit i sol estar relacionat amb la diversitat de les dades. (27)

S'anomena iteracions al número de lots necessaris per completar una època. El motiu pel qual s'entrena la xarxa en més d'una època resideix en el fet de tenir un conjunt de dades limitat i estar utilitzant el descens del gradient com a procés iteratiu.

Funció de pèrdua (loss_function)

La funció de pèrdua expressa l'error de les prediccions obtingudes per la xarxa neuronal respecte les etiquetes reals. Matemàticament es pot entendre com el valor de la coordenada *y* del punt vermell de la *Figura 23* del subapartat anterior.

Les funcions més habituals són les següents: (28)

- Error quadràtic mig (*MSE*):
  Normalment s'utilitza per models de regressió lineal amb una configuració de la capa de sortida del tipus una neurona amb funció d'activació lineal.

$$Error = \frac{1}{n} \cdot \sum_{i=1}^{n} \left(y_r - y_p\right)^2$$

$Nomenclatura$: y_r $és\ valor\ real\ de\ sortida$
y_p $és\ el\ valor\ resultant\ de\ la\ predicció\ de\ la\ xarxa\ neuronal$
n $és\ el\ número\ de\ mostres\ per\ iteració$

- Pèrdua d'entropia creuada binària o logarítmica (*binary_crossentropy*):
  S'utilitza en models de classificació binària amb una configuració de la capa de sortida del tipus una neurona amb funció d'activació sigmoide. Cada probabilitat predita es compara amb el valor de sortida real (1 o 0) i es calcula una puntuació que penalitza la probabilitat en funció de la distància al valor esperat. La penalització és logarítmica i ofereix una puntuació petita per diferències petites (0,1 o 0,2) i una puntuació enorme per diferències grans (0,9 o 1,0). La *Figura 25* mostra que l'error es dispara a mesura que la probabilitat predita té valors propers a 0 per una etiqueta real de 1.

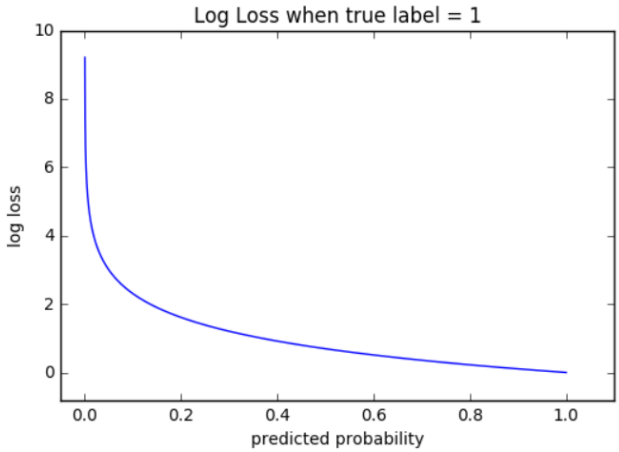


Figura 25: Error en funció de la predicció en probabilitat quan l'etiqueta real és 1. Font: (28)

$$Error = -\frac{1}{n} \cdot \sum_{i=1}^{n} y_r \cdot \log\big(p(y_r)\big) + (1 - y_r) \cdot \log\big(1 - p(y_r)\big)$$

$Nomenclatura$: y_r $és\ l'etiqueta\ real\ binària\ per\ una\ classe\ (1\ o\ 0)$
$p(y_r)$ $és\ la\ probabilitat\ predita\ de\ pertànyer\ a\ la\ classe$
n $és\ el\ número\ de\ mostres\ per\ iteració$

La fórmula és molt senzilla d'entendre, quan l'etiqueta real és 1, la segona meitat de la funció desapareix i l'error és el resultat del logaritme de $p(y_r)$. Quan l'etiqueta real és 0, la primera meitat es redueix a zero i l'error és el resultat del logaritme de $1 - p(y_r)$.

- Pèrdua d'entropia creuada categòrica (*categorical_crossentropy*):
  S'utilitza en models de classificació per múltiples classes amb una configuració de la capa de sortida del tipus una neurona per cada classe amb funció d'activació *softmax*. Es basa en pèrdua logarítmica igual que el punt anterior però per diferents classes.

Optimitzadors (optimizers)

Són algoritmes per optimitzar el descens del gradient i que generalment incorporen un ràtio d'aprenentatge adaptatiu. A part de poder-se configurar el valor inicial de *lr*, també es poden configurar altres paràmetres més complexos com el ràtio de caiguda en el descens del gradient.

L'optimitzador més utilitzat és *Adam* ja que s'ha demostrat en resultats empírics que aconsegueix bons resultats ràpidament per qualsevol tipologia de model. En concret, té la capacitat de descendre per la funció de cost evitant els mínims locals i arribant a convergir en el mínim global. (29)

Per tal de fer-ho més entenedor, a continuació es mostra un exemple molt senzill de la seqüència d'aprenentatge per la classificació binària d'un *dataset* amb 4 mostres, un valor de *batch_size* de 2 i una funció de pèrdua del tipus *binary crossentropy*.

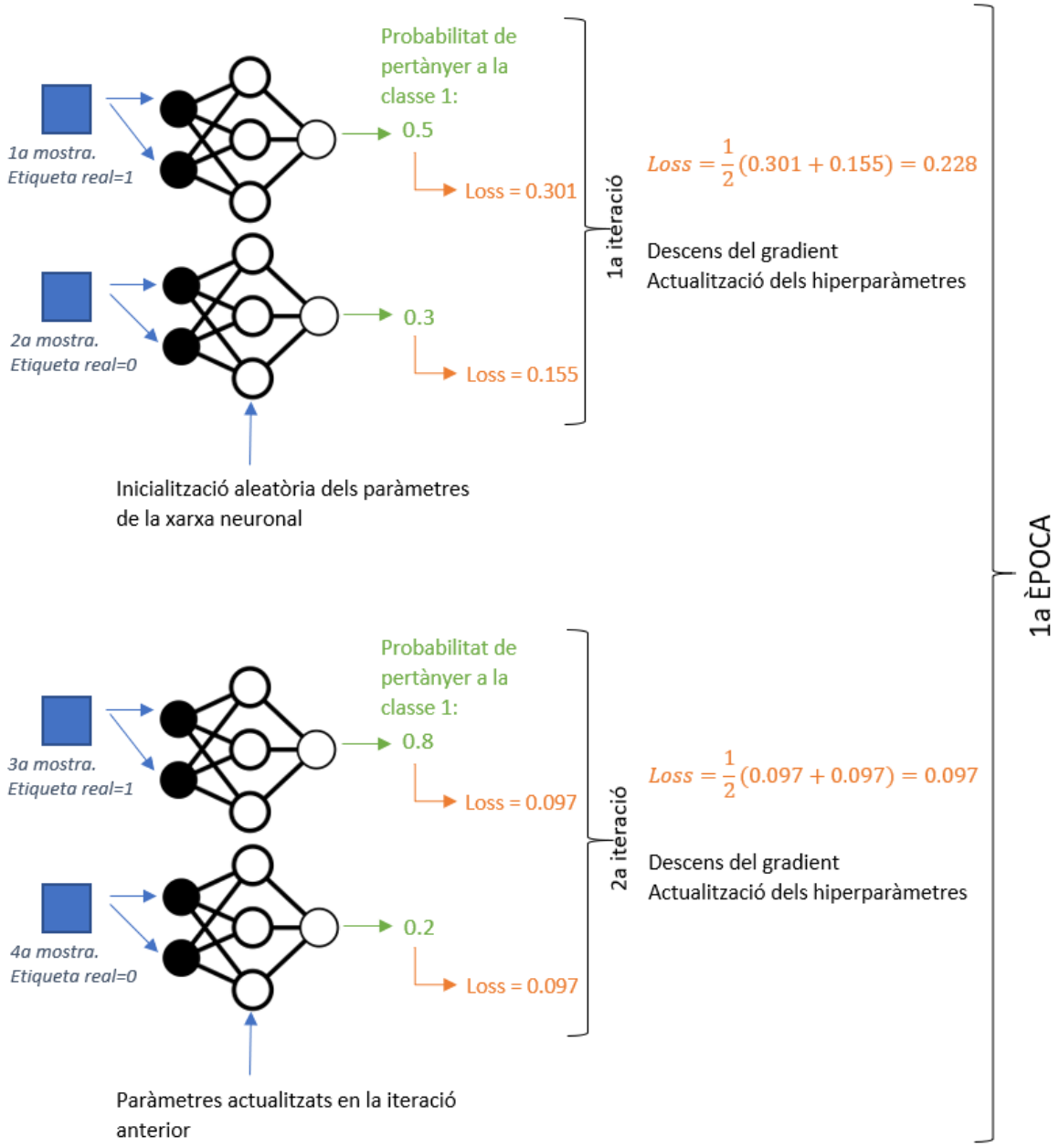


Figura 26: Exemple molt senzill de la seqüència d'aprenentatge d'una xarxa neuronal. Font: elaboració pròpia

Altres hiperparàmetres que s'han d'establir manualment són: el número de capes ocultes, número de neurones en cada capa oculta, funcions d'activació de cada neurona... En el cas de CNN són: número de capes convolucionals, número i mida dels filtres *kernel*, mida del *max-pooling*, configuració de *padding*...

3.2.4. El problema del *overfitting*

En *Machine Learning*, una de les principals problemàtiques tècniques és evitar que el model caigui en *overfitting* (sobreajust).

Es coneix com a *overfitting* el fenomen que es produeix quan durant l'entrenament s'ajusta tant bé als detalls de les dades d'entrenament (com si els memoritzés) que és incapaç d'aprendre i generalitzar per resoldre casos nous. Aquest fenomen apareix quan el model té un excés de flexibilitat i també modela el soroll de les dades d'entrenament. Un exemple seria el problema de la gestió de la variabilitat d'una màquina, es poden elaborar models tant adaptats a la seva operativa concreta que perden la seva validesa quan es produeixen petits canvis en la configuració o quan es prova en una altra màquina de la mateixa família. Això no interessa, l'objectiu és aconseguir models robustos enfront nous exemples.

En l'extrem oposat, el model també pot presentar underfitting (subajust) quan no se li dona la flexibilitat suficient per adaptar-se al núvol de punts de dades. (2)

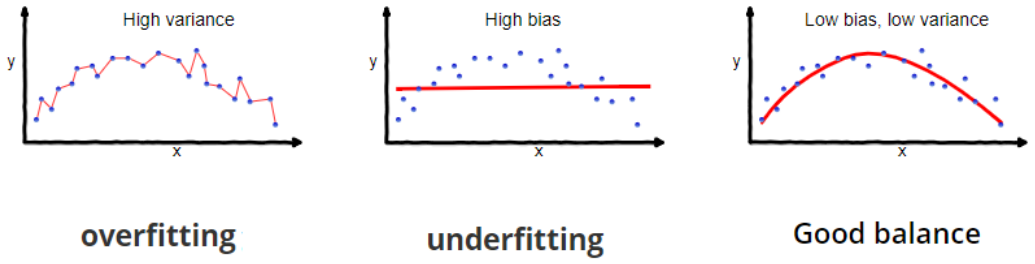


Figura 27: Overfitting i underfitting. Font: (30)

La *Figura 28* mostra un clar exemple d'*overfitting* en el qual a partir de les mil èpoques d'entrenament el rendiment del model per les dades de validació es comença a deteriorar mentre que el rendiment per les dades d'entrenament és del 100% (error igual a zero).

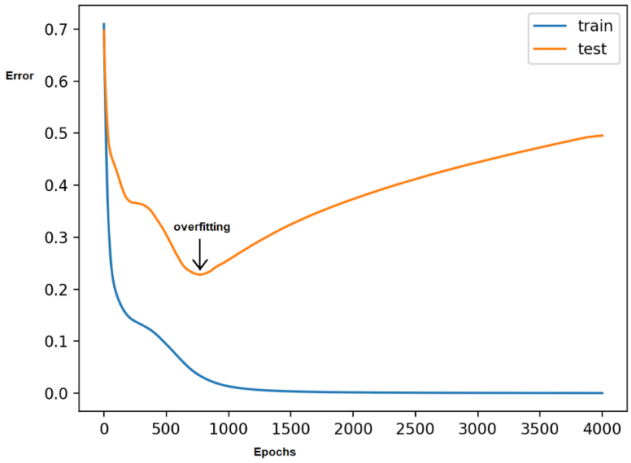


Figura 28: Gràfic variació de l'error en el temps per les dades d'entrenament I dades de prova. Font: (42)

4. DISSENY I IMPLEMENTACIÓ DEL SISTEMA INTEL·LIGENT

Aquest apartat té com a objectiu el disseny i implementació d'un sistema intel·ligent consistent en un algoritme d'Aprenentatge Automàtic que sigui capaç de modelar un determinat procés productiu a fi de classificar imatges de peces defectuoses o no defectuoses.

El context d'aquesta implementació es basa en un control de qualitat d'un procés de producció.

S'ha escollit utilitzar la tipologia de xarxa neuronal convolucional i a continuació se n'exposen els motius:

- La inspecció es realitzaria mitjançant visió artificial la qual s'encarregaria d'adquirir les imatges de la peça després del procés productiu. En vista de la *Taula 1* de l'apartat anterior, les xarxes neuronals convolucionals presenten majors avantatges en el processament d'imatges enfront d'altres tipologies.

La *Figura 29* mostra un esquema conceptual dels passos que s'han seguit per obtenir el model de classificació:

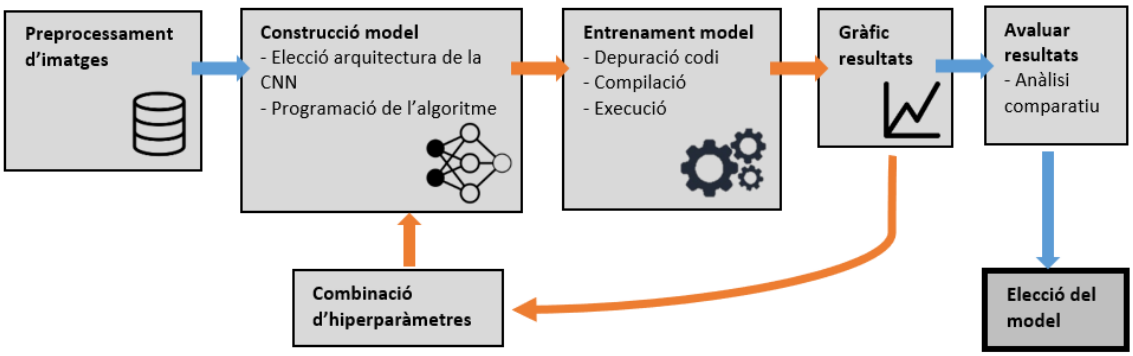


Figura 29: Diagrama conceptual per l'obtenció del model de classificació. Font: elaboració pròpia

En primer lloc, cal esmentar que s'han realitzat uns passos previs que no figuren en l'esquema anterior i s'aborden els següents subapartats *4.1. Tecnologies i eines utilitzades* i *4.2. Estudi del dataset*.

Partint del *dataset* d'imatges i havent establert l'objectiu d'interès, s'ha realitzat un primer pas de preprocessament de les imatges a fi de preparar-les per introduir-se en la xarxa neuronal. A continuació s'ha construït el model pel que fa al disseny de l'arquitectura de la xarxa neuronal convolucional i el desenvolupament de l'algoritme en línies de codi. Seguidament, s'entrena el model mitjançant l'execució a nivell de codi de programa i es representen gràficament les mètriques d'exactitud i error del model. Les fletxes de color taronja de l'esquema fan referència a un bucle que es repeteix per diferents combinacions d'hiperparàmetres a mode de marc experimental. A partir d'aquí, s'avaluen els resultats realitzant un anàlisi comparatiu dels diferents models obtinguts en funció de cada combinació d'hiperparàmetres per tal d'elegir el model òptim que ofereixi més bon rendiment.

4.1. Tecnologies i eines utilitzades

En aquest apartat es llisten els recursos utilitzats a nivell de llenguatge de programació, llibreries, entorns de programació i hardware.

4.1.1. Python

Entre els llenguatges de programació més utilitzats en el camp de la I.A., el llenguatge predominant és *Python*. Es tracta d'un llenguatge de programació d'alt nivell i de propòsit general àmpliament utilitzat. *Python* presenta una sintaxis que facilita la llegibilitat del codi i permet expressar conceptes en menys línies de codi comparat amb llenguatges com C. A més a més, el número de llibreries per *Machine Learning* suportades per *Python* és molt ampli i disposa d'una gran comunitat de desenvolupadors que donen suport i ofereixen disponibilitat a molts recursos online.

Cal esmentar que per programar aplicacions de *Machine Learning*, també destaquen altres llenguatges com R, el qual està més orientat a anàlisis estadístics, i *MatLab*. Aquest darrer presenta un entorn de computació numèrica amb facilitat per manipular matrius i disposa d'eines per dissenyar i entrenar xarxes gràficament mitjançant blocs.

S'han importat llibreries de Python per tal de facilitar el procés de desenvolupament del programa de les quals se'n destaca *Pandas* principalment per la reestructuració de les dades en matrius per la seva correcta representació en gràfics i *Matplotlib* per la creació de gràfics.

Per aquest treball s'ha escollit Python, versió 3.7, com a llenguatge de programació pels nombrosos avantatges que presenta i pels coneixements previs que he assolit durant el grau universitari.

Figura 30: Logotip Python. Font: (31)

4.1.2. Tensorflow i Keras

Referent a la programació de la xarxa neuronal, sempre hi ha la opció de programar pas a pas desde zero tenint ple coneixement de les operacions que s'hi realitzen i fent ús de les equacions del descens del gradient i backpropagation. En aquest treball s'ha escollit utilitzar eines per programar a més alt nivell i facilitar aquesta tasca: *TensorFlow i Keras*.

TensorFlow és una plataforma de codi obert per a *Machine Learning* creada per Google l'any 2015 i un dels referents en el desenvolupament de sistemes de *Machine Learning* i xarxes neuronals. Consta de múltiples eines, llibreries, recursos de la comunitat i tutorials que permeten als programadors implementar amb facilitat aplicacions amb tecnologia de *Machine Learning*. *Keras* es va integrar com a un mòdul de més alt nivell de *TensorFlow* que incorpora les seves funcionalitats.

L'augment de nivell aporta simplicitat a l'hora de programar però resta flexibilitat en el disseny de les arquitectures. Si el nivell 0 s'atribueix a la programació completa desde zero anteriorment esmentada, el nivell 1 correspondria a les llibreries de diferenciació automàtica de plataformes com *TensorFlow* que s'encarreguen de calcular automàticament les derivades parcials i es dissenya la xarxa neuronal com a una combinació de diferents operacions (suma ponderada, funció d'activació, backpropagation...). També es pot observar el disseny d'aquesta xarxa desde un punt de vista més abstracte, com a una combinació de capes, aquest correspondria al nivell 2 amb llibreries com *Keras*.

Figura 31: Logotip TensorFlow. Font: (32)

Figura 32: Logotip Keras. Font: (33)

4.1.3. Pycharm

D'entre la multitud d'entorns de programació s'ha escollit *PyCharm* per rebre'n recomacions. Es tracta d'un IDE de programació específicament pel llenguatge *Python* desenvolupat per l'empresa txeca *JetBrains*. *PyCharm* és un dels *IDE*'s més complerts i populars, ideal per qualsevol nivell d'experiència que inclou funcions de codificació, depuració, compilació i execució integrat en una API molt agradable. Incorpora un editor de codi amb ressaltat de sintaxis, funcions i suggeriments de codi així com també l'eina *SciView* amb la qual es poden mostrar gràfics i taules.

Figura 33: Logotip PyCharm. Font: (34)

4.1.4. Adobe Photoshop

S'ha treballat amb l'editor d'imatges *Adobe Photoshop* pel preprocessament i augment del *dataset* inicial. S'ha elegit aquest *software* pel fet d'oferir una interfície molt còmoda i senzilla d'utilitzar i per la multitud d'opcions d'edició per tal d'obtenir imatges el màxim de realistes.

Figura 34: Logotip Adobe Photoshop. Font: (35)

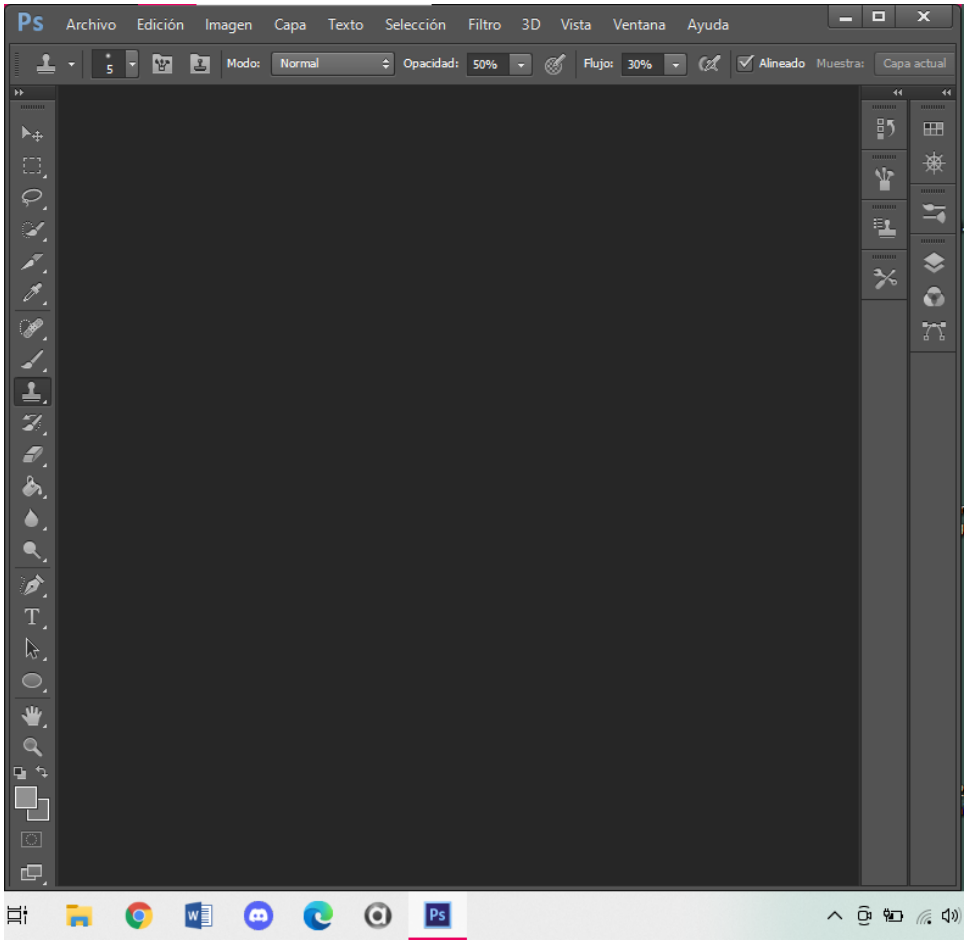


Figura 35: Interfície gràfica Photoshop. Font: elaboració pròpia

4.1.5. Hardware

Se sol sentir a dir que desenvolupar models de *Deep Learning* requereix de molta potència computacional i hardware per executar-se. De fet, entrenar models simples de *Deep Learning* en computadores sense GPU i poc potents pot ocupar hores, la qual cosa sol suposar un obstacle per als principiants.

La fase d'entrenament de la xarxa neuronal és la tasca més intensiva en recursos degut a la gran quantitat d'operacions matricials que s'hi realitzen. El desenvolupament de la I.A. i de l'aprenentatge profund s'ha vist en gran mesura facilitat per l'ús de les GPU les quals poden processar múltiples càlculs simultàniament enlloc d'un darrere l'altre. Una GPU és una unitat de processament de gràfics especialitzat amb memòria que realitza de manera convencional les operacions de punt flotant necessàries per la representació de gràfics de manera que allibera cicles de CPU per altres tasques. Les GPU presenten una gran quantitat de nuclis que permeten manejar càlculs múltiples i simples en paral·lel mentre que les CPU són millors per càlculs únics més complexes de forma seqüencial.

Cal esmentar que per evitar que la manca de hardware resulti una dificultat per entrenar models d'aprenentatge profund, hi ha altres opcions. *Google Collaboratory* és una plataforma online que ofereix un quadern de *Python* per programar i executar línies de codi amb llibreries preinstalades i executant-se al núvol en un hardware que *Google* posa a disposició pública de forma gratuïta. Els avantatges de treballar en el núvol són evidents però tot i així, moltes empreses aposten per "l'Edge Computing" consistent en la execució i programació en local propera a l'usuari o font de les dades la qual cosa genera fiabilitat, flexibilitat, velocitat i baixa latència enfront del "Cloud Computing". Per exemple, en el cas dels cotxes autònoms, el temps que pot tardar a rebre una resposta és clau, han de processar la informació captada de l'entorn a temps real. Un altre aspecte a considerar del "Edge Computing" és la seguretat, el fet de tenir dades en un entorn *cloud* les fa vulnerables i es poden veure compromeses.

L'entrenament i tota la programació de l'algoritme s'ha executat en local. Per aquest treball s'ha utilitzat un ordinador amb processador *Intel Core i7* i una targeta gràfica GPU *NVIDIA GeForce GTX*. En aquesta targeta s'han instal·lat controladors i eines CUDA (36) per l'arquitectura de càlcul paral·lel *NVIDIA* i la llibreria *cuDNN* (37) que permet la utilització de la GPU per *TensorFlow*. El hardware utilitzat ha sigut suficient i capaç de realitzar l'entrenament de la xarxa neuronal en minuts.

4.2. Estudi del dataset

El conjunt d'imatges ha estat proporcionat per l'empresa *Airtificial Intelligent Robots*. Correspon a un conjunt d'imatges directament captades a partir d'una càmera d'una determinada estació d'una línia de producció, sense cap processament. En aquesta estació s'hi realitzava un procés consistent en la dispensació de pasta tèrmica en diferents punts de la electrònica d'un motor per col·locar-hi la carcassa en la posterior estació. En acabar de dispensar la pasta es realitzava la fotografia de la peça per control de qualitat d'aquest procés.

Aquestes fotografies s'han realitzat garantint les següent condicions:

- Condicions de llum estables. Per aconseguir-ho es disposava d'un sistema d'il·luminació de color blau de manera que just abans de fer la fotografia s'apagaven els LED's de l'estació i s'encenia el sistema d'il·luminació. La llum blava fa ressaltar la pasta tèrmica i el resultat és una fotografia en escala de grisos on la pasta tèrmica queda de color clar i tota la electrònica de color més fosc.

- Fotografies realitzades amb la mateixa posició de la càmera i de la peça. Les peces passaven per les diferents estacions mitjançant un pallet.

A mode de nomenclatura, al llarg de la part pràctica es fa referència a imatges de peces defectuoses com a NOK i imatges de peces no defectuoses com a OK.

A continuació es mostra una imatge d'una peça amb dispensació no defectuosa i una defectuosa:

Figura 36: Peça OK. Font: Airtificial Intelligent Robots

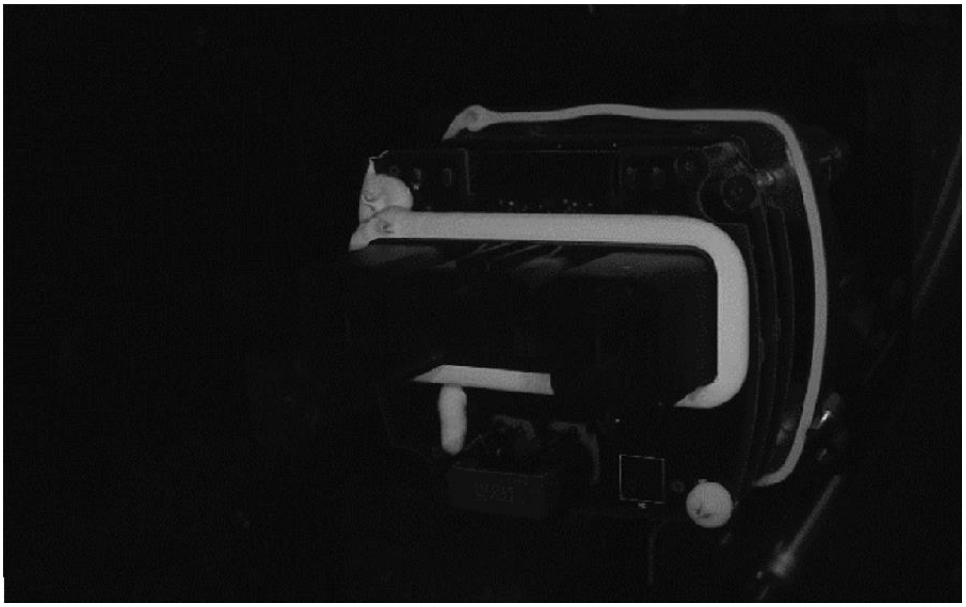


Figura 37: Peça NOK. Font: Airtificial Intelligent Robots

La *Figura 37* correspon a una imatge NOK provocada per un defecte d'excés de pasta en el punt superior esquerre de la peça.

Els possibles errors que es poden donar en aquest procés i poden provocar peces defectuoses són els següents:

- La pasta es pot assecar si es manté l'estació parada durant molta estona.

- Hi pot haver un excés o mancança de pasta en algun punt degut a que la seva fluïdesa podria venir afectada per la temperatura.

- Degoteig o rebava de la pasta al canviar de posició el capçal de dispensació.

4.2.1. Interès d'aplicar *Machine Learning*

Per realitzar un programa que sigui capaç de classificar les peces OK/NOK mitjançant *software* convencional, es definirien unes regions tal i com es mostra en la *Figura 38*. A partir d'aquí, els criteris que s'aplicarien per considerar una peça com a NOK podrien ser similars als següents:

- Gruix de la pasta: Presència de píxels blancs en les regions marcades en verd.

- Continuïtat de la pasta: la distància de cada píxel blanc a la línia vermella és superior a una distància *x* predefinida.

Al final, aquest programari convencional acaba sent un conjunt de regles i formats condicionals que van inspeccionant diferents regions per comprovar si compleixen algun d'aquests criteris per considerar la imatge com a NOK.

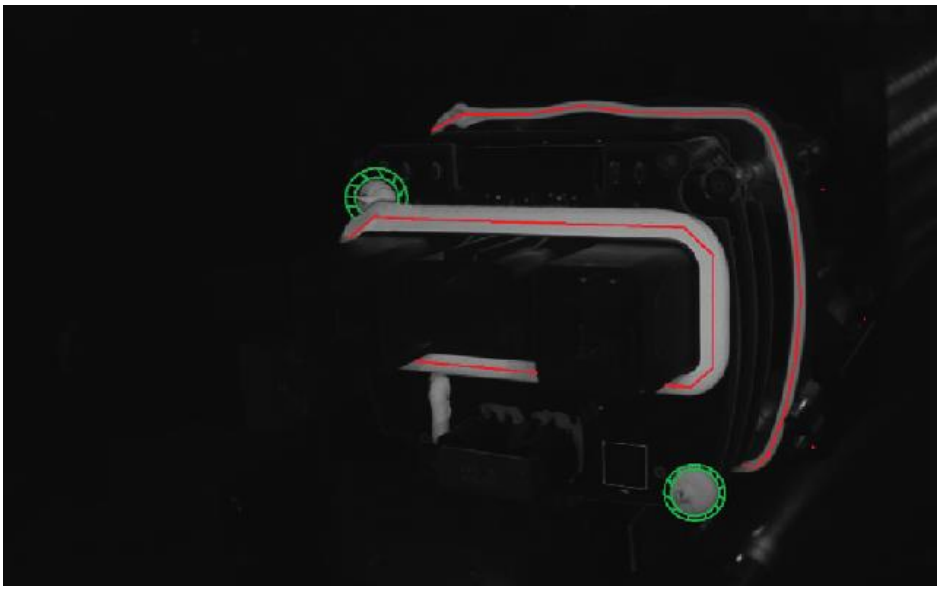


Figura 38: Exemple de criteris per considerar una peça com a NOK. Font: elaboració pròpia

No obstant, aquest programari podria presentar els següents desavantatges:

- Limitacions a nivell de defectes que no seria capaç de detectar. Per exemple, presència de pasta fora de les regions delimitades.

- Inversió de temps per part d'un enginyer de *software* per realitzar la tasca d'extracció de característiques.

- Ha d'englobar totes les opcions i ha de garantir que tots els possibles NOK quedin contemplats en les regles. Al cap i a la fi, l'objectiu és realitzar controls de qualitat robustos i fiables que no requereixin d'un operari que hagi de revisar cada resultat per assegurar que no es dona com a OK una peça que en realitat és defectuosa.

En aquest cas és interessant realitzar el control de qualitat mitjançant xarxes neuronals per la infinitat de casos defectuosos que hi podrien haver, és a dir, els defectes que poden ocórrer són de diferents tipologies però cap és el mateix, poden situar-se en diferents punts de la peça i poden presentar dimensions diferents. La xarxa neuronal té la capacitat de realitzar una interpolació entre tots els defectes, aprendre el que caracteritza una peça NOK i elaborar-ne un model.

4.3. Preprocessament de les imatges

El conjunt d'imatges inicial constava d'una proporció d'imatges de peces NOK molt petita respecte les imatges OK la qual cosa va fer necessari trobar un mètode per tenir la mateixa quantitat de peces OK que de NOK.

Tal i com s'ha esmentat en l'apartat anterior *2.3. Anàlisi del mercat*, un dels principals problemes que apareixen en el modelatge basat en dades, són la falta de dades fidedignes que representin la realitat a modelar en la seva totalitat. En qualsevol entorn és habitual tenir un conjunt de dades disponibles per treballar molt petit perquè la recopilació pot ser costosa o directament impossible. No obstant, els models entrenats amb un petit número d'exemples tendeixen a sobreajustar-se.

Per abordar aquest problema de l'escassetat de dades, majoritàriament s'utilitza la següent tècnica:

- *Data Augmentation* és una estratègia que permet augmentar significativament el conjunt de dades mitjançant transformacions de les imatges existents com rotació, modificació del color, injecció de soroll, zoom...

Per aquest *dataset* no tindria sentit aplicar *Data Augmentation* perquè s'ha suposat a l'inici que la peça sempre arriba amb la mateixa posició a la estació i les condicions de llum són sempre estables.

Així doncs, per tal d'equilibrar el número d'imatges es va optar per generar imatges NOK provocant defectes en algunes imatges OK mitjançant l'editor d'imatges *Adobe Photoshop*. Per fer-ho es van tenir en compte una sèrie de consideracions:

- El *dataset* ha de ser representatiu dels tipus de defectes més comuns que es poden donar en aquest procés en concret.

- En la línia dels possibles defectes reals i en base a l'abast i dimensions d'aquests, crear defectes de diferents mides i en altres punts de la peça.

- Aconseguir imatges NOK el màxim de realistes ja que *Photoshop* ofereix eines per copiar textures i colors.

- Les imatges OK que s'utilitzin per entrenar la xarxa no es poden utilitzar a posteriori per convertir-les en NOK ja que es pot alterar el procés d'aprenentatge.

- Intentar aconseguir el màxim d'aleatorietat provocant diferents tipus de defectes en la mateixa proporció i que no mostrin cap patró de posicions de defectes. Per exemple, no crear diferents defectes en els mateixos punts de la peça per cada imatge.

Altrament, també s'ha realitzat un preprocessament de totes les imatges abans d'entrar-les a la xarxa neuronal consistent en:

- Reducció de la mida de les imatges amb la finalitat de reduir la complexitat de la xarxa neuronal, el número de paràmetres a entrenar i el temps d'entrenament requerit.
  Totes les imatges s'han reduït de 1920x1200 píxels a 300x300 ja que s'ha considerat una mida suficient pel que s'ha de detectar. La tasca s'ha realitzat mitjançant l'editor *PhotoShop*.

- Escalat de la imatge per normalitzar-la obtenint píxels en un rang de 0 a 1. La tasca s'ha realitzat mitjançant un mòdul de la llibreria *TensorFlow* anomenat *ImageDataGenerator*.

A continuació es mostren dues imatges de peces NOK resultants de l'edició en *Photoshop* i la reducció de mida:

Figura 39: Peça NOK 300x300 amb defectes provocats. Font: elaboració pròpia

Figura 40: Peça NOK 300x300 píxels amb defectes provocats. Font: elaboració pròpia

Finalment, el número d'imatges obtingudes és de 200 imatges OK i 200 imatges NOK. Aquest total s'ha dividit en dos conjunts d'una forma aleatòria:

- Dades d'entrenament (*train_set*): conjunt d'imatges que s'utilitzaran per entrenar el model.

- Dades de validació (*val_set*): conjunt d'imatges per obtenir el rendiment del model. El model no haurà vist mai aquestes imatges i això permetrà detectar possibles casos d'*overfitting*.

Normalment, les dades d'entrenament representen el 80% del total de mostres disponibles i les dades de validació representen el 20%. D'aquesta manera, el conjunt *train_set* consta de 160 imatges per cada classe i el conjunt *val_set* consta de 40 imatges per cada classe.

A continuació, en la *Figura 41* es mostra l'estructura en carpetes de l'emmagatzematge dels dos conjunts de dades en local:

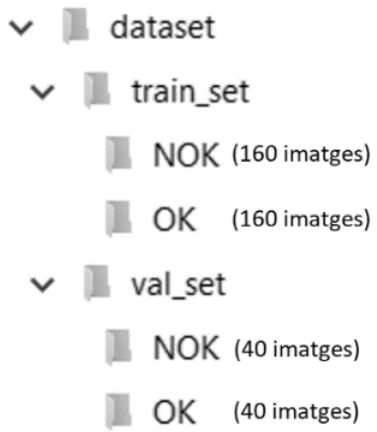


Figura 41: Captura de pantalla de l'estructura de carpetes en local. Font: elaboració pròpia

4.4. Disseny i construcció del model

Aquest apartat consta d'una primera part on es defineix l'arquitectura de la xarxa neuronal convolucional, a continuació s'assignen els valors dels hiperparàmetres i finalment es mostra la implementació de l'algoritme a nivell de programari i línies de codi.

4.4.1. Arquitectura de la CNN

La xarxa neuronal convolucional consta de la següent estructura:

Layer (type)	*Output Shape*	*Param #*
conv2d (Conv2D)	*(None, 300, 300, 8)*	*80*
max_pooling2d (MaxPooling2D)	*(None, 150, 150, 8)*	*0*
conv2d_1 (Conv2D)	*(None, 150, 150, 16)*	*1168*
max_pooling2d_1 (MaxPooling2D)	*(None, 75, 75, 16)*	*0*
flatten (Flatten)	*(None, 90000)*	*0*
dense (Dense)	*(None, 224)*	*20160224*
dense_1 (Dense)	*(None, 1)*	*225*

Total params: 20,161,697
Trainable params: 20,161,697
Non-trainable params: 0

Taula 2: Estructura de la xarxa neuronal convolucional. Font: elaboració pròpia

L'entrada a la xarxa neuronal és la imatge de dimensions 300x300x1 que presenta una altura de 300 píxels, amplada de 300 píxels i 1 canal ja que són imatges en escala de grisos. La primera capa convolucional consta de 8 filtres *kernel* de (3x3) amb la qual cosa es generen 8 mapes de característiques de (300x300) o, el que és el mateix, un volum de (300x300x8) d'informació. A continuació s'aplica una reducció mitjançant *max-pooling* de (2x2) generant els mateixos 8 mapes de característiques però de 150x150 píxels cada un. La segona capa convolucional consta de 16 filtres *kernel* de (3x3) seguit d'una reducció *max-pooling* (2x2) que genera 16 mapes de característiques de 75x75 píxels cada un. En totes les capes convolucionals s'ha aplicat la tècnica *same padding* per l'omplenat de la imatge de manera que l'entrada a la capa convolucional tingui les mateixes dimensions que la capa anterior per mantenir un major control de la xarxa. La sortida d'aquesta darrera capa s'aplana per convertir-se en un vector pla de 90000 valors que conté tota la informació. Aquest vector ja es pot entrar en una xarxa neuronal *fully-connected* de 224 neurones amb funció d'activació *ReLu*. La sortida és una sola neurona amb funció d'activació sigmoide que dona la probabilitat que la imatge pertanyi a una classe.

La *Figura 42* mostra l'arquitectura de la CNN a mode d'esquema visual.

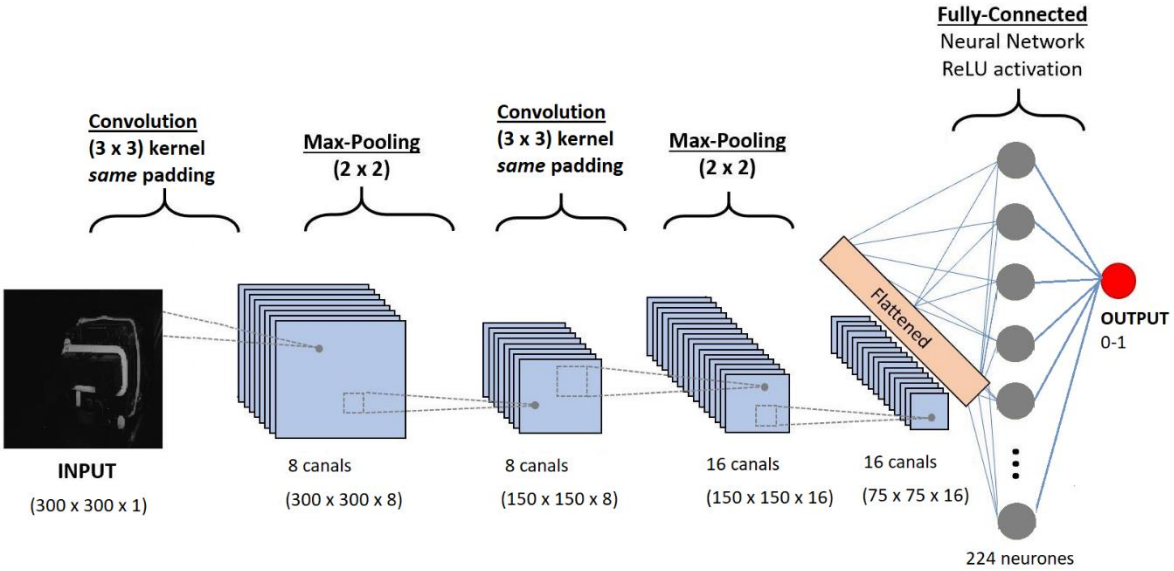


Figura 42: Arquitectura de la xarxa neuronal convolucional en qüestió. Font: elaboració pròpia

L'elecció d'aquesta arquitectura s'ha realitzat a partir d'estudiar les arquitectures més comunes i comparar-les amb les prestacions del *dataset* utilitzat en aquest treball:

- Considerant que les imatges són de dimensions molt reduïdes i en escala de grisos es podria afirmar que són relativament fàcils de processar i no requereixen de tantes capes convolucionals. A més a més, sempre és millor aconseguir arquitectures simples abans que arquitectures molt complexes que poden contenir elements redundants que augmentin el soroll en els resultats.

- Pel que fa a la variabilitat, el *dataset* utilitzat recull imatges de només dues classes diferents i presenta una variabilitat inferior comparat amb un *dataset* d'imatges per classificació binària de gats i gossos en el qual hi poden haver moltes races de gossos i gats, molts colors de pèl, l'animal pot adoptar moltes postures, la fotografia es pot prendre a l'aire lliure o a l'interior amb menys claror... Per tant, l'espectre o domini de característiques és molt inferior en el *dataset* utilitzat ja que es mantenen les mateixes condicions de llum, de posició de la peça i de color de manera que la variabilitat resideix únicament en:

    - Les diferents tipologies de defectes de la pasta així com també els marges pels quals es considera la peça com a OK.

    - Els components electrònics que s'observen de color més fosc i que també poden variar en les diferents imatges OK. Aquest últim aspecte no ha d'influir en la classificació i la xarxa neuronal ho ha d'aprendre.

Aquest fet de la variabilitat també es veu reflectit en el número de imatges que es necessiten per entrenar, com que la xarxa neuronal no ha d'aprendre tantes característiques, menys imatges seran suficients per realitzar un model amb prou confiança.

4.4.2. Optimització d'hiperparàmetres

El valor òptim d'alguns hiperpàrametres no es pot conèixer a priori per cada model de manera que és necessària la tasca crítica anomenada optimització d'hiperparàmetres en la cerca del millor rendiment final de la xarxa.

Sovint, els mètodes per l'optimització d'hiperparàmetres són manuals i es basen en l'experiència i comparació amb altres models optimitzats, mitjançant prova i error o per força bruta. No obstant, aquestes metodologies no garanteixen els millors valors d'hiperparàmetres pel model, per aquest motiu es necessitaria un mètode sistemàtic.

S'ha escollit emprar una metodologia basada en la cerca en quadrícula o *Grid Search* en la qual s'especifica un subconjunt d'hiperparàmetres i s'avalua el resultat per cada un d'ells de manera que finalment s'escull el millor resultat de tots els obtinguts.

S'han establert uns hiperparàmetres variables i fixes pel marc experimental. S'entén com a hiperparàmetres fixes aquells que pel tipus de *dataset* ja es poden determinar a priori. Per contra, els hiperparàmetres variables són aquells que inicialment no es coneix el seu valor òptim i per tant, són l'objectiu de l'ús de la tècnica de cerca en quadrícula.

Assignació del valor dels hiperparàmetres fixes

- Èpoques (*epochs*): s'ha establert un temps d'entrenament de 20 èpoques ja que s'ha considerat que és suficient per permetre al model arribar a la convergència i és prou petit per tal d'evitar el sobreentrenament i l'*overfitting*. No obstant, en alguns casos es pot augmentar si la combinació d'hiperparàmetres ho requereix per arribar a la convergència. De la mateixa manera, també s'ha programat una aturada de l'entrenament després de 5 èpoques amb un valor de rendiment inferior a l'anterior mitjançant el mòdul *EarlyStopping* de la llibreria de *Keras*. Gràcies a això, si alguna de les mètriques d'interès deixa de millorar en algun cas, s'atura l'entrenament a fi de no invertir més temps i recursos.

- Funció de pèrdua (*Loss_function*): tractant-se d'un model de classificació binària s'utilitzarà la funció de pèrdua *binary_crossentropy* amb la qual cosa la capa de sortida serà una sola neurona amb funció d'activació sigmoide.

- Optimitzador (*Optimizer*): s'ha escollit adam a priori per resultar efectiu per la majoria de models. S'ha utilitzat l'objecte *Adam* de la llibreria *Keras* del qual només s'ha parametritzat el ràtio d'aprenentatge i els altres paràmetres s'han deixat per defecte.

Assignació del rang (o espai de cerca) dels hiperparàmetres variables

- Ràtio d'aprenentatge o *learning rate* (α, *lr*): donat que a priori no es coneix com serà la funció de cost per aquest model en concret, el valor d'aquest hiperparàmetre es variarà en el següent rang: [0,00001; 0,0001; 0,001; 0,004; 0,006; 0,008; 0,01].

- Mida del lot (*batch_size*): considerant que la mida del *dataset* és relativament petita, s'han descartat valors de *batch_size* superiors a 32 que restarien flexibilitat. S'ha optat per un descens de gradient de *mini-batch* ja que és el més habitual i s'ha escollit variar el valor d'aquest hiperparàmetre en el següent rang: [4; 8; 16; 32].
  Els valors s'han escollit en potència de dos per no generar alteracions en l'entrenament per incompatibilitats amb la GPU.

Així doncs, el marc experimental es basa en executar l'algoritme per cada una de les combinacions possibles dels hiperparàmetres variables d'acord amb la *Taula 3*.

		lr						
		0.00001	0.0001	0.001	0.004	0.006	0.008	0.01
batch_size	4	x	x	x	x	x	x	x
	8	x	x	x	x	x	x	x
	16	x	x	x	x	x	x	x
	32	x	x	x	x	x	x	x

Taula 3: Espai de cerca en quadrícula per cada combinació d'hiperparàmetres. Font: elaboració pròpia

4.4.3. Programació de la CNN

Pel que fa a la programació de la xarxa neuronal convolucional s'ha creat un fitxer de programa (*main.py*) on s'han importat diferents mòduls de la llibreria *Keras* de *TensorFlow*. En aquest apartat només es fa referència a algunes línies de codi d'especial interès, el codi de programa complert es troba en l'*Annex A* amb alguns comentaris per fer-ho més entenedor.

La *Figura 43* es tracta d'una captura de pantalla de l'IDE de *PyCharm* on s'ha treballat.

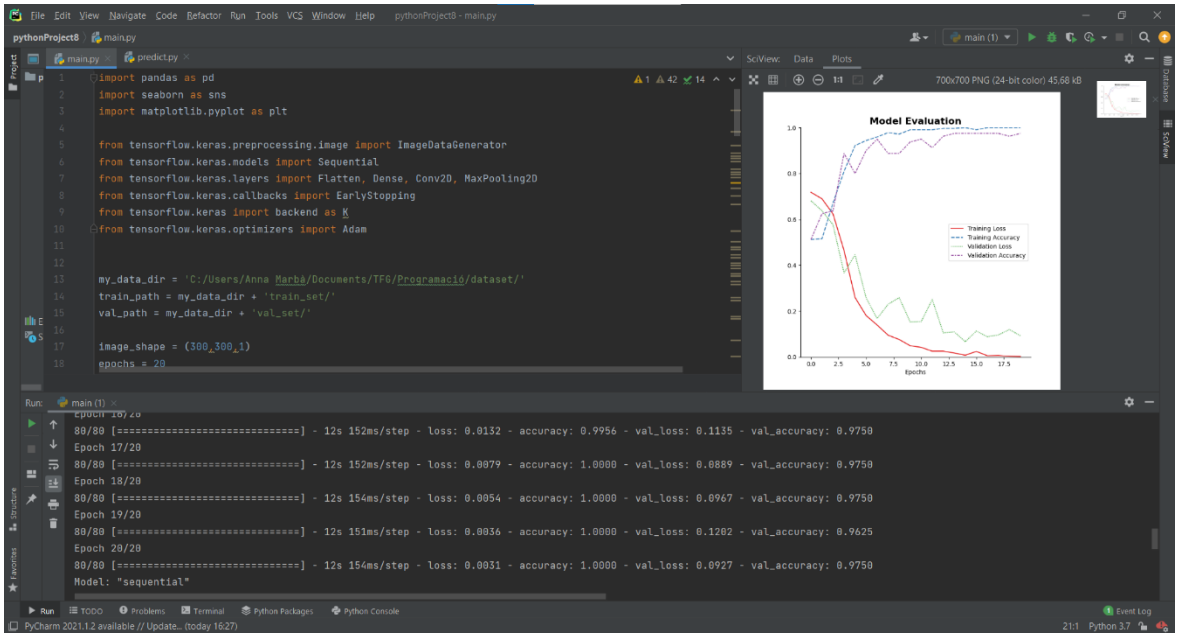


Figura 43: Captura de pantalla de l'IDE de PyCharm. Font: elaboració pròpia

El programa consta de 5 parts diferenciades:

I. Tibar el *dataset* del directori local i preparar les imatges:

Tal i com s'ha esmentat abans, el mòdul *ImageDataGenerator* s'encarrega de preparar les imatges per entrar-les a la xarxa neuronal.
Mitjançant la funció *flow_from_directory()* de la llibreria i prèviament havent definit un mode de classificació binari, s'assigna de forma automàtica una etiqueta classificant cada imatge en funció de l'estructura de carpetes on estan desades. Addicionalment també es realitza la separació de imatges per lots de forma automàtica especificant el *batch_size*.

```
image_gen = ImageDataGenerator(rescale=1/255)
train_set = image_gen.flow_from_directory(train_path,
                                          target_size=image_shape[:2],
                                          color_mode="grayscale",
                                          batch_size=batch_size,
                                          class_mode='binary',shuffle=True)
val_set = image_gen.flow_from_directory(val_path,
                                          target_size=image_shape[:2],
                                          color_mode="grayscale",
                                          batch_size=batch_size,
                                          class_mode='binary',shuffle=False)
```

La *Figura 44* mostra que efectivament l'algoritme ha trobat les 320 imatges corresponents a l'entrenament i 80 a la validació i les ha classificat en dues classes.

Figura 44: Captura de pantalla de l'IDE PyCharm. Font: elaboració pròpia

II. Creació del model:

El mòdul *models* s'ha usat per la funció *Sequential()* a fi de crear un objecte anomenat "model" corresponent a la CNN formada per una seqüència de capes. A partir d'aquí s'ha afegit una línia de codi amb la instrucció *add* per cada capa tal i com es mostra en les següents línies de codi:

```
model = Sequential()

model.add(Conv2D(filters=8, kernel_size=(3,3),padding='same',
input_shape=image_shape, activation='ReLu',))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(filters=16, kernel_size=(3,3),padding='same',
input_shape=image_shape, activation='ReLu',))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())

model.add(Dense(224, activation='ReLu'))

model.add(Dense(1, activation='sigmoid'))
```

III. Compilar el model:

Per l'entrenament de la CNN primer s'ha de compilar l'arquitectura usant la funció *compile()* on s'hi especifica la funció de pèrdua, l'optimitzador i les mètriques.

Per al seguiment i avaluació del rendiment de la xarxa neuronal s'utilitza un mòdul de mètriques *accuracy* i l'objecte *History* de la llibreria *Keras* en el qual s'emmagatzemen els seus valors per tal de representar-les gràficament al finalitzar cada època i per cada conjunt de dades (entrenament i validació).

Les mètriques són les següents:

- Pèrdua ("*loss*"): és el valor de la funció de pèrdua prèviament mencionada en l'apartat *3.2.3. Hiperparàmetres d'entrenament*.
- Precisió ("*accuracy*"): calcula la freqüència amb la qual les prediccions són iguals a les etiquetes. S'estableix un llindar de manera que les prediccions superiors a 0,5 es converteixen en 1 i, en cas contrari, en 0. Llavors, la precisió és el resultat del número de casos en els quals la predicció equival al valor de la etiqueta real dividit del número total de mostres.

D'aquesta manera, un augment de la precisió i un decreixement de la pèrdua a cada època es traduirà en un bon rendiment del model.

```
model.compile(loss='binary_crossentropy',
              optimizer=Adam(lr=lr),
              metrics=['accuracy'])
```

IV. Entrenar el model:

La funció *fit()* entrenarà el model. S'hi especifiquen les dades d'entrada, número d'èpoques i conjunt de dades de validació entre d'altres.

```
model.fit_generator(train_set,epochs=epochs,
                    validation_data=val_set,
                    callbacks=[early_stop])
```

V. Desar el model:

Finalment, el model i els respectius paràmetres de la xarxa neuronal fruit de l'entrenament es desen en un directori amb un nom de fitxer *model.h* i *weights.h* mitjançant la funció *save()*.

```
model.save('C:/Users/Anna Marbà/Documents/TFG/Programació/model/model.h5')
model.save_weights('C:/Users/Anna
Marbà/Documents/TFG/Programació/model/weights.h5')
```

A fi de realitzar prediccions d'imatges noves s'ha creat un nou fitxer *Python* anomenat *predict.py*. La *Figura 45* mostra visualment els dos fitxers *Python* utilitzats i com es relacionen. El programa complert figura en l'*Annex B*.

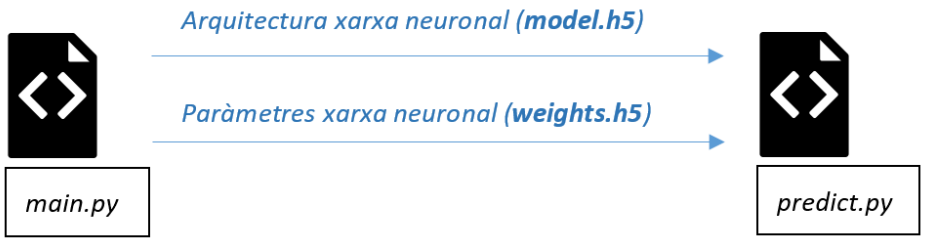


Figura 45: Relació entre els dos fitxers de programa main.py i predict.py. Font: elaboració pròpia

A grans trets, el programa *predict.py* consta de 3 parts:

I. Reconstruir la xarxa neuronal amb els paràmetres que s'han après

Per fer-ho, es carreguen els fitxers anteriors (*model.h5* i *weights.h5*) amb la funció *load*.

```
model = load_model(model.h5)

model.load_weights(weights.h5)
```

II. Carregar la imatge a predir

```
img=load_img('C:/Users/Anna Marbà/Documents/TFG/Programació/prova/301.bmp')
```

III. Predir la classe de la imatge d'entrada

La funció *predict()* permet mostrar per pantalla el valor de la classificació del model per una imatge en concret. Degut a que la neurona de sortida és sigmoide, la sortida de la xarxa neuronal és un valor numèric entre 0 i 1 que s'interpreta com la probabilitat que la imatge d'entrada sigui OK. A partir d'aquest valor, s'ha programat una classificació binària OK/NOK manual mitjançant un llindar tal i com es mostra a continuació:

```
prediction = model.predict(img)
print(prediction)  #probabilitat que la imatge pertanyi a la classe OK. Valor de sortida de la xarxa neuronal
if prediction < 0.5:
    print("classification: NOK")
else:
    print("classification: OK")
```

4.5. Entrenament del model

Executant el codi de programa *main.py* comença el procés d'entrenament. Durant aquest procés les mètriques es mostren per pantalla en l'IDE *PyCharm* després de cada època tal i com s'observa en la *Figura 46*.

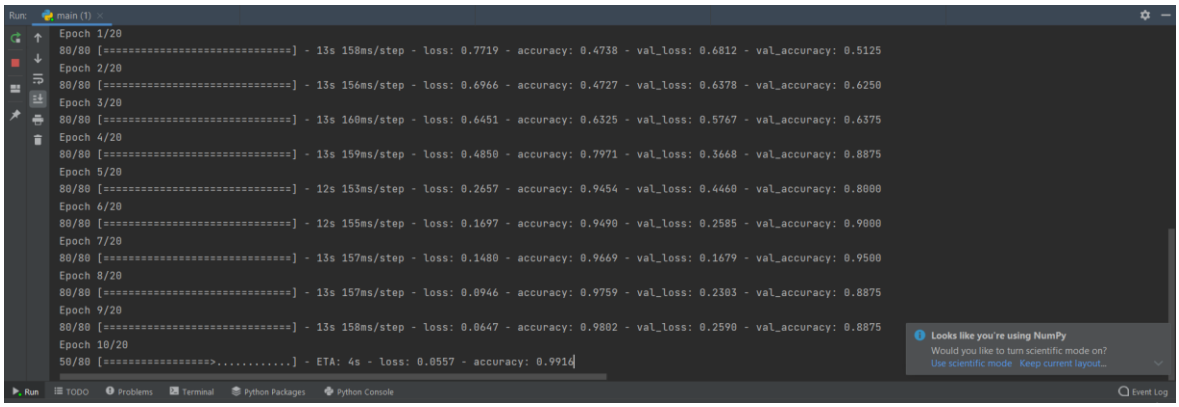


Figura 46: Procés d'entrenament en curs. Font: elaboració pròpia

5. AVALUACIÓ I ANÀLISI DE RESULTATS

Per tal de comparar el rendiment de la xarxa neuronal per diferents hiperparàmetres, en aquest apartat s'analitzen els resultats obtinguts del marc experimental. Es defineix rendiment en termes de:

- Convergència: el model assoleix la convergència quan arriba a un estat durant l'entrenament en el qual el valor de pèrdua ja no millora i es manté en un valor estable.
- Precisió: indica com de precisa és la resposta del model comparat amb la resposta correcta.
- Error de l'entrenament (pèrdua): un baix error indica que el model és capaç d'aprendre.
- Robustesa: el model no caigui en *overfitting*.
- Rapidesa computacional.

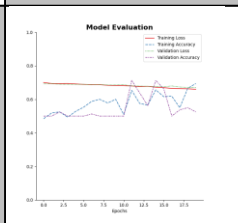
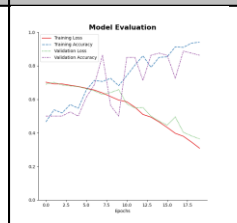
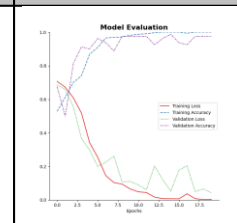
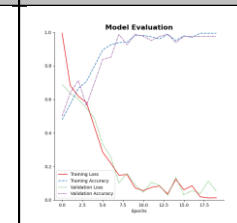
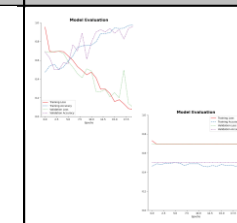
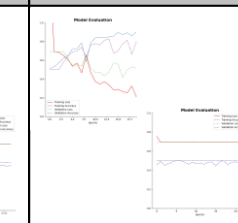
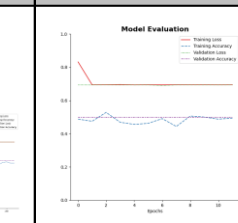
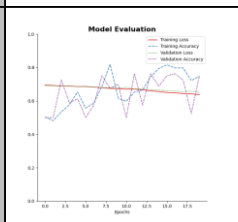
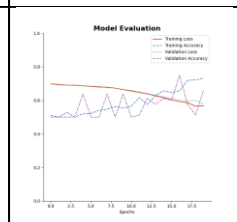
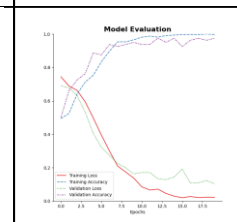
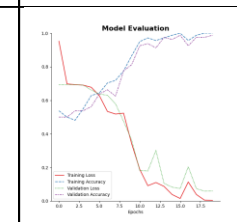
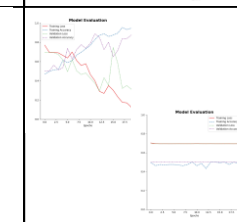
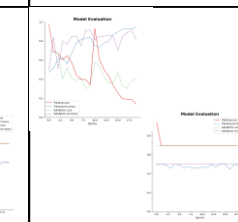
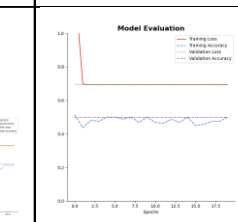
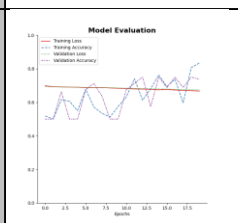
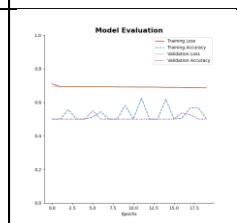
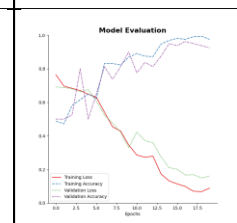
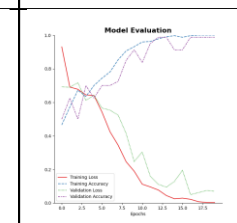
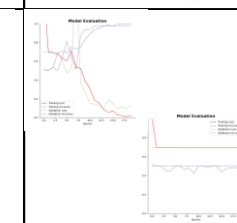
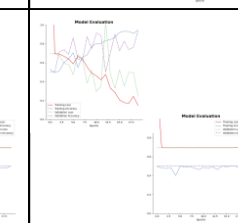
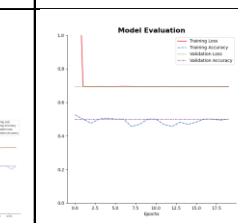
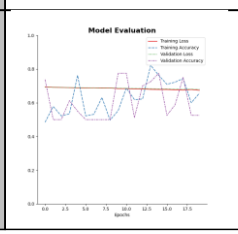
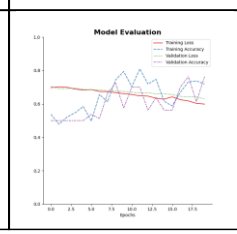
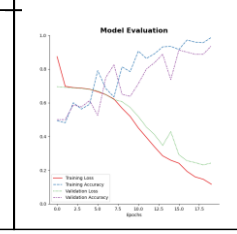
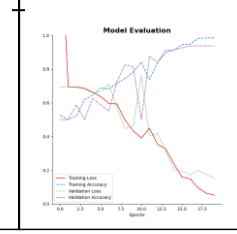
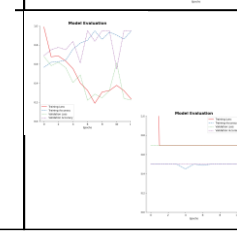
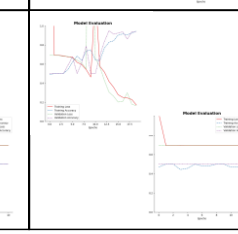
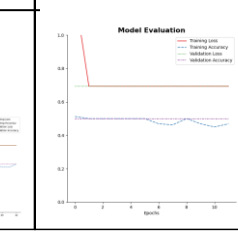
La *Taula 4* de la pàgina següent conté un gràfic amb el resultat de les mètriques per cada combinació d'hiperparàmetres (per cada configuració) després de 20 èpoques d'entrenament.

Es pot entendre que cada gràfic correspon a un model diferent, entenent-se model com el resultat final del procés d'aprenentatge fruit d'una determinada configuració de la CNN.

Consideracions prèvies:

En l'obtenció d'aquesta taula cal remarcar que s'han realitzat vàries execucions de l'algoritme per cada configuració a fi d'assegurar estabilitat entre execucions. Tal i com s'ha esmentat en el marc teòric, la inicialització de pesos de la xarxa neuronal és aleatòria i els resultats obtinguts en diferents execucions poden variar.

En concret, pels valors de *lr*=0,006 i *lr*=0,008, el resultat obtingut en cada execució variava molt de l'obtingut en l'execució anterior. Per aquest motiu, un sol gràfic no es considerava representatiu per una configuració i s'han afegit dos gràfics corresponents als límits dels resultats obtinguts.

batch_size	lr: 0.00001	0.0001	0.001	0.004	0.006	0.008	0.01
4							
8							
16							
32							

Taula 4: Resultats obtinguts per cada combinació d'hiperparàmetres. Font: elaboració pròpia

En la taula anterior s'observa el següent:

Per diferents valors de *lr*:

- ***lr*=0,00001 i *lr*=0,0001:**
  La tendència del gràfic mostra aprenentatge per l'augment en el valor de les mètriques de precisió i la caiguda de l'error però no augmenten a un nivell tant exponencial com en altres casos.
  No arriba a convergir en 20 èpoques d'entrenament.

  - Baixa velocitat d'aprenentatge.
  - Aquests valors podrien resultar massa petits i fer ineficient l'algoritme degut a un entrenament llarg per arribar a la convergència.

  Per explicar-ho millor mitjançant un exemple gràfic, la *Figura 47* mostra una funció de cost *f(x,y)* en la qual s'utilitza un *lr* massa petit. Encara que convergeix en un mínim, el número d'iteracions per aconseguir-ho és molt alt ja que els passos en cada iteració són molt petits.

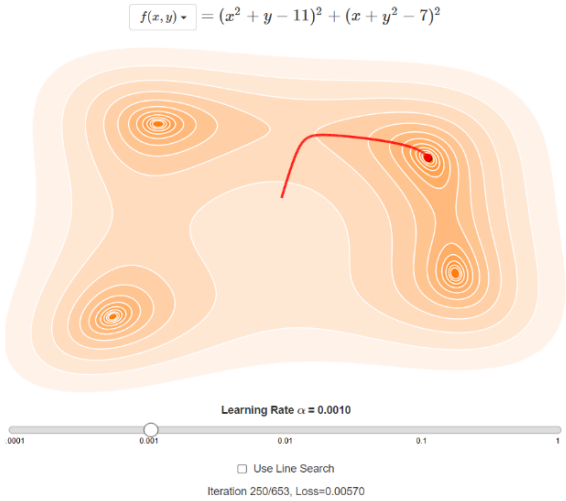


Figura 47: Efecte d'un valor massa petit de lr per convergir en un mínim de la funció de cost. Font: (38)

- ***lr*=0,001 i *lr*=0,004:**
  S'obtenen els millors resultats.
  L'algoritme aconsegueix una precisió de les dades d'entrenament superior al 90% en tots els casos encara que suaument inferior en les dades de validació.

  - Això podria indicar un possible cas d'*overfitting*.

- ***lr*=0,006 i *lr*=0,008:**
  Gran inestabilitat de resultats en cada execució.
  En el millor dels casos, s'obtenen resultats bons amb una precisió superior al 90%.
  En el pitjor dels casos no convergeix.

  - ➢ No s'haurien de considerar aquests valors de *lr* com a òptims degut a la poca garantia de resultats en l'execució de l'algoritme.

- ***lr*=0,01:**
  No convergeix.
  No mostra cap procés d'aprenentatge ja que la mètrica de pèrdua es manté constant a un valor alt i superior a la precisió de manera que l'entrenament de l'algoritme per més èpoques no obtindria millors resultats.

  - ➢ Això pot indicar que l'algoritme s'hagi quedat en un bucle infinit al ser incapaç d'entrar en el mínim local de la funció de cost ja que la precisió oscil·la mantenint-se a un valor baix.

    La *Figura 48* mostra un exemple gràfic en el qual s'utilitza un valor de *lr* massa gran que fa que els passos pel descens del gradient siguin massa llargs i sigui incapaç de convergir en un mínim. L'entrenament queda en bucle.

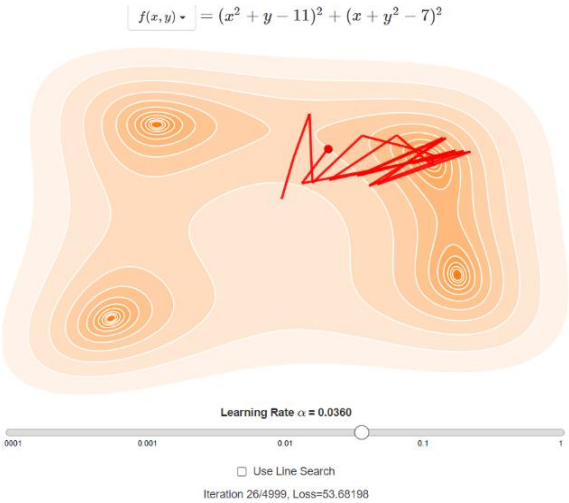


Figura 48: Efecte d'un valor massa gran de lr per convergir en un mínim de la funció de cost. Font: (38)

Per diferents valors de *batch_size*:

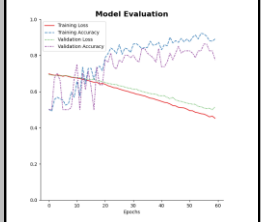
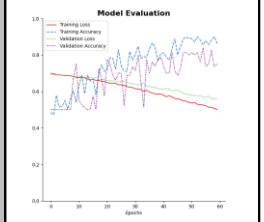
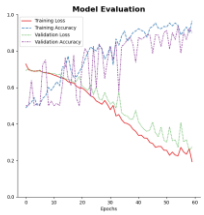
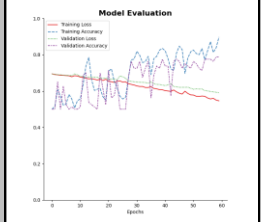
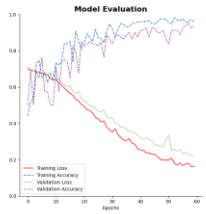
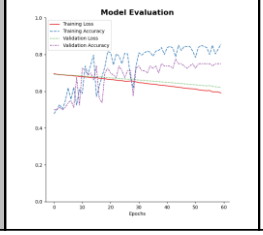
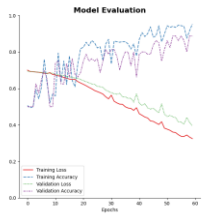
En general només s'observa que un valor més elevat en la mida del lot provoca major temps d'entrenament per arribar al mateix resultat que l'obtingut amb una mida de lot més petita.

- Així doncs, la principal diferència és el temps per assolir la convergència:
  A major mida del lot, menor velocitat d'aprenentatge la qual cosa és lògica ja que es realitzen menys actualitzacions de pesos.

5.1. Correlació entre hiperparàmetres i rendiment final

En aquest apartat es realitza un anàlisi a fi d'establir una possible correlació entre els dos hiperparàmetres *lr* i *batch_size* i l'impacte en el rendiment final del model pel *dataset* en qüestió.

Per tal de comparar els diferents models, s'ha considerat comparar-los en la convergència per totes les configuracions. Per aquest motiu, es va repetir l'entrenament per les configuracions amb *lr* igual a 0,00001 i 0,0001 en 60 èpoques i el resultat es mostra en la *Taula 5*.

batch_size	lr = 0.00001	lr = 0.0001
4		
8		
16		
32		

Taula 5: Resultats obtinguts per més èpoques d'entrenament. Font: elaboració pròpia

En la *Taula 5* s'ha obtingut més soroll en les mètriques probablement per l'augment dels passos i actualitzacions.

A partir dels valors de les mètriques de les dades d'entrenament obtinguts per cada configuració en la convergència de l'algoritme, s'han elaborat dos gràfics de correlació que s'exposen a continuació:

- **Estudi de l'impacte dels hiperparàmetres en la pèrdua:**

El gràfic de la *Figura 49* representa els valors de pèrdua (en tant per u) en funció de *lr*. En el següent nivell de detall es mostra l'impacte del *batch_size* per un mateix valor de *lr*.

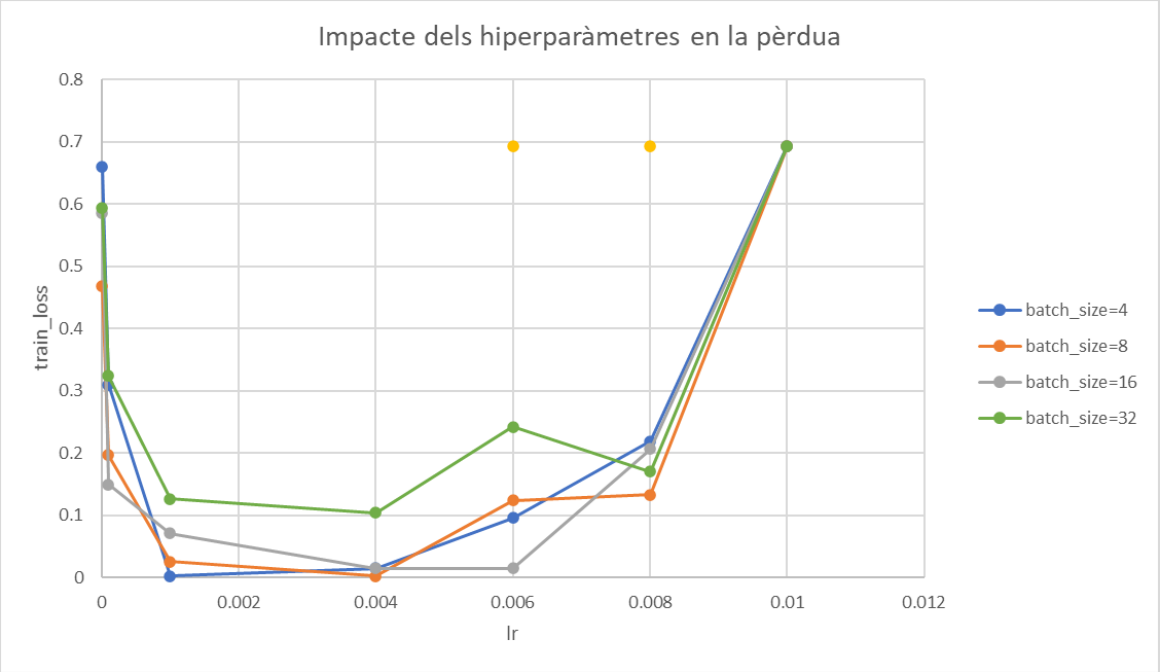


Figura 49: Gràfic Impacte dels hiperparàmetres en la pèrdua. Font: elaboració pròpia

Del gràfic se'n dedueix el següent:

- ***lr*=0,00001 i *lr*=0,0001:**
  S'obtenen valors de pèrdua relativament elevats degut probablement perquè uns passos tant petits del descens del gradient poden provocar que l'algoritme es quedi atrapat en un mínim local enlloc del mínim global de la funció de cost.

- ***lr*=0,001 i *lr*=0,004:**
  S'obtenen els valors més baixos d'error.

- ***lr*=0,006 i *lr*=0,008:**
  S'han representat només els millors resultats obtinguts de la totalitat d'execucions realitzades però aquests valors no són representatius ja que en alguns casos s'obtenien valors superiors al 60% d'error (punts de color groc al gràfic) degut a la inestabilitat que presenta aquesta configuració.

- ***lr*=0,01:**
  S'obtenen els valors més elevats de pèrdua ja que l'algoritme no convergeix.

Pel que fa a l'impacte del *batch_size*, no es pot establir una clara correlació ja que la diferència podria residir en els graus de llibertat entre execucions de l'algoritme. Per valors de *lr* de 0,001 i 0,004 es podria detectar una possible correlació basada en una major pèrdua amb l'augment del *batch_size* però només seria aplicable a aquestes dues configuracions ja que per altres valors de *lr* això ja no es compleix.

- **Estudi de l'impacte dels hiperparàmetres en la precisió:**

El gràfic de la *Figura 50* ve a mostrar el mateix que l'anterior però comptabilitzat en precisió (tant per u).

Efectivament, l'augment de la precisió significa un decrement en la pèrdua però cal remarcar la diferència que resideix entre aquests dos termes. A grans trets, la precisió interessa més aviat des de una perspectiva empresarial ja que es traduïble a l'objectiu comercial mentre que la pèrdua representa l'objectiu de l'algoritme per minimitzar l'error de predicció des de una perspectiva matemàtica.

En vista de la inestabilitat per valors de *lr* superiors a 0,004, en aquest cas ja no s'han representat.

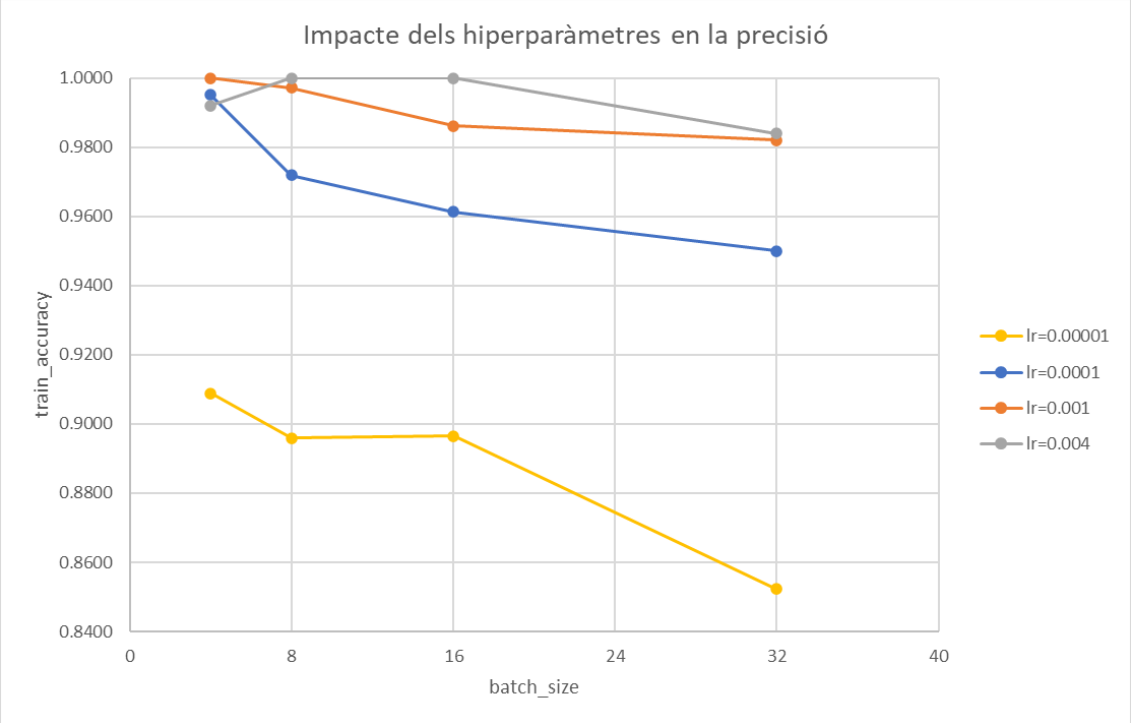


Figura 50: Gràfic Impacte dels hiperparàmetres en la precisió. Font: elaboració pròpia

Aquest gràfic corrobora les afirmacions anteriors ja que, en termes generals, pels valors més elevats de *lr* (*lr*=0,001 i *lr*=0,004) s'assoleix major precisió i, conseqüentment, menor error. Pel que fa al *batch_size*, si eliminem les dades inestables efectivament es mostra una correlació en la qual a menor mida del lot, major precisió i, per conseqüent, menor error.

Finalment es pot constatar per aquest *dataset* en concret que el valor del ràtio d'aprenentatge afecta principalment a la pèrdua i, en conseqüència a la precisió mentre que la mida del lot afecta principalment al temps d'entrenament en general i de forma indirecta a la precisió.

Cal esmentar que els valors utilitzats per elaborar els gràfics figuren a *l'Annex C*.

5.2. Elecció del model

La elecció del model s'ha efectuat tenint en compte els resultats obtinguts per cada configuració a fi d'escollir el que presenti un major rendiment.

La configuració per la qual s'ha obtingut un valor més alt de precisió i més baix d'error ha sigut per un ràtio d'aprenentatge de 0,001 i una mida de lot de 4. Els valors obtinguts es mostren en la *Taula 6 i la Figura 51*.

	train_set	val_set
Accuracy	100.00%	97.50%
Loss	0.0022	0.0429

Taula 6: Mètriques de rendiment per lr=0.001 i batch_size=4. Font: elaboració pròpia

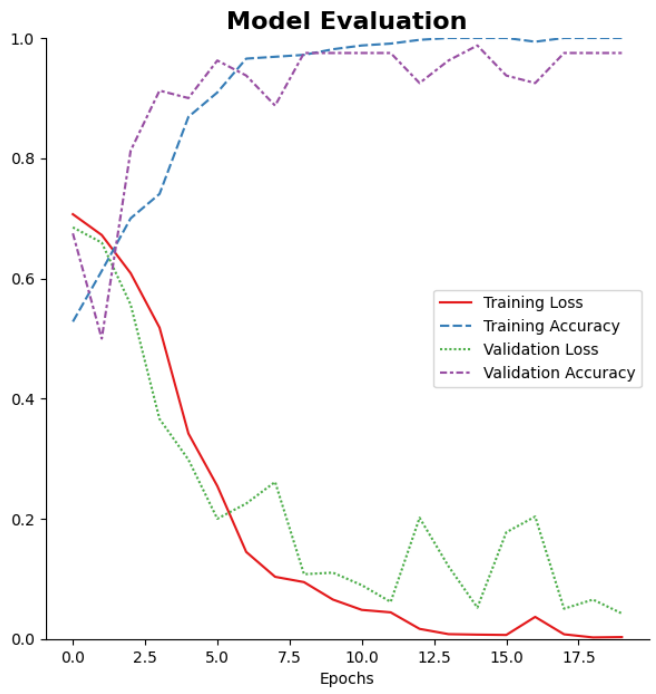


Figura 51:Gràfic de les mètriques de rendiment per lr=0.001 i batch_size=4. Font: elaboració pròpia

Per aquesta combinació d'hiperparàmetres el model assoleix la convergència al voltant de les 13 èpoques ja que a partir d'aquí la precisió es manté a un nivell aproximadament estable i no presenta millora. S'aconsegueix una precisió del 100% per les dades d'entrenament encara que una precisió lleugerament inferior per imatges que l'algoritme no ha vist amb un 97,50%. Aquesta precisió es considera acceptable ja que el model és capaç de classificar correctament el 97,50% de les imatges noves, tot i que es tracta d'una xifra millorable de cara a possibles implementacions futures. Es podria considerar un cas d'*overfitting* ja que sembla que la xarxa neuronal hagi après de memòria a classificar les dades d'entrenament i no ha après els patrons que hi resideixen, el motiu principal és la poca quantitat d'imatges en el *dataset*, la qual cosa es podria solucionar augmentant el nombre d'imatges d'entrenament. Tanmateix, la pèrdua és molt baixa amb un valor de 0.0022 i 0.0429 pel conjunt d'entrenament i validació respectivament, la qual cosa implica que l'algoritme ha sigut capaç d'arribar al mínim de la funció de cost.

Tal i com s'observa en la *Figura 51*, la pèrdua per les dades d'entrenament disminueix i la precisió augmenta a un ritme exponencial i força estable mentre que per les dades de validació s'observa una oscil·lació entre cada època. Aquest és un comportament típic i pot variar entre execucions i el motiu és el fet que l'objectiu de pèrdua és la de les dades d'entrenament per la qual cosa es veu directament afectat en el procés d'entrenament mentre que la pèrdua de validació es veu afectada indirectament de manera que serà més volàtil. Pel que fa a la eficiència computacional, l'algoritme ha executat les 1600 iteracions en poc més de 5 minuts.

5.3. Prediccions

Havent escollit el model, s'han realitzat prediccions per 6 imatges que l'algoritme no ha vist.

La *Figura 52* és una captura del que mostra per pantalla el programa després de l'execució del fitxer *predict.py*. En primer lloc, es mostra el valor de sortida de la xarxa neuronal i en segon lloc es mostra la predicció en format etiqueta de classificació OK/NOK.

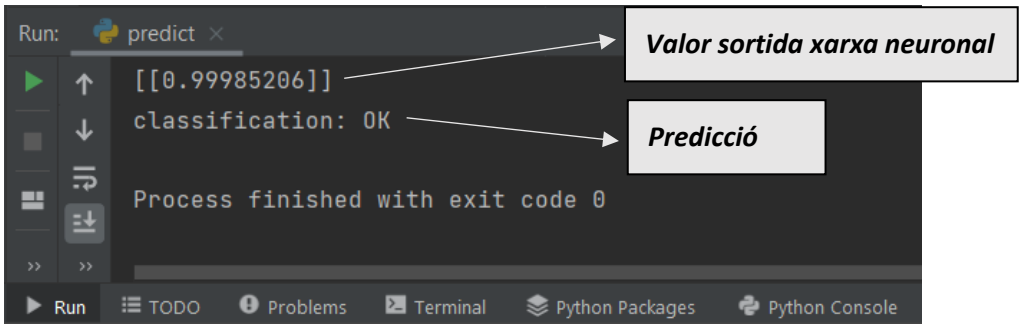


Figura 52: Captura de pantalla dels resultats de la predicció. Font: elaboració pròpia

Els resultats obtinguts per les 6 imatges es mostren en la *Taula 7*:

Imatge	Predicció	Valor de sortida xarxa neuronal
301.bmp	OK	0.99985206
305.bmp	OK	0.99989426
365.bmp	NOK	0.03599343
322.bmp	OK	0.99963140
371.bmp	NOK	0.00594696
370.bmp	NOK	0.00157022

Taula 7: Resultats de la predicció. Font: elaboració pròpia

Pel que fa a la interpretabilitat dels resultats, un valor de sortida de 1 indica que la xarxa neuronal està molt segura que la imatge d'entrada pertany a la classe OK. Un valor de sortida de 0 indica que està molt segura que la imatge pertany a la classe NOK. No és el mateix que el resultat de la predicció sigui NOK amb un valor de sortida de 0.001 que de 0.45, a vegades és important tenir en compte un altre factor: la confiança.

- S'entén **confiança** com el nivell de seguretat o fiabilitat amb la qual la xarxa neuronal classifica la imatge. Es representa en percentatge on 100% indica que la xarxa neuronal està completament segura de la classificació. Es calcula de la següent manera:

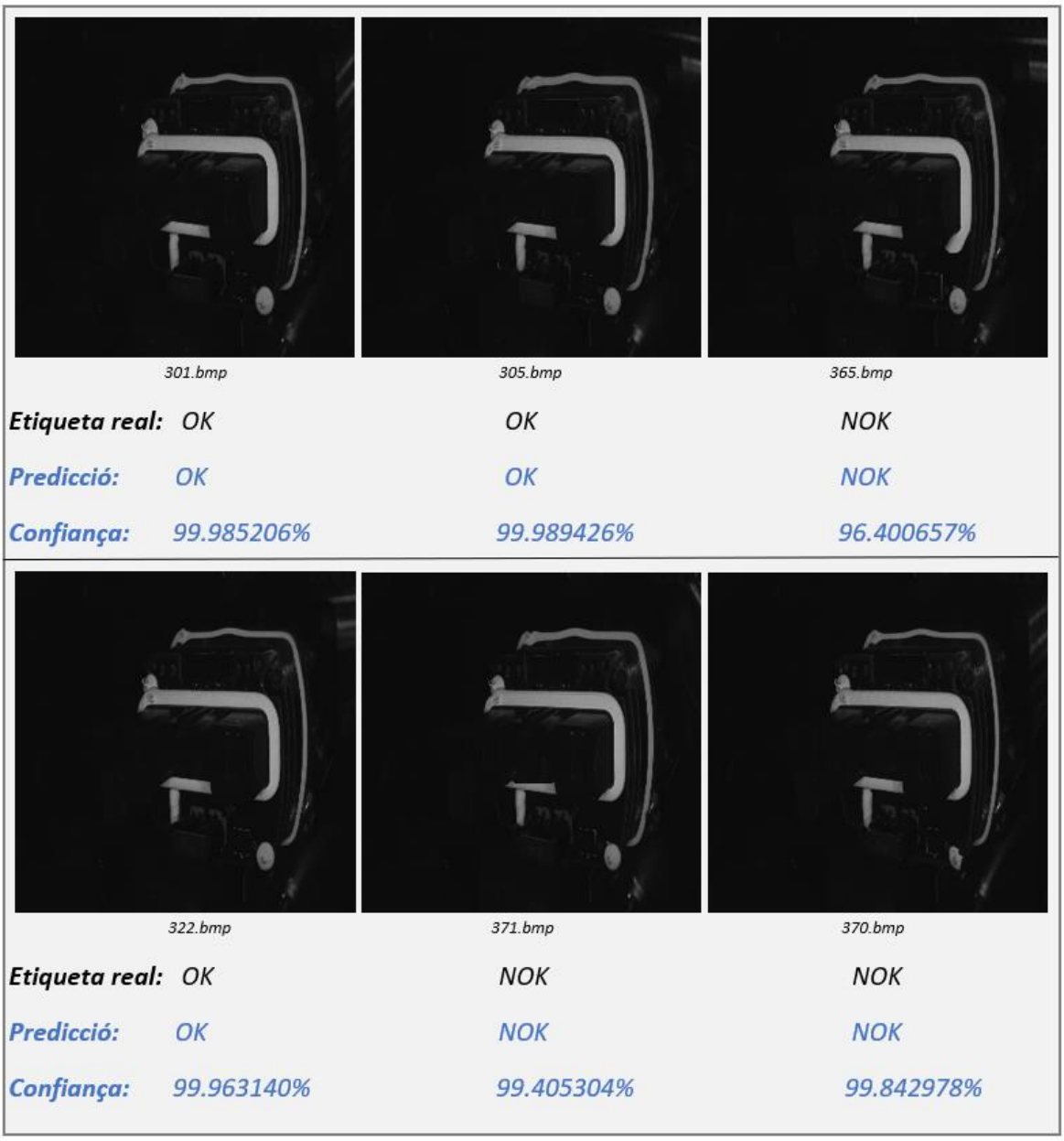
  - Si la predicció és OK:

$$confiança = valor\ sortida * 100$$

  - Si la predicció és NOK:

$$confiança = (1 - valor\ sortida) * 100$$

La *Taula 8* mostra per cada imatge la seva etiqueta real OK/NOK, la predicció OK/NOK i la confiança del model.



Taula 8: Prediccions de classificació i la seva etiqueta real. Font: elaboració pròpia

Les prediccions per aquestes 6 imatges han coincidit amb l'etiqueta real i la confiança de classificació del model en tots els casos és alta la qual cosa indica que els resultats són fiables.

S'han obtingut resultats satisfactoris i es verifica el bon rendiment de classificació obtingut amb l'elecció d'aquest model.

6. LÍNIES FUTURES

6.1. Millores

En aquest apartat s'exposen possibles millores del sistema intel·ligent de classificació d'imatges de cara a implementacions futures.

- Utilitzar mètodes més complexos i rigorosos per **l'optimització d'hiperparàmetres** que garanteixin valors més òptims com l'Optimització Bayesiana o Optimització Evolutiva basats en prediccions i estadística.

- Entrenar la xarxa neuronal amb **més imatges** per tal d'evitar l'*overfitting*.

    - Degut a que l'edició d'imatges no és la solució més òptima en la indústria, caldria buscar una altra manera per augmentar el conjunt de dades d'entrenament.

    - Implementar un **model d'aprenentatge continu** que vagi acumulant coneixement durant la vida útil de la màquina de manera que inicialment s'entrenen amb un primer conjunt de dades i en producció compten amb un sistema de detecció de novetat. D'aquesta manera, quan es detecta una nova casuística (imatge de la qual l'algoritme no té molta confiança per classificar-la) s'envia al sistema de supervisió remot, es cataloga per part d'un enginyer expert i s'afegeix al diccionari de casos. Es tracta d'una metodologia més complexa però fa front al problema de la falta de dades inicial.

- Per al complert control de qualitat del procés de dispensació de pasta es podrien realitzar **dues inspeccions** fent dues fotografies de la mateixa peça des de dos angles diferents de la càmera. L'entrada de la xarxa neuronal hauria de ser les dues imatges concatenades.

Altres

- Entrenar la xarxa neuronal amb les imatges **sense aplicar reducció de píxels** per la qual cosa es necessitaria una arquitectura molt més complexa de la xarxa neuronal i més temps d'entrenament però l'algoritme seria capaç de detectar moltes més característiques i detalls de forma més acurada i precisa. És veritat que pel *dataset* utilitzat en aquest treball no ha sigut necessari però per exemple, en fotografies de components electrònics es necessitaria molta més resolució d'imatge.

- Usar un algoritme d'aprenentatge no supervisat com ***k-means*** per segmentar la imatge en àrees similars que podria servir com una tècnica de reducció de dimensionalitat de les imatges abans d'entrar-les a una xarxa neuronal per la seva classificació. Especialment interessant en *datasets* complexos de color perquè faria reduir la complexitat de la xarxa neuronal.

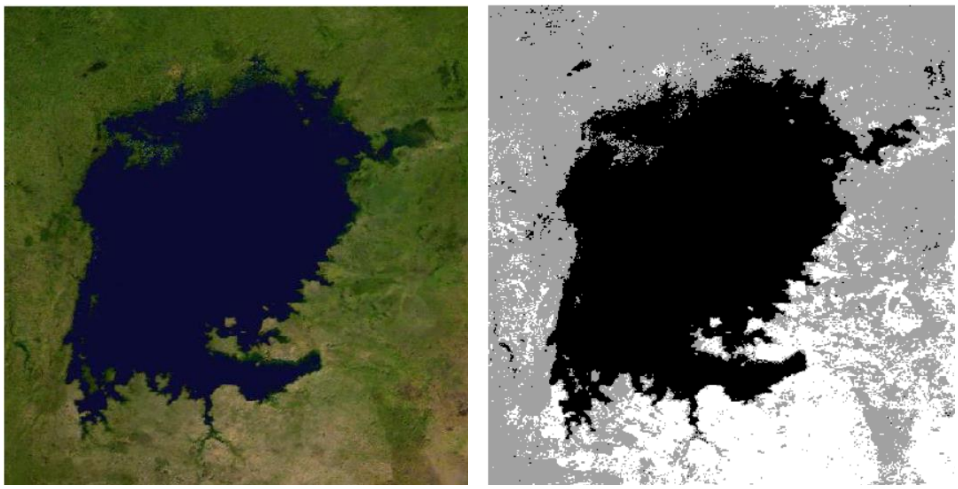


Figura 53: Segmentació d'imatges amb k-means. Font: (39)

6.2. Integració del model a la línia de producció

Al cap i a la fi, l'interès final en una indústria és la introducció del sistema intel·ligent de visió artificial a producció. Per aquest motiu, aquest apartat pretén abordar les possibles fases de la seva implantació per l'automatització del procés de control de qualitat.

Tal i com s'ha esmentat anteriorment, la inspecció visual es duu a terme en la mateixa estació on es realitza el procés productiu de dispensació de pasta.

La *Figura 54* mostra el diagrama de blocs conceptual del sistema intel·ligent que consta de dues fases.

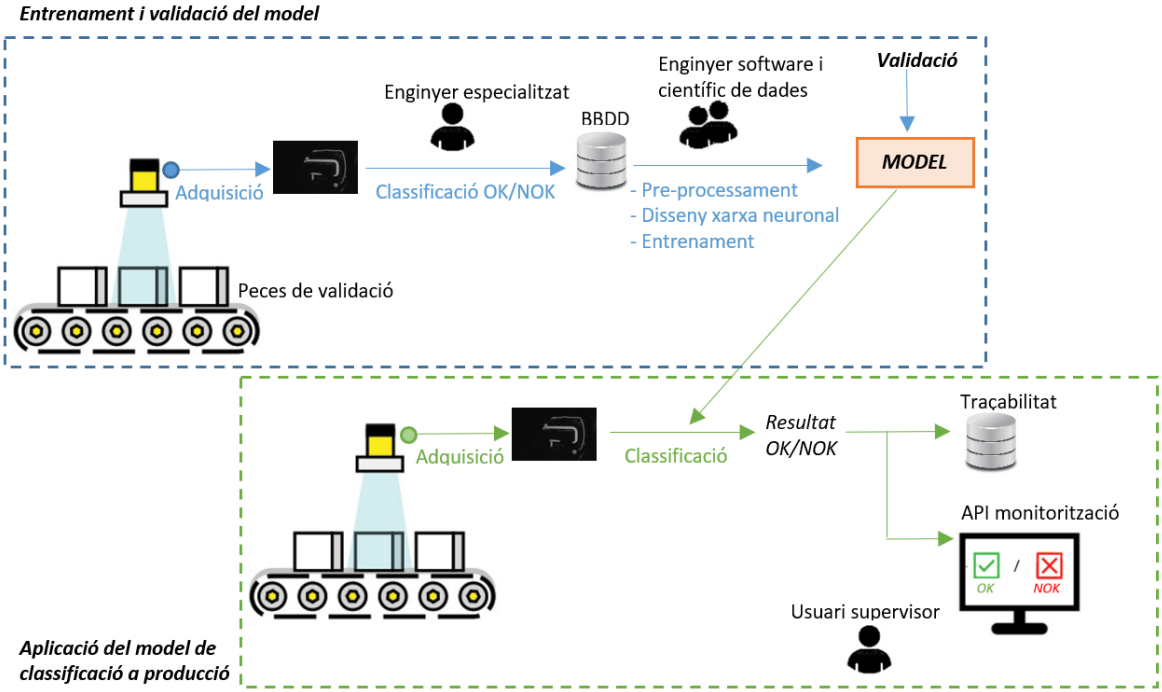


Figura 54: Diagrama conceptual fases integració del model de classificació en producció. Font: elaboració pròpia

La primera fase es duu a terme en les etapes de programació i validació de cada estació i procés de la línia de producció. En primer lloc es duu a terme l'adquisició d'imatges mitjançant visió artificial. Cada imatge adquirida és etiquetada com a peça defectuosa (NOK) o no defectuosa (OK) per part d'un enginyer especialitzat en el procés de dispensació en qüestió i amb capacitats per reconèixer si aquest procés ha sigut favorable o desfavorable en funció de la imatge capturada. Cada imatge etiquetada s'emmagatzema al servidor o base de dades pel seu posterior tractament. A continuació s'efectua el preprocessament de les imatges, disseny, programació i entrenament de la xarxa neuronal per implementar el model de classificació. Habitualment aquesta tasca és realitzada per enginyers de programari i científics de dades.

Finalment, el model obtingut és validat amb un altre conjunt de dades per assegurar el seu bon rendiment abans d'integrar-lo a producció. Els dos últims punts són els que s'han desenvolupat en l'apartat *4. Disseny i implementació del sistema intel·ligent* d'aquest treball.

La segona fase correspon a la etapa operativa i de producció de la línia. A diferència de la etapa de validació, les peces utilitzades són les peces que es subministren a client. La imatge de la peça amb la dispensació és adquirida pel sistema mitjançant visió artificial. Aleshores, a nivell de *software*, la imatge es converteix en la entrada d'un algoritme de classificació i, a nivell conceptual, la imatge és comparada amb el diccionari de casos que modelen aquest procés per tal de classificar la peça com a OK o NOK. Addicionalment, a nivell de traçabilitat, el resultat d'aquesta classificació juntament amb el codi d'identificació de la peça o del pallet es poden emmagatzemar en una base de dades. Gràcies a aquesta traçabilitat, es permet la comunicació entre estacions i, en cas de produir una peça defectuosa en la estació de dispensació, les següents estacions no manipularien la peça defectuosa o directament es rebutjaria i s'eliminaria de la línia de producció. A més a més, aquestes dades també són útils per realitzar anàlisis o indicadors d'empresa per computar l'efectivitat del procés de dispensació quantificant el número de peces defectuoses que s'han obtingut. Finalment, encara que aquest sistema intel·ligent estigui pensat per actuar de forma autònoma, seria adient supervisar el procés amb la integració d'una interfície de monitorització.

7. CONCLUSIONS

S'ha assolit el propòsit i objectius marcats a l'inici del treball amb èxit. S'ha seguit tot el procés d'implementació del sistema intel·ligent definit a l'inici del treball, des de l'elecció de l'algoritme fins la validació del model. S'ha aconseguit dissenyar i entrenar una xarxa neuronal convolucional capaç de classificar imatges amb uns resultats satisfactoris. La precisió obtinguda pel conjunt d'imatges de validació ha sigut alta amb un valor del 97.5% mentre que pel conjunt d'imatges d'entrenament ha sigut del 100%. A més a més, la pèrdua (error) en l'aprenentatge de l'algoritme ha resultat mínima amb un valor del 0.0429 i encara més baixa pel conjunt d'imatges d'entrenament amb un valor del 0.0022.

Aquest sistema ha estat dissenyat per introduir-lo en un procés de control de qualitat industrial per classificar peces defectuoses i no defectuoses a partir d'imatges adquirides per visió artificial. Tot i així, la seva integració a producció quedava fora de l'abast del treball.

Per altra banda, s'ha vist que disposar d'un *dataset* representatiu del procés és fonamental per obtenir un model de classificació fidel. De la mateixa manera, s'ha corroborat que una de les principals problemàtiques en l'entrenament, és el sobreajust del model degut principalment a poques dades d'entrenament. Molt probablement, si s'augmentés el conjunt d'imatges d'entrenament s'evitaria aquest sobreajust i s'obtindria encara millor rendiment per les imatges de validació.

Tanmateix, s'ha observat la influència dels hiperparàmetres *batch_size* i *lr* en el rendiment final del model.

L'Aprenentatge Automàtic aporta nombrosos beneficis entre els quals es troba la reducció de costos per l'automatització de tasques que no aporten valor afegit i, per tant, l'increment del rendiment productiu total. Ofereix una solució elegant a necessitats reals en la indústria.

Es conclou que els models de classificació elaborats mitjançant xarxes neuronals ofereixen una alternativa rendible, assequible i viable d'implementar que fa front a les limitacions del programari de lògica condicional en controls de qualitat industrials.

Finalment m'agradaria afegir que aquest Treball Final de Grau ha sigut un gran repte a nivell personal i acadèmic així com també una experiència enriquidora. A l'inici era conscient que es tractava d'una temàtica de la qual no disposava de coneixements previs i això comportaria realitzar un estudi i investigació exhaustius. Durant el desenvolupament del treball s'han enfrontat dificultats, s'han buscat solucions, s'han empleat metodologies per arribar a obtenir un *dataset* òptim, s'han entès les bases teòriques i s'han aplicat al cas d'estudi i *dataset* concrets.

Aquest treball no pretén promoure el reemplaçament de tot el codi del món per xarxes neuronals però sí identificar aquells casos en els quals ofereix una oportunitat de negoci i una solució efectiva. Una vegada identificat el cas d'aplicació, es requereix d'un estudi previ del *dataset* i elecció de la tipologia i hiperparàmetres de la xarxa neuronal més adients en funció d'una sèrie de factors.

Algunes opinions afirmen que les expectatives són massa altes i que es tornaran a repetir els errors passats que ens portaran a viure un altre període "d'Hivern de la I.A."; no obstant, l'impacte que està tenint la I.A. en qualsevol àmbit demostra el contrari. La revolució és real i tot just acaba de començar, l'Aprenentatge Automàtic i les xarxes neuronals són algoritmes innovadors que obren un nou món de possibilitats.

8. REFERÈNCIES BIBLIOGRÀFIQUES

1. **Schwab, Klaus.** *La Cuarta Revolución Industrial.* 001. s.l. : Debate, 2016. ISBN 9788499926940.

2. **Santana, Carlos.** Canal de divulgació. *DotCSV.* [En línia] [Consulta: 15 / 01 / 2021.] Disponible a: <https://www.youtube.com/channel/UCy5znSnfMsDwaLlROnZ7Qbg>.

3. **Qian, Jingyu.** A Survey on Sentiment Classification in Face Recognition. [En línia] 2018. [Consulta: 18 / 04 / 2021.] Disponible a: <https://iopscience.iop.org/article/10.1088/1742-6596/960/1/012030>.

4. **Turing, Alan.** *Computing Machinery and Intelligence.* 1950. ISBN 9786133619555.

5. **Canencia, Óscar López.** El hombre contra la máquina: 25 años del primer duelo entre "Deep Blue" i Gary Kasparov. Revista rtve. [En línia] 2021. [Consulta: 29 / 04 / 2021.] Disponible a: <https://www.rtve.es/deportes/20210210/ajedrez-kasparov-deep-blue-aniversario/2074420.shtml>.

6. **Rumelhart, David E., Hinton, Geoffrey E. i Williams, Ronald J.** *Learning representations by back-propagating errors.* s.l. : Nature 323, 533-536, 1986.

7. **Simon, Matt.** Inside the Amazon Warehouse Where Humans and Machines Become One. [En línia] 2019. [Consulta: 23 / 04 / 2021.] Disponible a: <https://www.wired.com/story/amazon-warehouse-robots/>.

8. **IPA, Faunhofer.** Rob-aKademI. [En línia] 2020. [Consulta: 21 / 04 / 2021.] Disponible a: <https://www.ipa.fraunhofer.de/en/reference_projects/rob-akademi.html>.

9. **McKinsey Global, Institute.** Jobs lost, jobs gained: Workforce transitions in a time of automation. [En línia] 1969. [Consulta: 15 / 04 / 2021.] Disponible a: <https://www.mckinsey.com/featured-insights/future-of-work/jobs-lost-jobs-gained-what-the-future-of-work-will-mean-for-jobs-skills-and-wages>.

10. **Gartner.** Gartner Survey Shows 37 Percent of Organizations Have Implemented AI in Some Form. [En línia] 2019. [Consulta: 15 / 05 / 2021.] Disponible a: <https://www.gartner.com/en/newsroom/press-releases/2019-01-21-gartner-survey-shows-37-percent-of-organizations-have>.

11. **Loukides, Mike i O'Reilly.** AI Adoption in the Enterprise 2021. [En línia] 2021. [Consulta: 20 / 04 / 2021.] Disponible a: <https://www.oreilly.com/radar/ai-adoption-in-the-enterprise-2021/>.

12. **Solé, Jaume Miralles.** *Proyectos de Inteligencia Artificial.* 2020. ISBN 9781661199456.

13. **Institute, IBM.** Shifting toward Enterprise-grade AI. [En línia] 2018. [Consulta: 24 / 04 / 2021.] Disponible a: <https://www.ibm.com/thought-leadership/institute-business-value/report/enterpriseai#>.

14. **Wikipedia.** Wikipedia. [En línia] 2021. [Consulta: 19 / 03 / 2021.] Disponible a: <https://en.wikipedia.org/wiki/Quality_control>.

15. **Microscan.** Microscan. [En línia] [Consulta: 15 / 04 / 2021.] Disponible a: <https://www.calvek.com/vision-artificial/ >.

16. **Dabhi, Ravirajsinh.** Kaggle. [En línia] 2020. [Consulta: 20 / 05 / 2021.] Disponible a: <https://www.kaggle.com/ravirajsinh45/real-life-industrial-dataset-of-casting-product>.

17. **Diazaraque, Juan Miguel Marín.** *Introducción a las redes neuronales aplicadas.* 2012.

18. **Lorenz, Edward.** La teoría del caos. 1963.

19. **Kobran, Daniel i Banys, David.** AI Wiki. [En línia] 2019. [Consulta: 20 / 02 / 2021.] Disponible a: <https://docs.paperspace.com/machine-learning/wiki/activation-function>.

20. **Mathworks.** Matlab para inteligencia artificial. [En línia] [Consulta: 20 / 02 / 2021.] Disponible a: <https://es.mathworks.com/>.

21. **Bagnato, Juan Ignacio.** ¿Cómo funcionan las Convolutional Neural Networks? Visión por Ordenador. [En línia] 29 / 11 / 2018. [Consulta: 12 / 03 / 2021.] Disponible a: <https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/>.

22. **Queguiner, Jean-Louis.** OVHcloud. [En línia] 15 / 02 / 2019. [Consulta: 10 / 03 / 2021.] Disponible a: <https://www.ovh.com/blog/deep-learning-explained-to-my-8-year-old-daughter/>.

23. **Powell, Victor.** Image Kernels. [En línia] [Consulta: 26 / 03 / 2021.] Disponible a: <https://setosa.io/ev/image-kernels/>.

24. **Torres, Jordi.** Jordi Torres. AI. [En línia] [Consulta: 16 / 03 / 2021.] Disponible a: <https://torres.ai/deep-learning-inteligencia-artificial-keras/>.

25. **Saha, Sumit.** A Comprehensive Guide to Convolutional Neural Networks. [En línia] 15 / 12 / 2018. [Consulta: 23 / 03 / 2021.] Disponible a: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>.

26. **Brownlee, Jason.** Difference Between a Batch and an Epoch in a Neural Network. [En línia] 2018. [Consulta: 20 / 03 / 2021.] Disponible a: <https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/>.

27. **Sharma, Sagar.** Epoch vs Batch Size vs Iterations. [En línia] 2017. [Consulta: 26 / 03 / 2021.] Disponible a: <https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9>.

28. **Algorithmia.** Introduction to loss functions. [En línia] 2018. [Consulta: 05 / 04 / 2021.] Disponible a: <https://algorithmia.com/blog/introduction-to-loss-functions>.

29. **Brownlee, Jason.** Gentle Introduction to the Adam Optimization. [En línia] 2017. [Consulta: 20 / 03 / 2021.] Disponible a: <https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/>.

30. **Singh, Seema.** Understanding the Bias-Variance Tradeoff. [En línia] 21 / 03 / 2018. [Consulta: 16 / 03 / 2021.] Disponible a: <https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229>.

31. **Python.** Python. [En línia] [Consulta: 11 / 03 / 2021.] Disponible a: <python.org>.

32. **TensorFlow.** TensorFlow. [En línia] [Consulta: 30 / 2 / 2021.] Disponible a: <https://www.tensorflow.org/>.

33. **Keras.** Keras. [En línia] [Consulta: 23 / 02 / 2021.] Disponible a: <https://keras.io/>.

34. **PyCharm.** PyCharm. [En línia] [Consulta: 17 / 03 / 2021.] Disponible a: <https://www.jetbrains.com/pycharm/>.

35. **Photoshop, Adobe.** Adobe Photoshop. [En línia] [Consulta: 14 / 03 / 2021.] Disponible a: <https://www.adobe.com/es/products/photoshop.html>.

36. **Developer, NVIDIA.** CUDA Toolkit. [En línia] [Consulta: 21 / 03 / 2021.] Disponible a: <https://developer.nvidia.com/cuda-toolkit-archive>.

37. **Developer, NVIDIA.** cuDNN. [En línia] [Consulta: 21 / 03 / 2021.] Disponible a: <https://developer.nvidia.com/cudnn>.

38. **Frederickson, Ben.** An Interactive Tutorial on Numerical Optimization. [En línia] 25 / 11 / 2016. [Consulta: 15 / 04 / 2021.] Disponible a: <http://www.benfrederickson.com/numerical-optimization/>.

39. **Oviedo, Universidad.** El algoritmo k-means aplicado a clasificación y procesamiento de imágenes. [En línia] [Consulta: 25 / 06 / 2021.] Disponible a: <https://www.unioviedo.es/compnum/laboratorios_py/new/kmeans.html>.

40. **Ahmed, Bubli.** Amazon warehouse robots. [En línia] 07 / 12 / 2016. [Consulta: 29 / 04 / 2021.] Disponible a: <https://www.youtube.com/watch?v=4sEVX4mPuto>.

41. **IPA, Fraunhofer.** Fraunhofer IPA. [En línia] 01 / 07 / 2020. [Consulta: 20 / 03 / 2021.] Disponible a: <https://www.ipa.fraunhofer.de/en/reference_projects/rob-akademi.html>.

42. **Brownlee, Jason.** Use Early Stopping to Halt the Training of Neural Networks At the Right Time. [En línia] 10 / 12 / 2018. [Consulta: 25 / 03 / 2021.] Disponible a: <https://machinelearningmastery.com/how-to-stop-training-deep-neural-networks-at-the-right-time-using-early-stopping/>.