Sistema d'Identificació d'Articles

Alumne: Ana Mª Vela Dies
Enginyeria d'informàtica

Consultor: David Masip Rodo

Curs 2015-16, 2on semestre

Aquesta obra està subjecta a una llicència de Reconeixement-NoComercial-SenseObraDerivada 3.0 Espanya de Creative Commons

Copyright © 2016 Ana Mª Vela Dies.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
A copy of the license is included in the section
entitled "GNU Free Documentation License".

C) Copyright

© Ana Mª Vela Dies
Reservats tots els drets. Està prohibit la reproducció total o parcial d'aquesta obra per qualsevol mitjà o procediment, compresos la impressió, la reprografia, el microfilm, el tractament informàtic o qualsevol altre sistema, així com la distribució d'exemplars mitjançant lloguer i préstec, sense l'autorització escrita de l'autor o dels límits que autoritzi la Llei de Propietat Intel•lectual

Agraïments

Gràcies a totes les persones que en algun moment o altre m'han acompanyat en aquest camí, formeu part d'aquest projecte.

Al meu amic Àngel, amb qui vaig començar aquesta aventura ja fa uns quants anys. Ningú millor que tu sap quantes hores i esforç li hem dedicat. Gràcies per estar al meu costat, animar-me a continuar als moments complicats i donar-me l'empenta inicial per tirar endavant aquest projecte.

Al Sr.Maggi, i dic senyor no per l'edat sino perquè ets GRAN en molts aspectes! Un dels millors professionals amb qui he tingut la oportunitat de treballar, però encara millor amic. Probablement la persona que més ha cregut en mi i m'ha fet creure a mi també. Gràcies per estar sempre al meu costat i per l'empenta final en forma de col·laboració amb aquest projecte.

A la meva família, als meus amics i companys pel vostre recolzament i la vostra comprensió. No posaré noms perquè segur que em deixaria algú, però vosaltres ja sabeu de qui estic parlant! Se que aquests mesos us he dedicat menys temps del habitual, que hi han hagut moments on m'heu vist més seria o amb menys paciència de la normal. Gràcies de veritat per donar-me l'espai que necessitava, pels vostres ànims i per la vostra comprensió.

Als coneguts que, en algun moment o altre, han mostrat interès en aquest projecte. La vostra cara de fascinació quan us explicava en què consistia el projecte m'ha donat força per continuar.

Finalment, voldria agrair al meu tutor, David Masip, la seva atenció i interès en aquest treball. Gràcies per animar-me en cada entrega perquè això m'ha fet continuar treballant amb més ganes i motivació cap al final. M'emporto un bon record final de l'ensenyament a distància, i de com, a vegades, no importa el tipus d'ensenyament sino la voluntat d'ensenyar i aprendre. I tu n'has tingut voluntat d'ensenyar-me i ajudar-me en tot moment. Moltes gràcies perquè sense la teva ajuda i les teves orientacions aquest projecte no hagués estat possible.

Maig 2016, Barcelona

.

FITXA DEL TREBALL FINAL

Títol del treball:	*Sistema d'Identificació d'Articles*
Nom de l'autor:	*Ana Mª Vela Dies*
Nom del consultor:	*David Masip Rodo*
Data de lliurament (mm/aaaa):	*06/2016*
Àrea del Treball Final:	*Visió per Computador*
Titulació:	*Enginyeria d'Informàtica*

Resum del Treball (màxim 250 paraules):

En el mon de la logística, l'ús de components afegits al producte en forma de codi de barres, chips RFID, codis QR, etc.. és habitual a l'hora d'identificar els productes. Fins arribar al punt de convertir-se en elements imprescindibles dintre de la cadena de subministrament.

Els avenços en diferents camps de la intel·ligència artificial, especialment en la visió per computador, han permès els últims anys desenvolupar projectes amb una precisió semblant a la que caracteritza a la visió humana. Fins a arribar a tal punt, que comencen a existir aplicacions quotidianes que fan ús d'aquestes tecnologies.

El present projecte pretén explorar algunes de les tècniques i mecanismes destacats dintre de la visió per computador, per acabar desenvolupant e implementant una solució dintre del camp del reconeixement d'objectes que permeti, a partir d'una imatge, reconèixer de quina referència es tracta dintre d'un magatzem o un entorn productiu.

La implementació d'aquesta solució es realitzarà fent ús de diferents llibreries sota el llenguatge de programació de codi obert Python, dissenyant una xarxa neuronal convolucional, (CNN – Convolutional Neural Network): un tipus de xarxa que modela de forma molt simplificada el funcionament del cervell humà. L'aprenentatge es basa en la cerca de característiques comuns en petits grups, i en l'entrenament de la xarxa perquè cada cop millorin els resultats amb l'experiència.

Abstract (in English, 250 words or less):

In logistics, technologies such as barcodes, RFID chips, QR codes, etc. are so commonly used to identify products that they have become essential in the supply chain.

Lately, improvements in the Artificial Intelligence area, specially in computational vision field, have allowed the development of projects with a high precision, close to human vision, and its spread of use is so extended that common applications are using this technology.

This project explores some of the most popular techniques and mechanisms used in computational vision. The aim is to develop and implement a solution for objects recognition that allows, from an image, to obtain its product reference within the warehouse.

The implementation of this solution is based on different libraries of Python open source programming language. A Convolutional Neural Network (CNN) will be designed. This is a kind of networks that models in a very simplified way the functioning of human brain. Learning process is based on the finding of common characteristics in small groups and on the training of the networks so that results are enhanced with experience.

Paraules clau (entre 4 i 8):

Machine Learning, Deep Learning, Convolutional Neural Network, Python, Tensorflow, GPU

Índex

1. Introducció 1
1.1 Context i justificació del Treball 1
1.2 Objectius del Treball 1
1.3 Enfocament i mètode seguit 2
1.4 Planificació del Treball 3
1.5 Breu sumari de productes obtinguts 4
1.6 Breu descripció dels altres capítols de la memòria 4
2. Entorn de treball 6
2.1 Sistema Operatiu Os X 6
2.2 Python 6
2.3 Anaconda 6
2.4 Spyder 7
2.5 VLC Media Player 7
2.6 FileZilla 7
2.7 Referències web de la descàrrega del software 7
3. Estudi i Formació de la Tecnologia Actual 8
3.1 Arquitectura d'una xarxa neuronal. 8
3.2 Aprenentatge d'una xarxa neuronal 10
4. Estudi i formació dels projectes Deep Leaning existents 11
4.1 TensorFlow. 11
4.2 Classificació de dígits amb la base de dades de MNIST 12
4.3 Classificació d'imatges amb la base de dades de CIFAR10 14
5. Definició del Prototip Inicial 15
5.1 Capa d'entrada (Input Layer). 15
5.2 Capa de convolució 1. 16
5.3 Capa de convolució 2. 17
5.4 Capa Full Connected i Capa SofMax 17
5.5 Resum de característiques 18
6. Obtenció i Tractament d'imatges 19
6.1. Obtenció de les imatges 19
6.2. Generació de les imatges del conjunt d'entrenament. 20
6.3. Definició de les classes inicials 21
7. Implementació del Prototip Inicial 25
7.1. Estructura de fitxers 25
7.2. Capa d'entrada 26
7.3. Capes de convolució, Capa Full Connected i Capa SofMax 27
7.4. Mètode d' Entrenament i Mètode de Test 28
8. Resultats Obtinguts 30
8.1. Joc de Proves 1. Execució amb 10 classes i processament previ 1 30
8.2. Joc de Proves 2. Execució amb 10 classes i processament previ 2 31
8.3. Joc de Proves 3. Execució amb 20 classes. 32
9. Implementació Final 34
9.1. Estructura de fitxers. 34
9.2. Capa d'entrada, de Convolució, Full Connected i SoftMax 34
10. Resultats Finals Obtinguts 40
10.1. Joc de Proves 1. Execució amb 20 classes en GPU i 270 iteracions (dropout = 1). 40
10.2. Joc de Proves 2. Execució amb 30 classes en GPU i 270 iteracions (dropout = 1). 42
11. Conclusions 44
12. Millores i possibles línies de futur 45

12.1 Obtenció i Tractament d'imatges 45
12.2 Mètode d'entrenament i Test 45
11.3 Altres possibles millores 45
13. Glossari 46
14. Bibliografia 47
14.1. Publicacions. 47
14.2. Enllaços electrònics 47
15. Annexos 49
15.1 Annex 1. SIA_Train.py 49
15.2 Annex 2. SIA.py 52
Annex 3. Jocs de Prova 54

Llista de figures

Il·lustració 1 - Arquitectura d'una xarxa neuronal. 8
Il·lustració 2 - Representació de dígits en MNIST 12
Il·lustració 3 - Estructura de dades MNIST 12
Il·lustració 4 - Representació de imatges en CIFAR10 14
Il·lustració 5 - Representació Prototip Inicial SIA 15
Il·lustració 6 - Representació d'imatges SIA (Prototip amb 20 classes) 15
Ilustración 7 - Representació Capa de Convolució 1 SIA 16
Ilustración 8 - Representació Capa de Convolució 2 SIA 17
Il·lustració 9 - Obtenció d'imatges 19
Il·lustració 10 - Generació d'imatges Classe 0 20
Il·lustració 11 - Evolució accuracy i loss (prototip 10 classes) 30
Il·lustració 12 - Evolució accuracy i loss (prototip 10 classes) 31
Il·lustració 13 - Evolució accuracy i loss (prototip 20 classes) 33
Il·lustració 14 - Representació Prototip SIA amb imatges de 64x64p 35
Il·lustració 15 - Representació Prototip SIA amb imatges de 64x64p (pooling 2x2) 36

1. Introducció

1.1 Context i justificació del Treball

El present Projecte Final de Carrera (PFC) pretén apropar la tecnologia que permet reconèixer imatges, al sector de la logística interna dels magatzems o centres logístics. Es tracta d'un entorn on tots els processos tendeixen a automatitzar-se (robots, màquines automatitzades, etc..) i on constantment es busquen solucions per identificar els productes (etiquetes, codis de barres, chips RFIP, etc..). Solucions, per altra banda, que suposen un cost afegit per cada unitat de producte.

Dintre de la visió per computador, el meu projecte es centrarà en el camp del **reconeixement d'objectes.** L'objectiu és desenvolupar una aplicació, podríem anomenar-la **Sistema de Identificació d'Articles (SIA),** que permeti, a partir d'una imatge o foto, reconèixer de quin objecte es tracta. Per això serà necessari treballar amb una base de dades dels articles existents, i donada una foto captada amb una càmera, el sistema reconeixerà de quin article es tracta.

Existeixen diferents formes d'implementar-ho, però s'optarà per una "**Convolutional Neural Network**" o CNN, que modela de forma molt simplificada el funcionament del cervell estructurant-ho en capes, on cada neurona rep connexions entrants amb algunes neurones (no totes) de capes anteriors. L'aprenentatge es basa en la cerca de característiques comuns en petits grups d'entrada (p.ex. pixels), i en l'entrenament de la xarxa perquè cada cop millorin els resultats amb la experiència. Aquest tipus de xarxa té un funcionament molt efectiu per tasques de visió artificial, especialment en la classificació d'imatges.

1.2 Objectius del Treball

A continuació es descriuen els objectius generals i específics del projecte final de carrera.

Objectius Generals

En acabar aquest projecte, s'espera haver desenvolupat una aplicació/script que permeti a partir d'una foto, saber de quin material es tracta.

Actualment l'identificador d'un article dintre d'un magatzem/botiga és la seva etiqueta o codi de barres. Amb aquest projecte, es pretén que l'identificador d'un article sigui ell mateix. Prenent un foto de l'article en qüestió, es podria saber de què es tracta i, a partir d'aquí, tenir tota la informació associada al material (per exemple: la referència, descripció o preu).

Inicialment es treballarà amb una base de dades d'imatges petita, que es pot anar desenvolupant i fent créixer en funció del cost computacional necessari.

Objectius Específics

Per aconseguir aquests objectius generals serà necessari assolir uns objectius més específics, com són:

- Comprendre els conceptes de Machine Learning i Deep Learning.
- Entendre el funcionament d'una CNN, el seu disseny i estructura.
- Conèixer la programació del Deep Learning,
- Conèixer els diferents tipus de sistemes d'aprenentatge
- Conèixer les diferents eines per la gravació de vídeos i tractament d'imatges.
- Treballar amb l'ús del software Open Source de Google anomenat TensorFlow sota Python per tal de desenvolupar el sistema d'aprenentatge i de test.

1.3 Enfocament i mètode seguit

Dintre del Deep Learning o aprenentatge automàtic, TensorFlow és un sistema d'intel·ligència artificial de Google que està especialitzat en el reconeixement de veu, text e imatges. Al novembre de 2015 Google va alliberar el codi perquè tothom pogués utilitzar-ho, i a partir d'aquest moment ha esdevingut un referent en el món del reconeixement automàtic. És per això, que l'enfocament d'aquest projecte i el mètode seguit es basarà en l'ús i estudi de l'eina **TensorFlow**, pilar bàsic del projecte.

Com es podrà veure de forma més detallada a la planificació del projecte, s'ha optat per dividir el projecte en diferents fases, dedicant la primera part a l'estudi de la tecnologia i a la comprensió del projectes existents.

Respecte a l'ús de la **tecnologia**, com ja s'ha comentat, TensorFlow serà el pilar bàsic del projecte. Però al seu entorn hi conviuran altres programaris que faran més fàcil el seu ús (entorn de desenvolupament, eines de debug, etc...), o bé que el complementaran (llibreries especialitzades en imatges, processament matemàtic, etc...). Més endavant es dedicarà un capítol a aquest tema.

Respecte a l'estudi dels **projectes existents**, n'hi han dos que han sigut especialment importants: MNIST y CIFAR-10. El primer, tracta sobre el reconeixement de dígits: es a dir, donada una imatge, obtenir les possibilitats de que sigui cadascú dels 10 possibles dígits (0-9). El segon, tracta sobre la classificació d'imatges: és a dir, donada una imatge, classificar-la dintre de les 10 categories existents. N'hi hagut més projectes, però aquests dos juntament amb la lectura del llibre del Jordi Torres han sigut el punt inicial d'aquest projecte.

A partir de la comprensió d'aquests dos projectes, i de la construcció de un entorn de treball on poder posar en marxa els projectes existents, ha nascut aquest projecte. Inicialment amb la definició d'un **prototip** basat en una xarxa neuronal convolucional o CNN, que més tard serà desenvolupat, entrenat i avaluat per acabar obtenint el **producte final**.

1.4 Planificació del Treball

El projecte s'ha dividit en 4 grans fases que coincideixen amb el lliurament de cada una de les PACs i amb el lliurament final:

- **Fase 1:** Es farà la descripció del projecte, es fixaran els objectius i el seu abast. En aquesta etapa es farà l'anàlisi de tot el programari necessari, els algoritmes existents, i s'estudiaran les diferents alternatives possibles. En resum, es començarà a definir l'entorn de treball amb el qual es treballarà. Les principals activitats seran:
    - Lectura i comprensió dels recursos oferts pel tutor.
    - Recerca d'informació a Internet sobre les aplicacions a utilitzar (comprensió a alt nivell) per ajudar a fer estimacions.
    - Primera presa de contacte amb el software necessari: Python, TensorFlow, etc...
    - Elaboració del pla de treball.
    - Entrega PAC 1

- **Fase 2:** es definirà el prototip inicial del projecte, prenent com a model els algoritmes de reconeixement d'objectes existents. Les principals activitats seran:
    - Instal•lació i aprenentatge del software necessari: Python, TensorFlow, etc...
    - Recerca d'informació sobre la transformació d'imatges.
    - Recerca sobre els diferents algoritmes, tipus de xarxes, etc.. i determinació de quin és el més adient pel problema plantejat.
    - Disseny del prototip inicial.
    - Elaboració de la Memòria del Projecte.
    - Entrega PAC 2

- **Fase 3:** es desenvoluparà el prototip inicial i s'obtindran els primers resultats. Les principals activitats seran:
    - Captació i processament de fotos, a partir de gravacions de vídeo.
    - Estudi i formació sobre l'arquitectura del model triat.
    - Adaptació dels paràmetres de la xarxa (CNN) segons les necessitats del model.
    - Proves i resultats inicials.
    - Elaboració de la Memòria del Projecte.
    - Entrega PAC 3

- **Fase 4:** es finalitzarà el projecte. S'obtindran els resultats finals, s'extrauran conclusions i es finalitzarà la memòria del projecte. Les principals activitats seran:
    - Adaptacions i correccions del codi: implementació final.
    - Elaboració de la Memòria Final del Projecte.
    - Entrega Projecte Final.
    - Elaboració de la Presentació Virtual.

A continuació es pot veure un resum de la descomposició en tasques i la distribució en temps (es pot veure amb més detall a l'arxiu de MS Project):

Nom de la Tasca	Duració (dies)	Inici	Fi
Definició del Projecte	6	24/02/2016	01/03/2016
Definició de l'entorn de treball	6	02/03/2016	09/03/2016
Definició Prototip Inicial	24	10/03/2016	06/04/2016
Desenvolupament Prototip Inicial	24	07/04/2016	04/05/2016
Finalització del Projecte	24	05/05/2016	01/06/2016

1.5 Breu sumari de productes obtinguts

A la fi d'aquest projecte s'ha obtingut una aplicació desenvolupada en codi Python, que consta de dos fitxers:

1. SIA.py: fitxer a on es defineix l'estructura de les dades, i les operacions bàsiques de tractament de les imatges.
2. SIA_train.py: fitxer a on es defineix el model de la xarxa, el mètode d'aprenentatge i el de test.

Aquests dos fitxers interactuen amb un conjunt de dades d'entrenament i test, formades a partir d'imatges reals que s'han pres al magatzem. Per tal de facilitar l'execució, s'han construït dos arxius amb el contingut de les imatges:

1. SIA_data.npz: fitxer amb 500 imatges d'entrenament de cada una de les 30 classes finals.
2. SIA_test.data.npz: fitxers amb 50 imatges de test de cada una de les 30 classes finals.

Els resultats de l'execució s'escriuen en uns fitxers de log, que es poden visualitzar amb l'ús de l'eina TensorBoard. Aquests fitxers mostren l'evolució del procés d'aprenentatge i permeten visualitzar algunes variables clau del model, així com comparar diferents execucions del codi.

1.6 Breu descripció dels altres capítols de la memòria

El present document consta de diferents capítols que fan referència a aspectes importants del projecte. A continuació es farà una breu descripció de cada un d'ells:

- **Capítol 2. Entorn de treball.**
  En aquest apartat es descriu l'entorn de treball necessari per a la realització del PFC. S'ha realitzat un repàs al programari i les eines necessàries per a construir el Sistema d'Identificació d'Articles.

- **Capítol 3. Estudi i Formació de la Tecnologia Actual.**
  En aquest apartat es fa una introducció a diferents fets destacats de la tecnologia actual, com són:
  - els principals tipus d'algoritmes de Machine Learning (supervisat / no supervisat),
  - les diferents formes de implementació,descrivint amb més detall les xarxes neuronals convolucionals, que serà la que s'utilitzarà per implementar el projecte.
  - els processos d'aprenentatge i avaluació d'un xarxa neuronal.

- **Capítol 4. Estudi i Formació de projectes Deep Learning existents.**
  En aquest apartat es presenten dos projectes que s'han utilitzat per entendre el funcionament intern de l'aprenentatge automàtic. Permeten veure el procés complet de creació, entrenament i avaluació d'un model de xarxa neuronal convolucional.

- **Capítol 5. Definició del prototip inicial.**
  En aquest apartat es presenta el prototip dissenyat inicialment del SIA, presentant el model i definint la seva arquitectura. Així mateix, es presenten algunes de les seves característiques principals.

- **Capítol 6. Obtenció i Tractament d'imatges.**
  En aquest capítol es presentaran les diferents tècniques que s'han realitzat per tal d'aconseguir les imatges que formaran part del conjunt de dades d'entrenament i test.

- **Capítol 7. Implementació del Prototip Inicial.**
  En aquest capítol es descriurà com s'ha implementat el prototip.

- **Capítol 8. Resultats Obtinguts (Prototip).**
  En aquest capítol es presentaran el primers resultats del prototip.

- **Capítol 9. Implementació Final.**
  En aquest capítol es descriuran les modificacions afegides respecte a la implementació del prototip inicial.

- **Capítol 10. Resultats Finals Obtinguts.**
  En aquest capítol es presentaran el resultats finals del producte obtingut.

- **Capítol 11. Millores i Possibles Línies de Futur.**
  En aquest apartat es comenten algunes possibles millores i ampliacions de l'aplicació

- **Capítol 12. Conclusions.**
  En aquest apartat s'exposen les conclusions que es poden extreure després de realitzar el treball del PFC.

2. Entorn de treball

En aquest apartat es descriu l'entorn de treball necessari per a la realització del PFC. S'ha realitzat un repàs al programari i les eines necessàries per a construir el Sistema d'Identificació d'Articles. Es detallen per cadascuna de les eines:

- Descripció de la funcionalitat i característiques
- Seguiment del procés d'instal·lació del programari i les possibles incidències que s'han anat trobant.

En aquest PFC s'ha apostat per l'ús de programari lliure, tant pel que fa a programari genèric (sempre que ha estat possible) com pel que fa a eines directament relacionades amb el món del Deep Learning.

2.1 Sistema Operatiu Os X

La única excepció pel que fa a ús de software lliure ha estat l'ús d'aplicacions amb llicència com el Sistema Operatiu d'Apple Os X i les eines d'ofimàtica (MS Word, MS Project) per a realitzar la documentació.

El pc amb el que s'ha realitzat el projecte és la màquina utilitzada per l'alumne tant pels estudis de la UOC com per les seves tasques personals, i aquest porta instal·lat el S.O. Os X. Per aquesta raó s'ha decidit mantenir-lo.

2.2 Python

Python és un llenguatge de programació que va ser creat al 1991 per Guido van Rossum, i actualment es desenvolupa com un projecte de codi obert, administrat per la Python Software Foundation

Es tracta d'un llenguatge de programació interpretat, d'alt nivell, multi plataforma i que compta amb gran quantitat de mòduls amb funcionalitats ja desenvolupades.

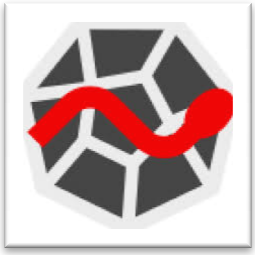
2.3 Anaconda

Anaconda és una distribució de Python ja preparada amb les principals llibreries especialitzades en temes matemàtics o científics. És gratuït i es considera software lliure.

Existeixen versions pels tres sistemes operatius més importants (Mac, Windows i Linux), i es tracta d'una distribució no invasiva, que es pot instal·lar amb diferents versions de Python sense problemes.

Compta amb un instal·lador de llibreries propi, conda, que permet instal·lar llibreries addicionals i amb un entorn gràfic. També amb un intèrpret de Python anomenat iPython, que conté característiques avançades a l'hora d'escriure els programes.

2.4 Spyder



Spyder és un entorn de desenvolupament interactiu (IDE) que s'inclou en la instal·lació d'Anaconda.

Va ser creat per Pierre Raybaut al 2009 i soporta diferents versions de Python, així com pot treballar en diferents sistemes operatius.

Els components bàsics d'Spyder són el seu editor, la terminal per executar els programes, l'explorador de variables i l'inspector d'objectes.

2.5 VLC Media Player

VLC Media Player és un reproductor multimèdia lliure i de codi obert, que funciona en la majoria de sistemes operatius (Windows, Linux, Mac OsX, Android, etc...) i que es capaç de reproduir la majoria d'arxius de vídeo.

Una de les seves funcionalitats menys conegudes és l'edició de vídeos, permetent extreure múltiples fotos a partir de gravacions de vídeo.

2.6 FileZilla

FileZilla és un client FTP lliure i de codi obert que permet la transferència de fitxers mitjançant els protocols FTP, SFTP i FTPS.

Va ser creat al 2001 per un grup d'estudiants de informàtica i tot i que inicialment va ser dissenyat per funcionar amb Windows, actualment ja funciona en la majoria de sistemes operatius i es considera multiplataforma.

2.7 Referències web de la descàrrega del software

A la següent taula s'indiquen les referències web des d'on s'ha descarregat el software necessari per al projecte que s'ha detallat als anterior apartats:

Aplicació	URL de descarrega del Software
Python	https://www.python.org/downloads/
Anaconda/Spyder	https://www.continuum.io/downloads
VLC Media Player	http://www.videolan.org/vlc/
FileZilla	https://filezilla-project.org

3. Estudi i Formació de la Tecnologia Actual

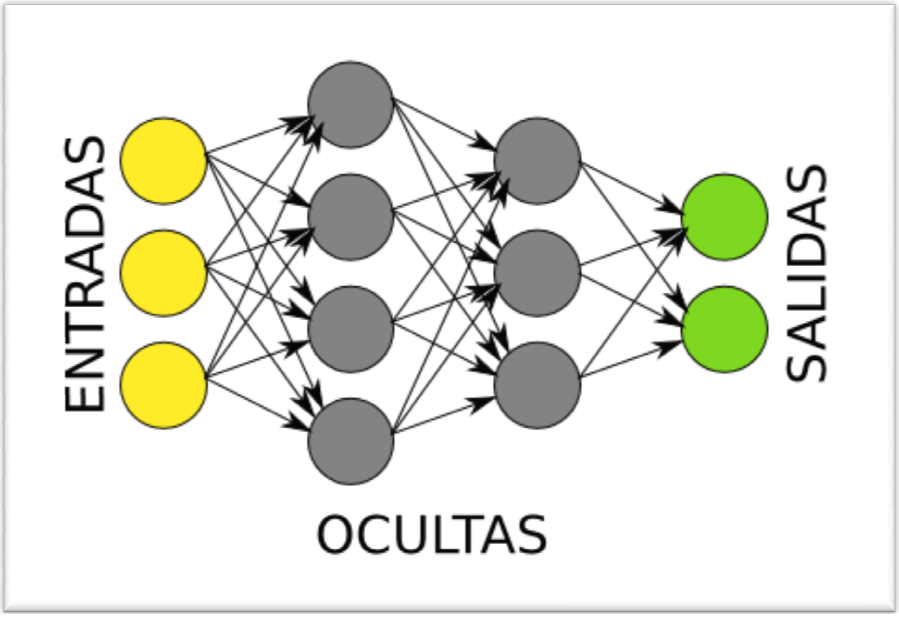
Per apropar-nos a la tecnologia TensorFlow cal conèixer primer els diferents algoritmes de Machine Learning. Els algoritmes de Machine Learning es poden classificar de diferents formes, però es podria resumir en:

- **Algoritmes d'aprenentatge supervisat:** estem donant a l'algoritme un conjunt de dades de que saben com ha de ser la sortida, y on hi ha una relació entre la entrada i la sortida. Normalment es tracta de problemes de "regressió" i "classificació".
- **Algoritmes d'aprenentatge no supervisat:** el procés es porta a terme tenint la informació del conjunt de dades formats només per l'entrada. Es busca l'estructura de les dades mitjançant l'agrupació (clustering) de les dades segons la relació entre les variables d'entrada.

Dintre del Deep Learning, podríem parlar d'una sèrie d'algoritmes que tenen com a propòsit simular el comportament del nostre cervell. Existeixen diferents models per implementar Deep Learning (arbres de decisió, algoritmes genètics, xarxes bayesianes, etc...), però basarem aquest projecte en les **xarxes neuronals convolucionals** o CNN, un tipus de xarxa neuronal que en el últims anys ha aconseguit resultats molt interessants en el reconeixement d'imatges.

3.1 Arquitectura d'una xarxa neuronal.

A la següent figura podem veure l'arquitectura genèrica d'una xarxa neuronal artificial:



Il·lustració 1 - Arquitectura d'una xarxa neuronal.

Cada cercle representa una neurona. Les neurones s'organitzen en capes de tal forma que la capa inferior (**input layer** o capa d'entrada) rep senyals de les entrades, i les neurones en la capa superior (**output layer** o capa de sortida) donen la resposta. Entre mig, la xarxa pot tenir varies capes intermèdies anomenades capes ocultes (**hidden layer**).

A les xarxes neuronals, les neurones d'una capa es comuniquen amb les que pertanyen a una capa superior per rebre informació, i aquestes a la seva vegada comuniquen la seva informació a les neurones de la capa inferior.

La principal operació que es realitza a les xarxes de neurones consisteix en multiplicar els valor de la neurona pels pesos de les seves connexions sortints. Cada neurona de la següent capa rep números de varies connexions entrants i el primer que fa és sumar-les totes.

Però a part d'aquesta operació, n'hi ha una altre operació que es realitza a totes les capes de la xarxa excepte a la d'entrada, es tracta de la funció d'activació. Aquesta funció rep com a entrada la suma de tots els números que arriben per les connexions entrants, transforma el valor mitjançant una formula i produeix un nou número. Existeixen diferents opcions, però una de les funcions més habituals és la funció sigmoide.

Una **xarxa neuronal convolucional o CNN** és un tipus particular de xarxa neuronal que funciona amb la mateixa arquitectura de les xarxes neuronals, amb la diferència que cada neurona d'una capa no rep connexions entrants de totes les neurones de la capa anterior, sinó només d'algunes. Això afavoreix que una neurona s'especialitzi en una regió concreta de l'entrada.

Dos dels conceptes més importants dintre d'una CNN són la **convolució** i el **pooling**. Si pensem en una xarxa on la capa d'entrada siguin imatges, es podria dir que:

- la **convolució** transforma la matriu de píxels que representa la imatge en una matriu de característiques. El propòsit principal d'una capa convolucional és detectar característiques en les imatges (arestes, línees, etc..)
- el **pooling** consisteix en agrupar característiques amb alguna funció d'agrupació. El propòsit principal és saber si una determinada característica es troba en qualsevol lloc de la imatge, sense donar importància a la ubicació exacta.

La capa d'entrada, o imput layer, estarà connectada amb una capa de neurones oculta (hidden), però només es connecten petites zones localitzades de l'espai. Per connectar cada neurona de la capa oculta amb el grup de neurones de la capa d'entrada s'usarà una matriu anomenada **kernel,** que vindrà definit per un valor de biaix i una matriu de pesos que permetrà detectar una característica concreta de la imatge a través del procés de convolució. Per tal de realitzar reconeixement d'imatges es habitual utilitzar diferents kernels a la vegada.

A les CNN's, aquesta capa convolucional normalment vindrà acompanyada d'una capa anomenada pooling, que simplifica i sintetitza la informació recollida per la capa convolucional. Hi ha diferents funcions per agrupar la informació, una de les més habituals és la coneguda com max-pooling, que aprèn amb el màxim de cada finestra. Dona invariància a la translació, rotació i escalat.

3.2 Aprenentatge d'una xarxa neuronal

Durant l'operativa d'una xarxa neuronal es poden distingir clarament dues fases: la fase **d'aprenentatge** (anomenada "**train**") i la fase **d'execució** (anomenada "**test**").

Un cop seleccionat el tipus de neurona artificial que s'utilitzarà en una xarxa neuronal i determinada la seva topologia es necessari entrenar-la per que la xarxa pugui ser utilitzada.

Durant la **fase d'aprenentatge**, la xarxa es entrenada per realitzar un determinat tipus de processament. Un cop aconseguit un nivell d'entrenament adequat, es passa a la fase d'execució, on la xarxa és utilitzada per portar a terme la tasca per la qual va ser entrenada.

L'aprenentatge d'una xarxa neuronal pot ser vist com el procés d'ajustar els paràmetres d'una xarxa. En un procés iteratiu, es va refinant la solució fins a aconseguir un nivell suficientment bo. La xarxa va adquirint la capacitat de generalitzar conceptes, de forma que, una xarxa neuronal pot tractar amb informació que no li va ser presentada durant la fase d'entrenament.

Un cop finalitzada la fase d'aprenentatge, la xarxa pot ser utilitzada per realitzar la tasca per la que va ser entrenada. Això es farà a la **fase d'execució**.

4. Estudi i formació dels projectes Deep Leaning existents

En aquest capítol es presentaran diferents treballs i exemples de projectes que s'han estudiat per tal d'entendre el funcionament intern de l'aprenentatge automàtic. Aquests exemples permeten veure tot el procés complet de creació, entrenament i avaluació d'un model de xarxa neuronal.

Tot i que es tracta d'una tecnologia recent, avui en dia es poden trobar a la xarxa nombrosos recursos. Ens centrarem en dos que han sigut especialment importants: **MNIST** y **CIFAR-10**. El primer, tracta sobre el reconeixement de dígits: es a dir, donada una imatge, obtenir les possibilitats de que sigui cadascú dels 10 possibles dígits (0-9). El segon, tracta sobre la classificació d'imatges: és a dir, donada una imatge, classificar-la dintre de les 10 categories existents. N'hi hagut més projectes, però aquests dos juntament amb la lectura del llibre del Jordi Torres han sigut el punt inicial d'aquest projecte.

Abans de entrar en detall dels diferents projectes estudiats es presentarà TensorFlow, una de les principals eines per desenvolupar aquest projecte.

4.1 TensorFlow.

TensorFlow és un paquet de codi obert que recentment a alliberat Google. Es tracta d'una llibreria de Machine Learning, que ens permet construir, entrenar i executar xarxes neuronals Deep Learning de forma àgil.

La informació a TensorFlow es guarda internament en una estructura en forma de **graf**, que conté tota la informació de les operacions i les dades. Això facilita conèixer les dependències entre operacions i permet assignar dispositius de manera **asíncrona i en paral·lel**, fet que permet aconseguir execucions molt ràpides. TensorFlow també consta **d'operacions complexes implementades** de forma eficient, com poden ser la multiplicació de matrius.

L'estructura de dades bàsica amb la que treballa TensorFlow s'anomena **Tensor**. Podria ser una espècies de array n-dimensional que té com a propietats un tipus estàtic de dades (booleà, string, float, etc...). El número de dimensions del Tensor s'anomena **rank**, i TensorFlow permet crear-ne de qualsevol dimensió, a la vegada que ofereix una sèrie d'operacions per permetre la seva manipulació.

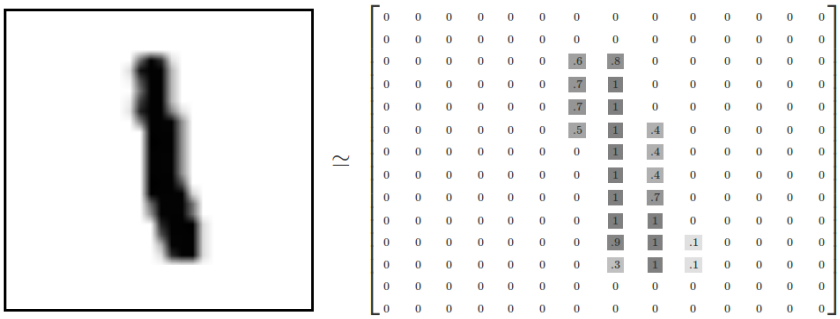
TensorFlow permet obtenir dades de diferent forma: des de fitxers de dades al començament del programa, treballant amb variables i constants, o utilitzant variables del propi codi de Python. TensorFlow s'utilitzarà sota Python que es tracta d'un llenguatge de programació interpretat, d'alt nivell, multi plataforma i que compta amb gran quantitat de mòduls amb funcionalitats ja desenvolupades. Entre aquest mòduls es troba el propi TensorFlow.

4.2 Classificació de dígits amb la base de dades de MNIST

El conjunt de dades de MNIST està format per imatges en blanc i negre de dígits fets a mà. Conté aproximadament 60.000 imatges per entrenar el model i 10.000 més per testejar-lo. Les dades es poden obtenir de la pàgina *The MNIST database*[7].

Sense entrar en el detall del **processament previ de les imatges**, cal dir que és una part important del projecte. Les imatges originals en blanc i negre han sigut normalitzades a 20x20 píxels, s'han aplicat tècniques anti-aliasing, i finalment s'han centrat en una de 28x28 píxels. El resultat són imatges semblants a la següent, on cada dígit inclou una etiqueta que indica quin dígit representa:

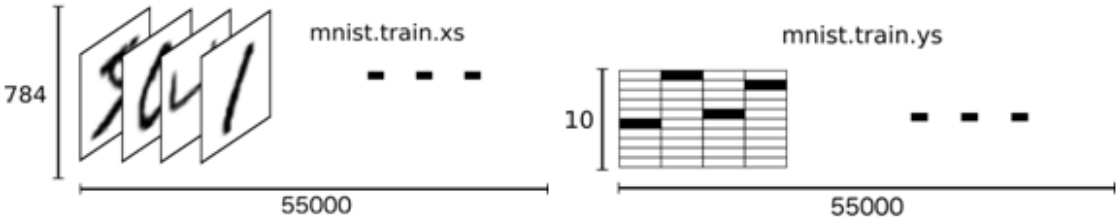
Per definir el **model**, partim de dos **conjunts de dades** (mnist.train i mnist.test), on cada element constarà de la imatge i la seva etiqueta. Cada imatge de **28x28** píxels, es pot emmagatzemar a una matriu de 28x28, on cada posició correspondrà a la intensitat del color amb un valor comprés entre un 0 i un 1. Cada etiqueta, es representarà en un vector de 10 posicions (una per cada possible dígit):



Il·lustració 2 - Representació de dígits en MNIST

En resum, tenim:

- Tensor Mnist.train.images de 2D: (55.000 imatges, 784 pixels)
- Tensor Mnist.train.labels de 2D: (55,000 imatges, 10 possibles valors)



Il·lustració 3 - Estructura de dades MNIST

A partir d'aquí, no es vol entrar en el detall dels dos models que ja estan explicats al llibre *Hello World en TesorFlow*[1], o a la web de MNIST[7] però si remarcar els fets destacats de cadascú dels processos més importants: definició del model, aprenentatge i test.

Com ja hem comentat anteriorment, la capa d'entrada estarà formada per un conjunt d'imatges de 28x28 píxels. A continuació, es connecten una sèrie de capes, entre elles les capes de convolució i pooling per acabar en una SoftMax.

Un fet a destacar dintre del model, que després es farà servir al model del SIA, es que per aconseguir classificar cada imatge d'entrada, intentarem obtenir les probabilitats de que sigui cadascú dels 10 possibles dígits. Per tant, tindrem un **vector de sortida de 10 probabilitats**, on cada posició correspondrà a la probabilitat de que sigui un dígit determinat. La suma de probabilitats serà 1. Això s'aconseguirà mitjançant una **capa de sortida amb una funció d'activació Softmax**. Primer es calculen les **evidències** de que una imatge correspongui a una determinada classe, i després es converteixen les evidències en **probabilitats**.

Un cop vist de forma simplificada el model, passem a detallar el fets destacats, de la capa **d'entrenament** i **d'avaluació** del model, que més endavant utilitzarem en el nostre model.

Per **entrenar** el nostre model, podem fer servir diferents algoritmes d'entrenament. Aquests algoritmes, s'encarreguen d'actualitzar el conjunt de paràmetres de la xarxa de forma iterativa per minimitzar la funció d'error. Es podrien utilitzar diferents algoritmes (Gradient Descent, Adam Optimizer, etc..) i de fet a la web oficial de MNIST ens indiquen els resultats obtinguts segons diferents algoritmes utilitzats.

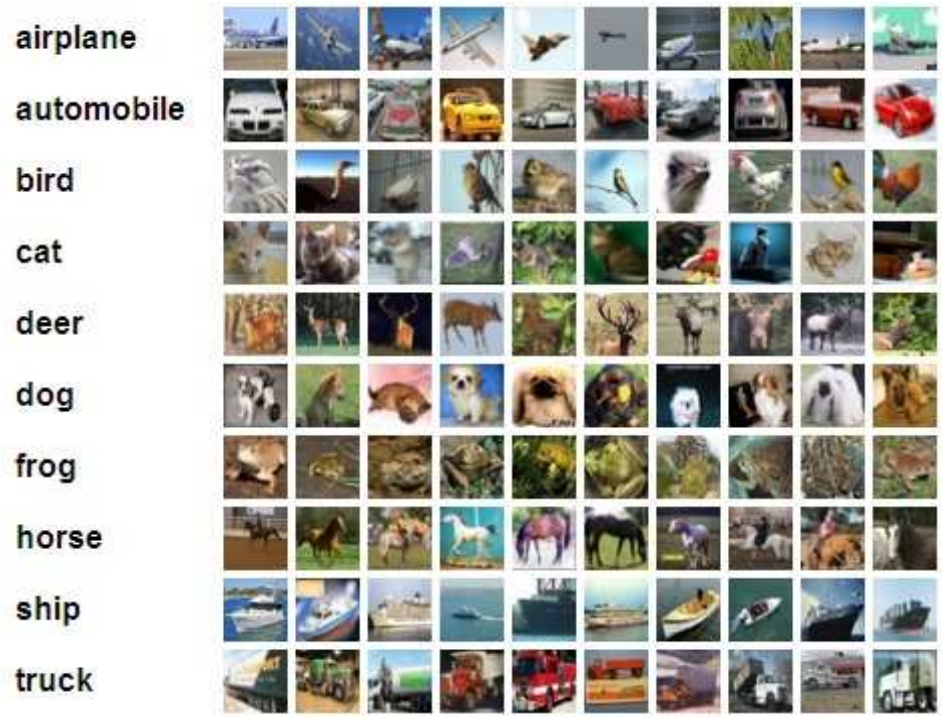
A la fase **d'avaluació**, per veure com es de bo el model, es calcula el percentatge d'encerts tractant de veure on es va predir l'etiqueta correcta. Això es fa amb la funció **argmax**, que retorna l'índex de l'entrada més alta del vector de probabilitats i el compara amb l'etiqueta correcta. D'aquesta forma s'aconsegueix saber si la predicció ha sigut correcta, i fent la mitja de totes les imatges, obtindre el percentatge d'encerts del model.

Els **resultats** obtinguts amb la implementació d'aquest model han sigut de prop d'un 97% d'encerts. Si ho comparem amb el 91% obtingut amb una xarxa neuronal bàsica, la millora és significativa.

4.3 Classificació d'imatges amb la base de dades de CIFAR10

El conjunt de dades de CIFAR10 està format per imatges en color RBG que es classifiquen en diferents categories. Conté aproximadament 60.000 imatges a color de 32x32 píxels que pertanyen a 10 classes exclusives de 6.000 imatges cadascuna. Les dades es reparteixen en 50.000 per entrenar o 10.000 per testejar. Les dades es poden obtenir de la pàgina *The CIFAR10 database*[8].

En aquest cas tampoc es detallarà el **processament previ de les imatges,** però cal destacar que es tracta d'imatges RGB i per tant el model de dades variarà. Cada imatge estarà representada per 32x32 píxels amb 3 frames que corresponen a cada un dels colors RGB.



Il·lustració 4 - Representació de imatges en CIFAR10

En aquest cas, no entrarem a detallar el procés d'entrenament, el de test i en l'estructura de la xarxa, ja que amb més complexitat que l'anterior, és semblant al del model del MNIST.

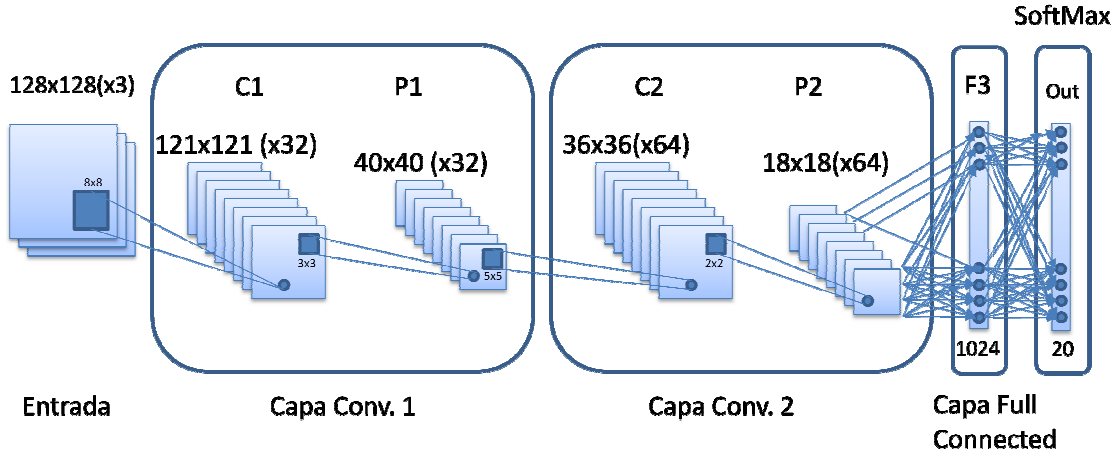
El que si que ha sigut important és analitzar la forma de tractar imatges en color, ja que l'anterior exemple tractava imatges en format de grisos. Pert tant, l'espai de píxels definit canvia d'una a tres finestres.

Cada fila del vector emmagatzema una imatge de 32x32. Les primeres 1204 entrades contenen els valors corresponents al canal de color vermell, les següents 1024 entrades al verd i les 1024 finals al blau. Les imatges s'emmagatzemen segons l'ordre de les files; és a dir, les primeres 32 entrades del vector fan referència als valors del píxels del canal vermell de la primera fila de la imatge.

5. Definició del Prototip Inicial

En aquest capítol es presentarà un prototip inicial del Sistema de Identificació d'Articles (SIA), presentant el model i definint la seva arquitectura. Així mateix es farà una primera aproximació a quines seran algunes de les seves característiques principals, tot i que aquestes poden anar evolucionant a mesura que s'entreni el model.

A continuació es mostra una representació simplificada del model triat, per després passar a detallar les característiques de cada una de les capes:



Il·lustració 5 - Representació Prototip Inicial SIA

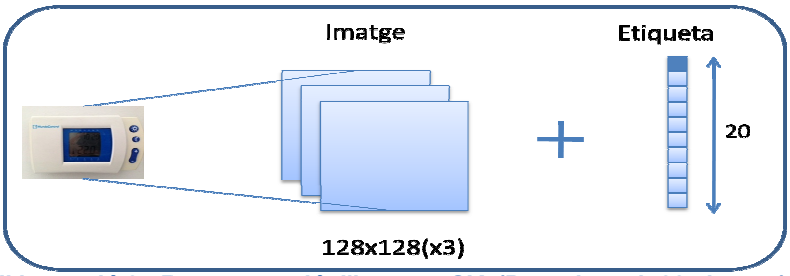
5.1 Capa d'entrada (Input Layer).

La primera capa del model, capa d'entrada o imput layer, estarà formada per dos conjunts de dades (train i training), on cada element constarà de la imatge i una etiqueta que definirà a quina classe pertany.

La imatges del SIA seran reduïdes a la mida de 128x128 píxels en color RGB. Cada imatge pertanyerà a una única classe de les 20 amb les que inicialment s'identificaran.

També s'han definit la mida dels conjunts de dades. Inicialment el conjunt de dades d'entrenament estarà format per 500 fotos de cada classe (10.000) i el conjunt de test estarà format per 50 fotos de cada classe (1000).

Cada imatge de **128x128** píxels, es pot emmagatzemar en tres matrius de 128x128, on cada matriu correspon a un dels colors RGB (vermell, verd i blau). Cada posició correspondrà a la intensitat de cadascú dels colors amb un valor comprés entre un 0 i un 1.



Il·lustració 6 - Representació d'imatges SIA (Prototip amb 20 classes)

Cada etiqueta, representarà el valor d'un dels 20 articles classificats, per poder tenir en un futur un vector de 20 posicions, on s'indiqui la correspondència entre el valor de l'etiqueta i la descripció de la classe:

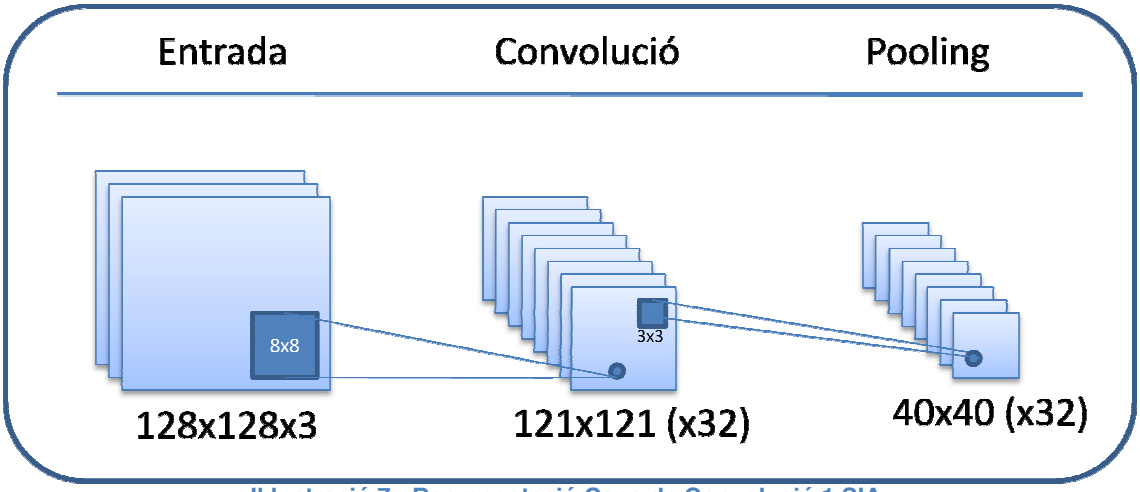
Etiqueta	Classe	Descripció
0	CO14555	CRONOTERMOSTATO DIGITAL MUNDOCONTROL HP-510
1	TF06001	VALVULA OBUS PARA 1/4" SOLDAR CON TAPON NORMAL
2	IM03008	TERMOMETRO CIRCULAR DIAM.57 CAPILAR 1 mt. 50/350 C.
...		
...		
...		
19		

5.2 Capa de convolució 1.

Com ja s'ha comentat anteriorment, el seu propòsit és detectar característiques a les imatges. D'això s'encarrega una primera capa de neurones ocultes (hidden) on cada neurona de la capa oculta serà connectada amb una petita regió de 8x8 neurones de la capa d'entrada:

La regió de 8x8 o finestra farà moviments d'1 píxel de distància (stride leght = 1), donant com a resultat un espai de 121x121 píxels. Es proposa utilitzar 32 Kernels per tal de detectar diferents característiques en la imatge.

A continuació es connectarà una capa de pooling que simplificarà la informació recollida per la capa convolucional. Farem servir una regió de 3x3, però en aquest cas no hi ha desplaçament sinó que es fa una divisió de l'espai, i s'aplica la tècnica de max-pooling per tal de condensar la informació, seleccionant el valor màxim de la regió de 3x3. Amb tot això, obtenim com a resultat un espai de 40x40 píxels.

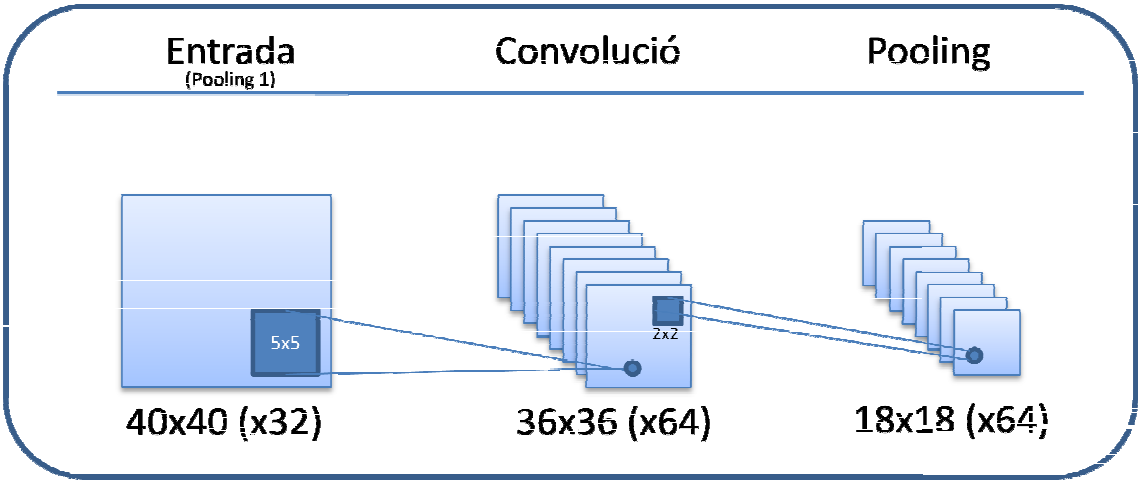


Il·lustració 7 - Representació Capa de Convolució 1 SIA

5.3 Capa de convolució 2.

A continuació de la primera capa de convolució, afegirem una altre per tal d'obtenir una xarxa més profunda. Aquesta segona capa tindrà 64 característiques o kernels i una finestra de 5x5 píxels que es desplaçarà també d'1 en 1. La mida de la capa de convolució resultant serà de 36x36x64, tenint en compte que ara la entrada és la capa resultant del primer procés de convolució.

Per la capa de pooling tornarem a fer servir el max-pooling, aquest cop amb una regió de 2x2. Obtindrem d'aquesta forma, una capa de 18x18 píxels amb 64 característiques.



Il·lustració 8 - Representació Capa de Convolució 2 SIA

5.4 Capa Full Connected i Capa SofMax

El següent pas ha sigut afegir una capa totalment connectada (Fully connected Layer), que servirà per alimentar una capa final de SoftMax.

S'usarà una capa amb 1024 neurones per tal de processar la imatge sencera, que rebrà com a entrada les 64 característiques de 18x18 resultants de la segona capa de convolució. Aquesta capa prendrà decisions a partir del que s'ha après a les convolucionals.

Abans de connectar aquesta capa amb la de Softmax, aplicarem el que es coneix per dropout, que consisteix en eliminar neurones de forma "aleatòria" per tal de millorar el rendiment predictiu. [1]

Per acabar, afegirem una capa SoftMax que ens permetrà codificar la solució final, obtenint-ne les probabilitats de cada imatge respecte a les 20 classes existents.

5.5 Resum de característiques

A continuació es mostra una taula amb el resum de les característiques de cada capa i alguna de les funcions d'activació que a priori s'utilitzaran:

Capa	Característiques	Resultat
Capa d'entrada	128x128x3	128x128x3
Capa Convolucional 1	Funció d'activació: Rectified linear unit (ReLu) Nº de característiques=64, Finestra=5x5	121x121x32
Capa Pooling 1	Max-pooling Pool Size = 3x3	40x40x32
Capa Convolucional 2	Funció d'activació: Rectified linear unit (ReLu) Nº de característiques=64, Finestra=5x5	36x36x64
Capa Pooling 2	Max-pooling Pool Size = 2x2	18x18x64
Fully Conected Layer	Dropout	1024
SoftMax		20

6. Obtenció i Tractament d'imatges

En aquest capítol es presentaran les diferents tècniques que s'han realitzat per tal d'aconseguir les imatges que formaran part del conjunt de dades d'entrenament i test.

6.1. Obtenció de les imatges

L'obtenció de les imatges s'ha realitzat al mateix centre logístic per reproduir unes condicions de l'entorn els més semblants possibles a la realitat.

A partir d'una càmera de 8Mpx d'un dispositiu mòbil (iPhone 6 Plus) s'han enregistrat petits vídeos de l'objecte en diferents posicions. D'aquests vídeos s'han extret fotos cada "x" segons, utilitzant una de les funcionalitats que ofereix VLC Media Player: "filtres per escena" o "Scene Filter".

Il·lustració 9 - Obtenció d'imatges

A partir dels vídeos, s'han obtingut 150 imatges. S'han seleccionat 50 pel conjunt de test, i s'han apartat per tal de que no hi hagués elements repetits. Les altres 100 imatges formaran part del conjunt d'entrenament, però han sigut tractades amb un script a Python per tal de crear més imatges del nostre model.

Per cada imatge, s'han creat 4 de noves aplicant tractaments de suavitzat d'imatges. Aquests tractaments es basen en aplicar diferents filtres a la imatge, i són útils per eliminar soroll i contingut d'alta freqüència[12].

S'han provat diferents tècniques de suavitzat d'imatges, però finalment s'ha optat per utilitzar les següents tècniques que proporciona la llibreria openCv: convolució 2D (amb un kernel de 5x5), blurring, medianBlur i BilateralFilter. En tots els casos s'obtenen diferents imatges que aconsegueixen eliminar sorolls, contorns, ombres, petites taquest o imperfeccions, etc...

Amb aquesta tècnica s'ha aconseguit crear un conjunt de dades de 500 elements per cada classe. A la següent figura es pot veure una representació de les imatges generades a partir d'una original:

Il·lustració 10 - Generació d'imatges Classe 0

6.2. Generació de les imatges del conjunt d'entrenament.

Com s'ha comentat anteriorment, a partir d'una imatge original s'han creat 4 de noves a partir de diferents tècniques de suavitzat d'imatges. A continuació es detalla el codi utilitzat:

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
import glob
import os
import tensorflow as tf

num_clases=20
im_id = 0
path = '/Users/Ana/PFC_Img/Test_Data/'

for j in range(num_clases):
   files = glob.glob(path+'Training/Class' + str(j) + '/'+'Class*.jpg')
   for filename in files:
        img = cv2.imread(filename,-1)
        img = cv2.resize(img, (128,128),interpolation=cv2.INTER_AREA)
        #preproceso imagen
        kernel = np.ones((5,5),np.float32)/25
        dst = cv2.filter2D(img,-1,kernel)
        blur = cv2.blur(img,(5,5))
        median = cv2.medianBlur(img,5)
```

```
bilat = cv2.bilateralFilter(img,9,75,75)
#Fin preproceso
position = j
im_id = im_id + 1
cv2.imwrite(path+'Training/Class' + str(j) + '/average'+ str(im_id) + '.jpg',dst)
cv2.imwrite(path+'Training/Class' + str(j) + '/blur'+ str(im_id) + '.jpg',blur)
cv2.imwrite(path+'Training/Class' + str(j) + '/med'+ str(im_id) + '.jpg',median)
cv2.imwrite(path+'Training/Class' + str(j) + '/bil'+ str(im_id) + '.jpg',bilat)
```

6.3. Definició de les classes inicials

A l'hora de fer la primera selecció d'articles, s'ha intentat triar una mostra que representés diferents famílies o classes d'articles. Tot i així, hi ha una certa limitació, tant en número de referències, com en referències que es diferencien d'altres per només mil·límetres (per exemple: vàlvula de !" o vàlvula de '").

Així mateix, els embalatges originals dels productes també han esdevingut un element de decisió. S'ha decidit treballar amb articles sense embalatge pensant en les possibles utilitats de l'aplicació.

Per tot això, inicialment s'ha triat una mostra amb la qual no tinguéssim limitacions per mesura, en la qual els articles fossin bastant diferents en forma o color per poder testejar el prototip; però al mateix temps, on hi hagués elements semblants per veure com responia el prototip amb aquestes condicions.

A continuació es mostra l'aspecte de les 20 classes inicials que s'han triat:

Imatge	Etiqueta	Classe	Descripció
	0	CO14555	CRONOTERMOSTATO DIGITAL MUNDOCONTROL HP-510
	1	GF02007	ENVASE BOTELLA 6KG. 47BAR
	2	XV01002	VOLANTE V. COMPUERTA HISPAVAL DN-40/50/65

Imatge	Etiqueta	Classe	Descripció
	3	VP01250	VALVULA ESCUADRA 7/8 CON MANETA
	4	VP01260	VALVULA MONTANTE DE 1" CON PALOMILLA
	5	HF07299	BOTELLA MULTIGAS 300
	6	AC03010	REDUCCION ACERO ZINCADO 1-1/4" a 3/4"
	7	AC71227	R580 COLECT.SIMPLE 3 CIRCUITOS PARA ADAPTADOR 3/4 x 16
	8	!"#$%%$&	'()"!!*+,& - /-&)*0-1& %2/ %. &
	9	AC71517	VALVULA ESFERA MH CROMADA CON PALOMILLA ROJA 3/4 x #

Imatge	Etiqueta	Classe	Descripció
	10	-0%.%#. &	03*!04(& & 5'+6(7*+,03&& -"34*6",!*+, &
	11	-0%89$9 &	DISCO PARA CORTE D.115 MM.
	12	:6#;$9% &	DESTORNILLADOR PUNTA ESTRELLA MODELO 0
	13	56#;### &	HACHA MANGO 320 mm. Y 800 gr.
	14	HF04026	TENAZA PICO LORO 240mm. C/CREM
	15	AS04525	TE 90º D.32 BLANCA
	16	MT11327	INTERRUPTOR MAGNETOTERMICO 2 POLOS

Imatge	Etiqueta	Classe	Descripció
	17	:6#<$;# &	MALETA UNIVERSAL DOBLE COMPLETA
	18	HF06133	BOTELLA 1 LT. ACEITE SINTETICO BOMBA VACIO
	19	WE04415	PALA VEP D.200 5 W MULTIANCLAJE

7. Implementació del Prototip Inicial.

En aquest capítol es descriurà com s'ha implementat el prototip inicial per presentar a continuació els primers resultats obtinguts.

7.1. Estructura de fitxers

A continuació s'explicarà com s'ha estructurat el codi i es detallaran els fets destacats en la implementació del prototip. El prototip del SIA consta de dos fitxers principals:

1. **SIA.py:** es descriu l'estructura bàsica de la nostre classe i es defineixen una sèrie de funcions que necessitarem a l'hora d'implementar la xarxa neuronal convolucional.
2. **SIA_train.py:** implementa la xarxa neuronal convolucional, amb la definició de cada una de les capes i els processos d'entrenament i test.

Dintre del primer es troben les rutines de tractament d'imatges, tant del conjunt d'imatges d'entrenament com del conjunt d'imatges de test. A continuació es farà una breu descripció de les funcions més importants:

- **Read_training_data (selft):** construeix el model de dades d'entrenament a partir de les imatges obtingudes. Per cada imatge es farà:
    1. processament previ de les imatges. A la capa d'entrada es detallarà en què ha consistit.
    2. re-dimensionament de la imatge a 128x128
    3. s'enregistrarà la imatge en una estructura de tupla, on la primera dimensió serà la etiqueta o label (que indentificarà la classe a la qual pertany la imatge) i la segona correspondrà a les dades de la imatge en forma de píxels.

- **Next_batch (selft, num):** rutina que retorna els següents X elements que es demanin (en funció del paràmetre "num").

- **Read_testing_data (self):** construeix el model de dades de test a partir de les imatges seleccionades per aquest conjunt. Per cada imatge es farà el mateix tractament que pel conjunt de dades:
    - processament previ de les imatges.
    - Re-dimensionament de la imatge a 128x128
    - s'enregistrarà la imatge amb la mateixa estructura definida anteriorment a les dades d'entrenament.

A continuació, es detallaran els fets destacats de cada capa del model, que es troben implementats al segon dels fitxers (SIA_train.py).

7.2. Capa d'entrada

A la primera capa del model, s'han pres dues decisions que varien lleugerament la proposta inicial:

1. Emmagatzemar els píxels de la imatge en un vector de: [1, 128*128*3]
2. Validar el prototip en dos fases:
1. A la primera, es reduirà a 10 el nombre de classes a identificar.
2. A la segona, un cop validat el funcionament del prototip amb 10 classes, s'augmentarà el número de classes de 10 a 20, tal i com s'havia definit inicialment.

En tots dos casos, es respectarà el definit inicialment, i el conjunt de dades d'entrenament estarà format per 500 fotos de cada classe (5.000/10.000) i el conjunt de test estarà format per 50 fotos de cada classe (500/1000).

La primera decisió ha estat pressa per tal de facilitar el codi obtingut. La segona, s'ha pres per tal millorar el rendiment del prototip ja que, al fer l'execució en una CPU, la capacitat de processament es veia notablement afectada. A la propera entrega es valorarà el fet de fer l'execució en una GPU i d'entrenar el model amb un número major d'iteracions.

Abans de començar a descriure les diferents capes del model, comentar que s'han fet 2 processaments previs de les imatges diferents abans d'enregistrar-les, per veure com afectava als resultats finals. Un dels objectius d'aquests tractaments han sigut solucionar les possibles deficiències d'il·luminació de l'entorn dels magatzems. Aquests tractaments han consistit en:

1. restar a la imatge la mitja dels valors dels píxels de tota la imatge
2. restar a cada canal de la imatge (RGB) la mitja de dels valors dels píxels del canal.

Opció 1	**Opció 2**

Els resultats obtinguts a l'hora de fer el procés d'entrenament són molt semblants en qualsevol dels casos (amb 10 classes o 20). Però, a l'hora de fer el procés de test s'han obtingut resultats lleugerament millors amb la opció 2.

Per aquest motiu, els resultats obtinguts que s'exposaran als següents apartats faran referència a la utilització de la segona de les opcions.

7.3. Capes de convolució, Capa Full Connected i Capa SofMax

A aquestes capes s'ha respectat el model definit inicialment per tal de veure quins resultats s'obtenien inicialment. Per tant, s'ha implementat:

1. Una primera capa de convolució, connectada a una regió de 8x8 amb 32 kernels. A continuació una capa de pooling amb una regió de 3x3. Amb això, obtenim com a resultat un espai de 40x40 píxels amb 32 característiques.

```
W_conv1 = weight_variable([8, 8, 3, 32])
b_conv1 = bias_variable([32])

h_conv1 = tf.nn.relu6(conv2d(x_image, W_conv1) + b_conv1)
h_pool1 = max_pool_3x3(h_conv1)
```

2. Una segona capa de convolució, amb 64 kernels i una finestra de 5x5. A continuació una capa de pooling amb una regió de 2x2. Amb això, obtenim com a resultat un espai de 18x18 píxels amb 64 característiques.

```
W_conv2 = weight_variable([5, 5, 32, 64])
b_conv2 = bias_variable([64])

h_conv2 = tf.nn.relu6(conv2d(h_pool1, W_conv2) + b_conv2)
h_pool2 = max_pool_2x2(h_conv2)
```

3. A continuació s'ha afegit una capa totalment connectada (Fully connected Layer), que servirà per alimentar una capa final de SoftMax. S'ha usat una capa amb 1024 neurones. Entre les dues capes, s'ha aplicat el dropout, per finalitzar amb la capa SoftMax que ens retornarà les probabilitats de cada imatge respecte les 10 classes existents.

```
W_fc1 = weight_variable([18 * 18 * 64, 1024])
b_fc1 = bias_variable([1024])

h_pool2_flat = tf.reshape(h_pool2, [-1, 18*18*64])
h_fc1 = tf.nn.relu6(tf.matmul(h_pool2_flat, W_fc1) + b_fc1)

keep_prob = tf.placeholder("float")
keep_prob = 1.0
h_fc1_drop = tf.nn.dropout(h_fc1, keep_prob)

# softmax
W_fc2 = weight_variable([1024, num_clases])
b_fc2 = bias_variable([num_clases])

# Use a name scope to organize nodes in the graph visualizer
with tf.name_scope('Wx_b'):
        y_conv=tf.nn.softmax(tf.matmul(h_fc1_drop, W_fc2) + b_fc2)

# More name scopes will clean up the graph representation
with tf.name_scope('xent'):
         cross_entropy = -tf.reduce_sum(y_*tf.log(y_conv))
         ce_summ = tf.scalar_summary('cross_entropy', cross_entropy)

with tf.name_scope('train'):
         train_step = tf.train.AdamOptimizer(1e-4).minimize(cross_entropy)

with tf.name_scope('test'):
        correct_prediction          =          tf.equal(tf.argmax(y_conv,1),
tf.argmax(y_,1))
        accuracy = tf.reduce_mean(tf.cast(correct_prediction, "float"))
        accuracy_summary = tf.scalar_summary('accuracy', accuracy)
```

Per últim, un altre fet destacat del codi es que es va prendre la decisió de dissenyar les convolucions i el max_pool de tal forma que no omplís amb zeros les imatges resultants. Això s'ha aconseguint definint les següents operacions amb el paràmetre de padding igual a VALID:

```
def weight_variable(shape):
  initial = tf.truncated_normal(shape, stddev=0.1)
  return tf.Variable(initial)

def conv2d(x, W):
  return tf.nn.conv2d(x, W, strides=[1, 1, 1, 1], padding='VALID'  )

def max_pool_3x3(x):
  return tf.nn.max_pool(x, ksize=[1, 3, 3, 1], strides=[1, 3, 3, 1], padding='VALID'  )

def max_pool_2x2(x):
  return tf.nn.max_pool(x, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='VALID'  )
```

7.4. Mètode d' Entrenament i Mètode de Test

A continuació de les capes definides, s'han implementat els mètodes d'entrenament i de test. A la propera entrega es valorarà el fet de fer fitxers separats per cadascú dels mètodes, però s'ha optat en fer-ho d'aquesta forma per tal de no complicar la implementació del prototip.

S'ha definit un bucle en el qual es poden definir el número de iteracions, per a continuació fer l'execució del codi:

```
for i in range(100):
      if i % 5 == 0:  # Record summary data, and the accuracy
        feed = {x: batch[0], y_: batch[1]}
        result = sess.run([merged, accuracy], feed_dict=feed)
        summary_str = result[0]
        acc = result[1]
        writer.add_summary(summary_str, i)

        result = sess.run([merged, cross_entropy], feed_dict=feed)
        summary_str = result[0]
        ce = result[1]
        writer.add_summary(summary_str, i)
        print("Accuracy at step %s: %s" % (i, acc) + " - Loss: %s" % (ce) )
     else:
        batch = train.next_batch(50)
        feed = {x: batch[0], y_: batch[1]}
        sess.run(train_step, feed_dict=feed)

        train.read_testing_data()
        print("Accuracy Final")
        print("test accuracy %g"% sess.run(accuracy, feed_dict={
              x: train.test_data, y_: train.test_labels}))
        print   "accuracy",   sess.run(accuracy,   feed_dict={x:   train.test_data,   y_:
train.test_labels})
        prediction=tf.argmax(y_conv,1)
        z = prediction.eval(feed_dict={x: train.test_data}, session=sess)
        for i in range(train.test_data.shape[0]):
```

```
            print ("CLASE: " + str(np.where(train.test_labels[i]==1)) + " - Prediccio:
" + str(z[i]))
```

S'ha fet seguiment de dues variables a l'entrenament: accuracy i loss. La primera mesura com entrena el model: a més iteracions, el valor hauria d'augmentar. La segona mesura si el sistema continua aprenent: a més iteracions, el valor hauria d'anar reduïnt-se.

Al mètode de test, es mostra l'accuracy del model de dades de test, per tal de validar quines prediccions s'estan realitzant.

8. Resultats Obtinguts.

A continuació es mostren els resultats obtinguts pel prototip dissenyat després de l'execució en dos entorns diferents:

3. **Entorn 1.** Inicialment s'ha provat el model amb un conjunt de 10 classes, en el qual el conjunt de dades d'entrenament estava format per 500 fotos de cada classe (5.000) i el conjunt de test estava format per 50 fotos de cada classe (500) imatges.
    1. **Processament Previ 1**: restar a cada canal de la imatge (RGB) la mitja de dels valors dels píxels del canal.
    2. **Processament Previ 2:** restar a la imatge la mitja dels valors dels píxels de tota la imatge

4. **Entorn 2:** Posteriorment, s'ha fet créixer el número de classes tractades, passant a ser 20. El conjunt de dades d'entrenament ha passat a constar de 10.000 imatges i el de test consta de 1000.

8.1. Joc de Proves 1. Execució amb 10 classes i processament previ 1.

Després de l'execució de 100 iteracions, s'ha obtingut les següents dades:

Iteració	Accuracy Train	Loss Train	Accuracy Test
0	0,14	1173,45	
10	0,54	131,332	
20	0,76	71,9856	
30	0,8	44,3223	
40	0,84	23,8748	
50	0,9	22,7595	
60	0,92	7,8721	
70	0,9	20,0977	
80	1.0	1,15261	
90	1.0	2,32309	0,728

En cada 10 iteracions, s'han anat enregistrant les dades de les variables d'entrenament. Al següent gràfic es pot visualitzar quina ha sigut la seva evolució de les variables accuracy i loss:

Il·lustració 11 - Evolució accuracy i loss (prototip 10 classes)

Al final de l'execució, també es mostra la predicció que es fa de cada imatge del conjunt de test. La següent figura mostra de forma esquematitzada quines han sigut les prediccions:

	Predicció											
Clase	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Total**	**%Encert**
0	37	3		2	4	1		1		2	50	0,74
1		32		7		1	7	1	2		50	0,64
2		7	39			1	3				50	0,78
3		5		23		2	14	5		1	50	**0,46**
4	2			1	44	1		1	1		50	0,88
5				1		38	11				50	0,76
6		1	2	2		2	43				50	0,86
7				1			5	41	3		50	0,82
8						3	16	2	28	1	50	0,56
9	1	1	1	3			3	1	1	39	50	0,78
Total	40	49	42	40	48	49	102	52	35	43	500	

8.2. Joc de Proves 2. Execució amb 10 classes i processament previ 2.

Després de l'execució de 100 iteracions, s'ha obtingut les següents dades:

Iteració	Accuracy Train	Loss Train	Accuracy Test
0	0,16	752,771	
10	0,66	92,8198	
20	0,72	40,6463	
30	0,9	20,7027	
40	0,92	20,5586	
50	0,98	2,67059	
60	0,88	17,4787	
70	0,98	5,91848	
80	0,98	6,9374	
90	0,98	6,21919	
100	0,98		0,766

En cada 10 iteracions, s'han anat enregistrant les dades de les variables d'entrenament. Al següent gràfic es pot visualitzar quina ha sigut la seva evolució de les variables accuracy i loss:

Il·lustració 12 - Evolució accuracy i loss (prototip 10 classes)

Al final de l'execució, també es mostra la predicció que es fa de cada imatge del conjunt de test. La següent figura mostra de forma esquematitzada quines han sigut les prediccions:

	Predicció											
Clase	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Total**	**%Encert**
0	44			3				2		1	50	0,88
1		38		4		1	5	1		1	50	0,76
2		5	38	2				3		2	50	0,76
3		15	1	20		3	3	5		3	50	**0,4**
4		2	1		45					2	50	0,9
5		4				46					50	0,92
6		6				6	32	2	4		50	0,64
7		5		3		1	6	34		1	50	0,68
8		3		2				2	43		50	0,86
9	1	1		1	1		1		1	44	50	0,88
Total	45	79	40	35	46	57	47	49	48	54	500	

Després d'executar diverses vegades els dos jocs de proves, es pot observar que els resultats no varien massa en funció dels tipus de processament previ triat. Per aquest motiu, la següent execució només es farà del primer processament triat: restar a cada canal de la imatge (RGB) la mitja de dels valors dels píxels del canal.

D'aquesta forma, augmentant en lo possible el número d'iteracions, es pretén veure com respon el model davant un augment de les classes a predir. A continuació es mostren els resultats del següent joc de proves.

8.3. Joc de Proves 3. Execució amb 20 classes.

Després de l'execució de 150 iteracions, s'ha obtingut les següents dades:

Iteració	Accuracy Train	Loss Train	Accuracy Test
0	0,08	947,604	
10	0,34	295,841	
20	0,36	183,117	
30	0,44	137,51	
40	0,5	107,409	
50	0,74	41,2624	
60	0,7	61,8629	
70	0,66	80,9862	
80	0,7	67,6483	
90	0,7	70,1412	
100	0,76	43,8512	
110	0,76	46,3757	
120	0,86	26,5283	
130	0,84	35,4311	
140	0,86	32,348	0,479

En cada 10 iteracions, s'han anat enregistrant les dades de les variables d'entrenament. A l'haver de parar l'execució a la iteració 150 per limitacions amb la CPU, no es pot observar quina seria la progressió del mètode d'entrenament amb més iteracions. Al següent gràfic es pot visualitzar que en cada iteració l'accuracy anava augmentant i el loss disminuint.

Il·lustració 13 - Evolució accuracy i loss (prototip 20 classes)

Al final de l'execució, també es mostra la predicció que es fa de cada imatge del conjunt de test. La següent figura mostra de forma esquematitzada quines han sigut les prediccions:

	Predicció																					
Clase	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**Total**	**%Encert**
0	31	1		2	7			1		2	3		1			1				1	50	0,62
1	3	11		4	4		1	1	1	1	3	2	2	2	2	1	5		5	2	50	0,22
2			29	1		1						1	1	5	2	1				9	50	0,58
3	2	2		15	2	1	7	1			9	1			3				6	1	50	0,3
4	8			1	26		1	5			4			1	3		1				50	0,52
5						31	9				4		1		1					4	50	0,62
6	1	3	1	1			13	2			13	2	1		3					10	50	0,26
7		2		4	4		2	20			8	1			1		3		1	4	50	0,4
8		1			1				12		8	1			2		19	1	3	2	50	0,24
9	10			2	5			1		23	3	3			1					2	50	0,46
10		1		1		1	3				22	1	4	1	7	4	2		1	2	50	0,44
11	4	3		1	1		1					29	1		1		6			3	50	0,58
12					2	1	1			3	2	5	26	1	2		2	1	1	3	50	0,52
13							3				1		2	35	1					8	50	0,7
14		1			2	2	19			1	6	1		2	9		3			4	50	0,18
15				1				1			8		1	2	1	24	4	1	4	3	50	0,48
16				1			4	1	1		6	2		2	1		27			5	50	0,54
17				1		2	1				1	1					4	27	1	12	50	0,54
18		1		1			4	1			8		1		2	2			29	1	50	0,58
19							2				2	4					2			40	50	0,8
Total	59	26	30	36	54	39	71	34	14	30	111	54	41	51	42	33	78	30	51	116	1000	0,479

9. Implementació Final

En aquest capítol es descriuran les modificacions realitzades respecte al prototip inicial, i les decisions que s'han pres per tal de millorar el prototip i obtenir una versió final del producte.

9.1. Estructura de fitxers.

Respecte a l'estructura del **codi**, s'ha mantingut la mateixa estructura del prototip. S'ha fet algun canvi per tal de fer més llegible el codi, però en termes generals es podria dir que no s'han realitzat canvis rellevants. Al final de la memòria, al capítol d'annexos es pot veure el codi final.

Els **fitxers** corresponents a les imatges del conjunt d'entrenament i test mantenen la mateixa estructura, però han sofert alguns canvis que es detallaran al següent apartat.

9.2. Capa d'entrada, de Convolució, Full Connected i SoftMax

Respecte als canvis realitzats en alguna de les capes del model, s'han pres diferents decisions enfocades principalment en l'assoliment de dos **objectius**:

1. **Millorar els resultats obtinguts** en el prototip.
2. **Provar el producte obtingut en un entorn més real**, per veure com responia el producte obtingut i quins resultats s'obtenien.

Per tal de **millorar els resultats obtinguts** en el prototip, inicialment s'ha treballat modificant diferents paràmetres del model i mantenint el conjunt de dades d'entrenament i test en 20 classes. D'aquesta forma es pretenia comparar els resultats obtinguts en funció dels valors dels diferents paràmetres, mantenint el mateix joc de dades.

Inicialment es va pensar en modificar dos aspectes del prototip:

1. **La probabilitat del dropout.** Al prototip, la probabilitat del dropout es mantenia sempre amb el valor de 1. Això volia dir que la probabilitat d'agafar una connexió era sempre 1; per tant, les agafava totes. S'han fet diferents proves modificant el paràmetre amb valors compresos entre 1 i 0,8.
2. **El nombre d'iteracions.** A l'executar el prototip en una CPU, el màxim nombre d'iteracions que es podien executar sense problemes de memòria es trobava entorn de 150. Això suposava una limitació a l'hora de continuar entrenant el model, ja que es necessitaven més iteracions per veure com responia el model.

Com que els resultats obtinguts eren bons amb un baix número d'iteracions, es va decidir no ampliar el número d'imatges de cada classe, **mantenint les 500 imatges per classe definides inicialment** i executar el prototip en un servidor més potent, on el ús d'una **GPU** ajudaria a fer més iteracions i entrenar millor.

Aquí van arribar els primers problemes, ja que TensorFlow emmagatzema les dades en una estructura interna en forma de graf on a cada iteració del procés d'entrenament va afegint la informació necessària als nodes. Al fer-ho d'aquesta forma, a cada iteració el consum de memòria s'anava incrementant de forma

excessiva, fins a arribar al punt de que el Tensor que s'havia de tractar era massa gran per fer-ho, donant una excepció per problemes de memòria.

Arribat a aquest punt, es va prendre la decisió de provar el model **reduint la mida de les imatges de la capa d'entrada**. D'aquesta forma, es pretenia que es pogués utilitzar menys memòria a l'hora de tractar les imatges i el servidor permetés fer més iteracions del model. Això implicava modificar tota la mida del model dissenyat inicialment, de tal forma que el model resultat es mostra a la següent figura:

Il·lustració 14 - Representació Prototip SIA amb imatges de 64x64p

D'aquesta forma, es va tornar a fer probes del model mantenint el nombre d'iteracions i les 20 classes inicials amb les 500 imatges de cada classe. Es volia comparar els dos models (imatges de 64x64 i 128x128) en unes condicions els més semblant possibles. Després de diverses proves, els resultats van ser satisfactoris, perquè amb les mateixes iteracions i mantenint el valor del dropout a 1, el resultat de test va ser millor. A continuació es mostra els resultats obtinguts en una de les diferents execucions realitzades:

Classes	**20**		**20**	
Nº Iteracions	**150**		**150**	
Dropout (Keep_prob)	**1**		**1**	
Capa d'Entrada	**128x128x3**		**64x64x3**	
Iteració	**Acc Train**	**Loss Train**	**Acc Train**	**Loss Train**
0	0,08	947,604	0,08	810,324
10	0,34	295,841	0,28	393,172
20	0,36	183,117	0,3	250,911
30	0,44	137,51	0,42	179,578
40	0,5	107,409	0,48	120,45
50	0,74	41,2624	0,52	112,196
60	0,7	61,8629	0,6	78,1907
70	0,66	80,9862	0,68	64,9456
80	0,7	67,6483	0,66	54,263
90	0,7	70,1412	0,82	38,2635
100	0,76	43,8512	0,66	61,2364
110	0,76	46,3757	0,74	40,066
120	0,86	26,5283	0,88	16,7317
130	0,84	35,4311	0,84	30,1004
140	0,86	32,348	0,84	26,383
Accuracy Test	0,479		0,557	

Partint d'aquest nou model que minimitzava el problema de l'ús de la memòria, es volia iniciar de nou amb el procés de modificar els paràmetres del **dropout** i el **nombre d'iteracions** del model d'entrenament. Però en aquest moment es va pensar que el fet de reduir les imatges podia provocar que el pooling de 3x3 que s'aplicava a la primera capa no funcionés del tot bé ja que les imatges d'entrada eren més petites i conseqüència d'això s'obtenien imatges molt petites (7x7). Es va provar a modificar de nou el model fent un primer **pooling de 2x2**, de tal forma que el model resultat es mostra a la següent figura:

Il·lustració 15 - Representació Prototip SIA amb imatges de 64x64p (pooling 2x2)

D'aquesta forma, es va tornar a fer probes del model mantenint les mateixes condicions (iteracions i classes). Després de diverses proves, els resultats van ser satisfactoris, perquè amb les mateixes iteracions i mantenint el valor del dropout a 1, el resultat de test tornava a ser millor. A continuació es mostra els resultats obtinguts en una de les diferents execucions realitzades:

Classes	**20**		**20**	
Nº Iteracions	**150**		**150**	
Dropout (Keep_prob)	**1**		**1**	
Capa d'Entrada	**64x64x3**		**64x64x3**	
Pooling	**3x2 – 2x2**		**2x2 – 2x2**	
Iteració	**Acc Train**	**Loss Train**	**Acc Train**	**Loss Train**
0	0,08	810,324	0,06	946,414
10	0,28	393,172	0,34	237,992
20	0,3	250,911	0,4	207,542
30	0,42	179,578	0,28	198,557
40	0,48	120,45	0,8	64,9272
50	0,52	112,196	0,66	77,9491
60	0,6	78,1907	0,6	72,4206
70	0,68	64,9456	0,68	60,4595
80	0,66	54,263	0,7	69,9282
90	0,82	38,2635	0,7	63,0092
100	0,66	61,2364	0,78	67,014
110	0,74	40,066	0,66	70,7901
120	0,88	16,7317	0,82	32,7403
130	0,84	30,1004	0,76	45,81315
140	0,84	26,383	0,86	27,2421
Accuracy Test	0,557		0,608	

Deixant per definitiu aquest nou model, es va passar a executar el mateix model augmentant el **nombre d'iteracions** fins al màxim possible, que eren 220. Per això es va utilitzar la GPU del servidor de la universitat (lab.sunai.uoc.edu), que es va haver de configurar abans per l'ús del TensorFlow i Python.

L'execució amb 220 iteracions van acabar amb uns resultats de test de pràcticament un **70% d'encerts** (accuracy = 0,675). A partir d'aquí es van anar fer proves modificant el paràmetre de dropout i mantenint constant el nombre d'iteracions. Es va provar amb diferents valors i aquests són els resultats:

Dropout	Iteracions	Test Accuracy
1	220	0,675
0,9	220	0,579
0,8	220	0,537

Els resultats mostraven que feien falta més iteracions per tal de obtenir uns percentatges d'encert alt. Com ja hem dit abans, quan el paràmetre del dropout era més petit al model li costava més entrenar i amb les mateixes iteracions donava pitjors resultats. Per aquest motiu, i davant la limitació en quant a iteracions possibles, es va prendre la decisió de fixar el paràmetre del **Dropout a 1;** és a dir, no fer-ne. D'aquesta forma, tot i que el model estaria preparat per aplicar dropout, les limitacions de memòria semblaven indicar que la millor opció era no fer-ne ús.

En aquest punt, el model es podia haver donat per definitiu, però es va pensar en modificar el número d'elements que es processava en cada iteració (**batches**). Inicialment era 50, però es va provar amb conjunts més petits per poder fer més iteracions i veure com afectava això al resultat final. Es va provar amb diferents valors i aquests són els resultats de les diferents combinacions testejades:

Batches	Dropout	Max. Iteracions	Test Accuracy
50	1	220	0,675
40	1	260	0,696
25	1	320	0,621
10	1	410	0,546

Als resultats obtinguts no s'aprecien grans diferències, però si que es pot apreciar que amb conjunt d'elements molt petits la precisió es veu reduïda. Així que s'optarà per modificar el conjunt **d'elements a 40** ja que s'han obtingut resultats lleugerament millors, però es tindrà en compte que potser en un futur, quan el prototip s'executi amb més elements i es puguin fer més iteracions, seria recomanable tornar a deixar aquest paràmetre en 50.

Al fitxer **JocsProvaPrototip.xlsx** que forma part d'aquest projecte es mostra un resum dels resultats obtinguts en els diferents jocs de prova.

Finalitzaven en aquest punt les modificacions del model del prototip inicial. Aquestes modificacions han permès passar d'uns resultats d'avaluació de 0,479 a 0,696.

Cal remarcar també que al prototip es van triar diferents **processaments previs**. Finalment, s'ha decidit escollir la opció que donava millors resultats per al model final, que ha sigut el tractament que resta a cada canal de la imatge (RGB) la mitja dels valors dels píxels dels canals.

Per tal de fer més real el prototip inicial i veure com responia el model es va prendre la decisió d'ampliar el número de classes: es van afegir 10 classes més, per tenir un total de 30. Les noves 10 classes han tingut la particularitat de ser objectes molt semblants entre ells o bé entre altres objectes que ja existien al model inicial de 20 classes. Alguns articles es diferenciaven només pel color (classes 23, 26 i 27), altres per petites diferències (classe 0 i 29), etc...

A continuació es mostren les classes afegides:

Imatge	Etiqueta	Classe	Descripció
	20	AS01001	JUEGO SOPORTES PEQUEÑO
	21	AV02022	ANTIVIBRADOR METALICO M-50
	22	VD10101	REJA PROTECCION G-80 METALICA
	23	AI09000	ROLLO PLASTIESC 50 BLANCO
	24	AI19002	TUBO SILICONA TRASLUCIDO FUNGICIDA
	25	MA11267	IRINA GEL HIGIENIZANTE 100 ML

Imatge	Etiqueta	Classe	Descripció
	26	AI09002	ROLLO PLASTIESC 50 GRIS
	27	AI09003	ROLLO PLASTIESC 50 NEGRO
	28	AI09015	ROLLO CINTA POLIPROPILENO POLIESC 50 PLATA
	29	CO14555	CRONOTERMOSTATO DIGITAL MUNDOCONTROL HP-510

10. Resultats Finals Obtinguts

10.1. Joc de Proves 1. Execució amb 20 classes en GPU i 270 iteracions (dropout = 1).

Després de l'execució de 270 iteracions, s'ha obtingut les següents dades:

Iteració	Accuracy Train	Loss Train	Accuracy Test
0	0	892,132	
10	0,25	262,147	
20	0,525	77,5721	
30	0,5	103,517	
40	0,45	88,1551	
50	0,575	50,5613	
60	0,55	88,5784	
70	0,65	48,7417	
80	0,65	52,0142	
90	0,7	45,6567	
100	0,625	77,3268	
110	0,8	26,7698	
120	0,775	31,0943	
130	0,775	27,1388	
140	0,65	45,1161	
150	0,775	43,7175	
160	0,775	28,7791	
170	0,725	32,785	
180	0,775	45,3794	
190	0,825	15,3351	
200	0,9	9,03842	
210	0,9	19,7784	
220	0,875	13,2299	
230	0,875	13,8172	
240	0,925	8,53471	
250	0,925	8,74042	
260	0,875	18,9671	
			0,696

Al final de l'execució, també es mostra la predicció que es fa de cada imatge del conjunt de test. A continuació es mostra de forma agrupada quines han sigut les prediccions:

	%Encert
Classe 0	0,62
Classe 1	0,7
Classe 2	0,74
Classe 3	0,42
Classe 4	0,78
Classe 5	0,84
Classe 6	0,28
Classe 7	0,62
Classe 8	0,82
Classe 9	0,84
Classe 10	0,44

	%Encert
Classe 11	0,6
Classe 12	0,68
Classe 13	0,74
Classe 14	0,88
Classe 15	0,58
Classe 16	0,86
Classe 17	0,7
Classe 18	0,8
Classe 19	0,94

En cada 10 iteracions, s'han anat enregistrant les dades de les variables d'entrenament. Al següent gràfic es pot visualitzar quina ha sigut la seva evolució de les variables accuracy i loss:

Il·lustració 16 - Evolució accuracy i loss (Model final 20 classes)

10.2. Joc de Proves 2. Execució amb 30 classes en GPU i 270 iteracions (dropout = 1).

Després de l'execució de 270 iteracions, s'ha obtingut les següents dades:

Iteració	Accuracy Train	Loss Train	Accuracy Test
0	0,025	883,688	
10	0,2	310,166	
20	0,275	211,724	
30	0,3	200,934	
40	0,275	262,397	
50	0,375	148,979	
60	0,3	114,576	
70	0,5	112,862	
80	0,525	106,655	
90	0,5	111,09	
100	0,575	86,9211	
110	0,5	96,293	
120	0,625	67,3887	
130	0,5	92,5572	
140	0,6	61,6938	
150	0,525	74,2224	
160	0,575	51.2073	
170	0,7	41,9654	
180	0,7	36,7862	
190	0,75	33,2349	
200	0,775	50,9265	
210	0,775	38,0653	
220	0,725	42,6339	
230	0,8	42,1822	
240	0,75	57,2308	
250	0,725	49,5304	
260	0,75	41,1844	
			0,573333

Al final de l'execució, de forma opcional, també es mostra la predicció que es fa de cada imatge del conjunt de test. A continuació es mostra de forma agrupada quines han sigut les prediccions:

	%Encert (20)	%Encert (30)
Classe 0	0,62	0,56
Classe 1	0,7	0,38
Classe 2	0,74	0,26
Classe 3	0,42	0,36
Classe 4	0,78	0,66
Classe 5	0,84	0,88
Classe 6	0,28	0,38
Classe 7	0,62	0,46
Classe 8	0,82	0,76
Classe 9	0,84	0,54
Classe 10	0,44	0,3
Classe 11	0,6	0,68
Classe 12	0,68	0,52

	%Encert (20)	%Encert (30)
Classe 13	0,74	0,8
Classe 14	0,88	0,74
Classe 15	0,58	0,36
Classe 16	0,86	0,9
Classe 17	0,7	0,9
Classe 18	0,8	0,5
Classe 19	0,94	0,54
Classe 20		0,88
Classe 21	-	0,72
Classe 22	-	0,64
Classe 23	-	0,7
Classe 24	-	0,36
Classe 25	-	0,76
Classe 26	-	0,66
Classe 27	-	0,74
Classe 28	-	0,46
Classe 29	-	0,46

En cada 10 iteracions, s'han anat enregistrant les dades de les variables d'entrenament. Al següent gràfic es pot visualitzar quina ha sigut la seva evolució de les variables accuracy i loss:

Il·lustració 17 - Evolució accuracy i loss (Model final 30 classes)

11. Conclusions

En aquest apartat s'exposen les conclusions que es poden extreure després de realitzar el treball del PFC. A continuació les enumerem:

- S'ha aconseguit assolir un bon coneixement dels conceptes bàsics del món del Deep Learning, els diferents tipus d'algoritmes i sistemes d'aprenentatge.

- S'ha aconseguit entendre el funcionament d'una CNN, la seva estructura i disseny.

- S'ha aprofundit en conceptes de programació de xarxes neuronals convolucionals sota Python i la llibreria TensorFlow de Google. Tot i fer molts anys que l'alumne no desenvolupava aplicacions i desconeixia el llenguatge de programació Python, ha estat possible construir una aplicació que permetés identificar articles (SIA) mitjançant l'ús d'aplicacions i eines totalment noves per l'alumne.

- S'ha aconseguit minimitzar els problemes amb la gestió de la memòria del servidor i les limitacions a 270 iteracions al mètode d'entrenament. De fet, aquest problema, ha esdevingut una oportunitat per conèixer amb més detall tots el paràmetres que es podien arribar a modificar, i quin era el comportament del model en cada cas.

- S'ha descobert un món totalment desconegut per l'alumne, i que tot i ser una temàtica relativament nova, hi havia molta informació disponible a Internet en forma de documentació, projectes, etc....

- Les mesures per a mitigar els riscos o incidències durant el projecte han resultat amb èxit. L'alumne va tenir problemes per motius personals que afectaven al rendiment diari, però el problema es va solucionar treballant major nombre d'hores durant els caps de setmana.

- El resultat final obtingut ha resultat molt gratificant, tot i haver necessitat de moltes hores d'esforç per a realitzar-lo. S'han complert tots els objectius establerts i, tot i patir diferents entrebancs durant el projecte, s'han pogut complir totes les fites planificades, i s'ha pogut seguir el calendari de treball definit a l'inici del projecte.

- Finalment, cal dir que, tot i que no era un dels objectius del projecte, l'aplicació resultant pot esdevenir el punt d'inici d'un projecte per poder posar-la en funcionament en un futur en un entorn real.

12. Millores i possibles línies de futur

Tot i que l'aplicació implementada cobreix tots els objectius que s'havien plantejat, sempre hi ha marge de millora i d'evolució. En aquest apartat es comenten algunes possibles millores i ampliacions del Sistema d'Identificació d'Articles que no han pogut ser implementades per falta de temps, però que seria possible afegir en un futur per tal de millorar-la en diferents aspectes.

12.1 Obtenció i Tractament d'imatges

Pel que fa a la obtenció i tractament d'imatges hi ha diferents punts de millora, alguns més costosos que altres, però que afegirien més realisme al resultat final obtingut:

- **Ampliació del conjunt de classes**: per fer d'aquest sistema d'identificació d'articles un producte que pugui ser utilitzat en un entorn real, faria falta un nombre més alt de classes i veure com es comporta el model proposat.
- **Obtenció d'imatges en diferents entorns**: les imatges que formen part d'aquest model s'han enregistrat en condicions reals, però sempre buscant les millors condicions d'il·luminació o de l'entorn. Caldria provar-ho amb imatges menys nítides o en pitjors condicions per veure els resultats finals.
- **Processament previ més complet**: el processament previ que s'ha fet a les imatges ha sigut molt simple. Existeixen projectes i documentació sobre tractaments de dades molt més complexos, que queden fóra de l'abast d'aquest projecte, però que en un futur es podrien incloure per obtenir millors resultats finals..

12.2 Mètode d'entrenament i Test

Pel que fa al mètode d'entrenament, degut a la interpretació en forma de graf de l'estructura de dades que fa TensorFlow, s'han tingut diferents problemes a l'hora d'executar el codi, ja que només ha sigut possible executar un màxim de 270 iteracions per problemes de ús de la memòria del servidor amb GPU. Es per això que un punt de millora necessari a resoldre en el futur seria el de:

- **Ampliació del nombre d'iteracions en el mètode d'entrenament:** degut al problema amb l'ús de la memòria del servidor, s'ha optimitzat al màxim el model proposat, però seria interessant resoldre el problema computacional per tal de poder entrenar més el model i veure com milloren els resultats amb un nombre major d'iteracions.

11.3 Altres possibles millores

A continuació enumerem algunes altres possibles millores addicionals que es podrien realitzar:

- **Disseny de la interfície gràfica amb l'usuari:** si es vol utilitzar aquest sistema per la identificació d'articles d'un centre logístic, s'hauria de treballar en la construcció d'una interfície d'usuari que permetés a qualsevol persona amb un dispositiu mòbil capturar una imatge i fer-ne servir la utilitat del SIA.

13. Glossari

A continuació es descriuen algun del termes i acrònims més rellevants utilitzats dins la memòria:

- SIA – Sistema d'Identificació d'Articles
- CNN – Convolutional Neural Network o Xarxa Neuronal Convolucional
- GPU – Graphics Processor Unit o Unitat de processament gràfic
- Codis RFID – Radio Frecuency Identification o Identificació per Radio Frecuència.
- Codis QR – Quick Response o Codi de Resposta Ràpida.

14. Bibliografia

14.1. Publicacions.

!"#$%&'()$*&"+,-$./0$1$230$.4'5+6&74$*+58-$.74-8&";<$=>7)5)45)?$4$64$@'&A'4B45)?$+7$ C++@$D+4'7)7AE$
[En línea] http://www.jorditorres.org/libro-hello-world-en-tensorflow/
[Data de consulta: Febrer - Març de 2016]

!9#$F'G)56+<$=064,,)H)545)?$(I)B4GA+,$4BJ$K74$0LLE$
[En línea] http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
[Data de consulta: Febrer de 2016]

!M#$3KJ6)545)?<$=N+($L+K'&746$0&7O&6K5)&746$@+'$64$@4'4$'+5&7&5)B)+7G&$(+$ (QA)G&,E$
[En línea] http://digital.csic.es/bitstream/10261/84753/1/RED%20NEURONAL.pdf
[Data de consulta: Març de 2016]

!R#$$3KJ6)545)?<$ Classifying Plankton with deep neural networks.
[En línea] http://benanne.github.io/2015/03/17/plankton.html
[Data de consulta: Març de 2016]

!S#$3KJ6)545)?<$ "Facial Keypoints Detection"
[En línea] https://www.kaggle.com/c/facial-keypoints-detection/details/getting-started-with-r
[En línea] http://danielnouri.org/notes/2014/12/17/using-convolutional-neural-nets-to-detect-facial-keypoints-tutorial/
[Data de consulta: Març de 2016]

14.2. Enllaços electrònics

!M#$ .6&A$ @+',&746<$ =0&B$ (+H)7)'$ +6,$ @4'T&$ 4+,$ 4$ K'&746$ 5&7O&6K5)&746$ @+'$ 64$ 564,,)H)545)?$(I)B4GA+,VE$$
[En línea] http://qiku.es/pregunta/103416/como-decidir-los-parametros-de-una-red-neuronal-convolucional-para-la-clasificacion-de-imagenes-how-do-you-decide-the-parameters-of-a-convolutional-neural-network-for-image-classification
[Data de consulta: Març de 2016]

!R#$*+7,&'W6&X<$ Documentació sobre TensorFlow.
[En línea] !""#$%&&'")*$+,-.+'(+,/&
[Data de consulta: Febrer – Març de 2016]

!S#$*KG&')46,$,&J'+$*+7,&'W6&X<$
[En línea] !""#$%&&'")*$+,-.+'(+,/&0),$1+*$&,2(3&"4"+,15.$&6*1$" &#,+$&1*7)8(!"6.
[En línea] !""#$%&&'")*$+,-.+'(+,/&0),$1+*$&,2(3&"4"+,15.$&7))#9 :**&1*7)8(!"6.
[Data de consulta: Març de 2016]

!;#$3YG8&7<$ Documentació sobre Python
[En línea] https://www.python.org/
[Data de consulta: Febrer – Març de 2016]

!Z#$[L>/*<Web oficial Mnist
[En línea] http://yann.lecun.com/exdb/mnist/
[Data de consulta: Març de 2016]
$
!\#$0>WFN"W¥eb CIFAR10
[En línea] https://www.cs.toronto.edu/~kriz/cifar.html
[Data de consulta: Març de 2016]

!":#$F745&7(4$et Up d'Anaconda
[En línea] https://docs.continuum.io/anaconda/ide_integration
[Data de consulta: Març de 2016]
$
!""#$*KG&')46$,&J'+$0&7O&6KG)&746$L+K'46$L+GX&'],<$
[En línea] http://deeplearning.net/tutorial/lenet.html
[Data de consulta: Març de 2016]

!"9#$*'45G4B+7G$(I)B4GA+,$4BJ$^@+70O$$
[En línea] http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_table_of_contents_imgproc/py_table_of_contents_imgproc.html#py-table-of-content-imgproc
[Data de consulta: Abril- Maig de 2016]

!"M#$_`+,G)&7,$(+$*+7,&'W$6&X
[En línea] https://github.com(tensorflow/tensorflow/issues
[Data de consulta: Maig de 2016]

15. Annexos

A continuació es detallen els diferents documents que són massa extensos per incloure dins la memòria,

15.1 Annex 1. SIA_Train.py

```
# -*- coding: utf-8 -*-
#Fitxer per entrenar el model SIA.

import SIA as dat
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
import os


def weight_variable(shape):
  initial = tf.truncated_normal(shape, stddev=0.1)
  return tf.Variable(initial)

def bias_variable(shape):
  initial = tf.constant(0.1, shape=shape)
  return tf.Variable(initial)

def conv2d(x, W):
  return tf.nn.conv2d(x, W, strides=[1, 1, 1, 1], padding='VALID')

def max_pool_3x3(x):
  return tf.nn.max_pool(x, ksize=[1, 3, 3, 1], strides=[1, 3, 3, 1],
padding='VALID')

def max_pool_2x2(x):
  return tf.nn.max_pool(x, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1],
padding='VALID')


def main(_):
  num_clases = 30
  # Import data
  train = dat.SIA()
  train.read_training_data()
  batch = train.next_batch(40)

  #sess = tf.InteractiveSession()

  with tf.Graph().as_default():
    # Create the model
    x = tf.placeholder(tf.float32, shape=[None, 12288], name='x-input')
    y_ = tf.placeholder(tf.float32, shape=[None, num_clases], name = 'y-
input')
    x_image = tf.reshape(x, [-1,64,64,3])

    W_conv1 = weight_variable([8, 8, 3, 32])
    b_conv1 = bias_variable([32])

    h_conv1 = tf.nn.relu6(conv2d(x_image, W_conv1) + b_conv1)
    h_pool1 = max_pool_2x2(h_conv1)

    W_conv2 = weight_variable([5, 5, 32, 64])
    b_conv2 = bias_variable([64])


    h_conv2 = tf.nn.relu6(conv2d(h_pool1, W_conv2) + b_conv2)
    h_pool2 = max_pool_2x2(h_conv2)
```

```
    W_fc1 = weight_variable([12 * 12 * 64, 1024])
    b_fc1 = bias_variable([1024])

    h_pool2_flat = tf.reshape(h_pool2, [-1, 12*12*64])
    h_fc1 = tf.nn.relu6(tf.matmul(h_pool2_flat, W_fc1) + b_fc1)

    keep_prob = tf.placeholder("float")
    merged = tf.placeholder("float")
    accuracy = tf.placeholder("float")
    cross_entropy = tf.placeholder("float")
    keep_prob = 1.0
    h_fc1_drop = tf.nn.dropout(h_fc1, keep_prob)

    # softmax
    W_fc2 = weight_variable([1024, num_clases])
    b_fc2 = bias_variable([num_clases])

    # Use a name scope to organize nodes in the graph visualizer
    with tf.name_scope('Wx_b'):
        y_conv=tf.nn.softmax(tf.matmul(h_fc1_drop, W_fc2) + b_fc2)

    # Add summary ops to collect data
    w_hist = tf.histogram_summary('weights', W_fc2)
    b_hist = tf.histogram_summary('biases', b_fc2)
    y_hist = tf.histogram_summary('y', y_conv)

    # More name scopes will clean up the graph representation
    with tf.name_scope('xent'):
        cross_entropy = -tf.reduce_sum(y_*tf.log(y_conv))
        ce_summ = tf.scalar_summary('cross_entropy', cross_entropy)

    with tf.name_scope('train'):
        train_step = tf.train.AdamOptimizer(1e-4).minimize(cross_entropy)
        #train_step = tf.train.GradientDescentOptimizer(1e-
4).minimize(cross_entropy)

    with tf.name_scope('test'):
        correct_prediction = tf.equal(tf.argmax(y_conv,1), tf.argmax(y_,1))
        accuracy = tf.reduce_mean(tf.cast(correct_prediction, "float"))
        accuracy_summary = tf.scalar_summary('accuracy', accuracy)

    sess = tf.Session()
    sess.run(tf.initialize_all_variables())
    # Merge all the summaries and write them out to /tmp/mnist_logs
    #merged = tf.merge_all_summaries()

    merged = tf.merge_summary([accuracy_summary, ce_summ])
    writer = tf.train.SummaryWriter('/data/avelad/pfc/logs/sia_train',
sess.graph_def)

    for i in range(270):
      if i % 10 == 0:  # Record summary data, and the accuracy
        result = sess.run([merged, accuracy, cross_entropy], feed_dict={x:
batch[0], y_: batch[1]})
        writer.add_summary(result[0], i)
        print("Accuracy at step %s: %s" % (i, result[1]) + " - Loss: %s" %
(result[2]) )

      else:
        batch = train.next_batch(40)
        sess.run(train_step, feed_dict={x: batch[0], y_: batch[1]})

    train.read_testing_data()
    print("Accuracy Final. Num iter. = " + str(i) + " Keep prob:=" +
str(keep_prob) + "num classes = " + str(num_clases))
    print("test accuracy %g"% sess.run(accuracy, feed_dict={
       x: train.test_data, y_: train.test_labels}))
```

```
#
#
#    print("test accuracy %g"% sess.run(accuracy, feed_dict={
#       x: train.test_data, y_: train.test_labels}))
#    raw_input("Premi una tecla per continuar...")
#    print "accuracy", sess.run(accuracy, feed_dict={x: train.test_data, y_:
train.test_labels})
#    prediction=tf.argmax(y_conv,1)
#    z = prediction.eval(feed_dict={x: train.test_data}, session=sess)
#    print ("Prediccio: ") + str(z)
# descomentar para obtener los resultados de cada clase
#    for i in range(train.test_data.shape[0]):
#        print ("CLASE: " + str(np.where(train.test_labels[i]==1)) + " -
Prediccio: " + str(z[i]))
#        if i % 50 ==0:
#            raw_input("Premi una tecla per continuar...")

if __name__ == '__main__':
  tf.app.run()
```

15.2 Annex 2. SIA.py

```
# -*- coding: utf-8 -*-

#import cv2
import numpy as np
import glob
import os


class SIA(object):

    def __init__(self):
        self.permutation = 0
        self.pointer = 0

    def read_training_data(self):
        #Reads the data
        rows = 64
        cols = 64
        deep = 3
        num_clases = 30
        files = 500
        path = '/data/avelad/pfc/data/'
        if not os.path.exists('/data/avelad/pfc/data/SIA_data_30_64.npz'):
          im_id = 0
          self.labels = np.zeros((files*num_clases,num_clases))
          self.data = np.zeros((files*num_clases,rows*cols*deep))

          for j in range(num_clases):
            files = glob.glob(path+'Training/Class' + str(j) + '/'+'*.jpg')
            print(path+'Training/Class' + str(j) + '/'+'*.jpg')
            print ('Num_Files: ' + str(len(files)))
            for filename in files:
                im = cv2.imread(filename,-1)
                im=cv2.resize(im, (rows,cols),interpolation=cv2.INTER_AREA)
                #preproceso imagen
                im = im-np.average(im,(0,1))
                #Fin preproceso
                self.data[im_id,:]=np.reshape(im,(1,rows*cols*deep))
                position = j
                self.labels[im_id,position]=1
                im_id = im_id + 1
            #Save the data to disk the first time

          print('Saving data the first time')

np.savez('/data/avelad/pfc/data/SIA_data_30_64.npz',data=self.data,labels=self
.labels)

        else:
            print('Loading data')
            h = np.load('/data/avelad/pfc/data/SIA_data_30_64.npz')
            self.data = h['data']
            self.labels = h['labels']


        self.train = tuple((self.labels,self.data))
        self.limit = self.data.shape[0]
        self.permutation = np.random.permutation(self.limit)



    def next_batch(self,num):
        #there are enough samples
        if self.pointer + num < self.limit:
            ind = self.permutation[self.pointer:self.pointer+num]
```

```
            self.pointer += num
        else:
            remainder = self.limit-self.pointer
            new = num-remainder
            ind =
np.concatenate((self.permutation[remainder:self.limit],self.permutation[0:new]
))
            self.pointer = new
        batch = tuple((self.data[ind,:],self.labels[ind,:]))
        return batch



    def read_testing_data(self):
        #Reads the data
        rows = 64
        cols = 64
        deep=3
        num_clases=30
        path = '/data/avelad/pfc/data/'

        if not
os.path.exists('/data/avelad/pfc/data/SIA_test_data_30_64.npz'):


            files = glob.glob(path+'Test/'+'*.jpg')
            self.test_labels = np.zeros((len(files),num_clases))
            self.test_data = np.zeros((len(files),rows*cols*deep))
            im_id = 0
            for filename in files:
                im = cv2.imread(filename,-1)
                im = cv2.resize(im, (rows,cols),interpolation=cv2.INTER_AREA)
                #preproceso imagen
                im = im-np.average(im,(0,1))
                #fin preproceso
                self.test_data[im_id,:]=np.reshape(im,(1,rows*cols*deep))
                position = int((filename.split('*')[0][-2] +
filename.split('*')[0][-1]))
                print(str(position)+' '+str(im_id)+' '+filename)
                self.test_labels[im_id,position]=1
                im_id = im_id + 1

            #Save the data to disk the first time
            print('Saving the testing data the first time')

np.savez('/Users/Ana/PFC_Img/SIA_test_data_30_64.npz',data=self.test_data,labe
ls=self.test_labels)

        else:
            print('Loading data')
            h = np.load('/data/avelad/pfc/data/SIA_test_data_30_64.npz')
            self.test_data = h['data']
            self.test_labels = h['labels']
```

Annex 3. Jocs de Prova

064,,+,	20		20		20		20		20		20	
La$>G+'45)(	150		150		150		99:		220		220	
C'&@&KG$bc++@d	1		1		1		1		:-\		:-h	
04@4$(fg7G'4	128x128x3		;RU;RU		64x64x3		64x64x3		64x64x3		64x64x3	
gU+5K5	CPU		CPU		CPU		j32		GPU		GPU	
.4G58+	50		50		50		50		50		50	
3&&6)7	3x3-2x2		3x3-2x2		9U9i9U		2x2-2x2		2x2-2x2		2x2-2x2	
>G+'45	F55$*'4)	D&,,$*'4)	F55$*'4)	D&,,$*'4)	F55$*'4)	D&,,$*'4)	F55$*'4)	D&,,$*'4)	F55$*'4)	D&,,$*'4)	F55$*'4)	D&,,$*'4)
:	0,08	947,604	0,08	810,324	0,06	946,414	0,06	991,579	0	1426,79	0	1044,51
":	0,34	295,841	0,28	393,172	0,34	237,992	0,42	230,869	0,2	525,83	0,16	375,529
9:	0,36	183,117	0,3	250,911	0,4	207,542	0,48	141,731	0,18	504,237	0,3	321,127
M:	0,44	137,51	0,42	179,578	0,28	198,557	0,54	135,46	0,4	255,109	0,48	194,291
R:	0,5	107,409	0,48	120,45	0,8	64,9272	0,52	109,273	0,26	316,143	0,2	278,294
S:	0,74	41,2624	0,52	112,196	0,66	77,9491	0,56	84,0149	0,34	328,505	0,46	191,341
;:	0,7	61,8629	0,6	78,1907	0,6	72,4206	0,78	36,8936	0,46	278,367	0,44	201,15
Z:	0,66	80,9862	0,68	64,9456	0,68	60,4595	0,62	93,5305	0,52	200,429	0,36	186,227
\:	0,7	67,6483	0,66	54,263	0,7	69,9282	0,76	56,1837	0,42	203,609	0,56	129,205
h:	0,7	70,1412	0,82	38,2635	0,7	63,0092	0,66	80,0025	0,48	231,443	0,56	149,865
":::	0,76	43,8512	0,66	61,2364	0,78	67,014	0,74	44,0371	0,54	148,583	0,5	159,658
"":	0,76	46,3757	0,74	40,066	0,66	70,7901	0,7	46,133	0,44	183,189	0,52	120,853
"9:	0,86	26,5283	0,88	16,7317	0,82	32,7403	0,86	24,9782	0,56	126,584	0,62	112,271
"M:	0,84	35,4311	0,84	30,1004	0,76	45,81315	0,84	36,5077	0,46	207,139	0,46	120,332
"R:	0,86	32,348	0,84	36,383	0,86	27,2421	0,82	31,172	0,52	194,259	0,64	119,063
"S:	-	-	-	-	-	-	0,86	30,2783	0,64	139,606	0,6	115,491
";:	-	-	-	-	-	-	0,9	13,346	0,6	127,637	0,64	94,4198
"Z:	-	-	-	-	-	-	0,9	23,3185	0,6	123,454	0,7	87,726
"\:	-	-	-	-	-	-	0,88	25,2875	0,58	154,676	0,76	54,2403
"h:	-	-	-	-	-	-	0,96	14,2644	0,68	99,5102	0,68	96,3184
9::	-	-	-	-	-	-	0,86	30,2886	0,7	100,106	0,72	79,8181
9":	-	-	-	-	-	-	0,94	14,7899	0,74	113,369	0,82	42,5256
99:	-	-	-	-	-	-	-	-	-	-	-	-
9M:	-	-	-	-	-	-	-	-	-	-	-	-
9R:	-	-	-	-	-	-	-	-	-	-	-	-
9S:	-	-	-	-	-	-	-	-	-	-	-	-
9;:	-	-	-	-	-	-	-	-	-	-	-	-
F55K'45Y$*+	0,479		0,557		0,608		0,675		:-SM		0,579	