Universitat politècnica de Catalunya

Treball de Final de Grau

Compressió de Xarxes Neuronals

Ferran Noguera Vall

Director: Josep Llosa Espuny
Codirector: Eduard Ayguadé Parra
Especialitat: Computació

Facultat d'Informàtica de Barcelona

24 d'abril de 2019

Agraïments

M'agradaria aprofitar aquesta apartat per agrair als meus dos tutors Josep Llosa Espuny i Eduard Ayguadé Parra per la supervisió, ajut i temps dedicat a tirar endavant aquest projecte. Estic molt agraït per l'oportunitat que m'han donat per participar en aquesta investigació.

Un especial agraïment també a la meva família, en especial als meus pares per la seva paciència, esforç i consells sense els quals res hagués pogut ser possible.

Declaració

Declaro que aquest treball ha sigut realitzat per mi, Ferran Noguera Vall, i basat en les meves pròpies investigacions.

Tota informació que no pertany exclusivament al autor del projecte ha sigut especificada correctament en el document en forma de cita localitzada al final del treball.

Ferran Noguera Vall
24 d'abril de 2019

Resum

Avui en dia les xarxes neuronals estan en auge, fent-se cada cop més espai entre la quotidianitat de les persones, però aquestes segueixen tenint diversos problemes els quals n'estan endarrerint el seu avenç. Alguns d'aquests problemes són el seu elevat cost computacional i l'espai d'emmagatzematge que ocupen. En aquest projecte s'ha proposat una possible solució a aquests problemes, especialment centrada en l'emmagatzematge.

Es coneix de l'error-tolerància de les xarxes neuronals i s'aprofitarà d'aquesta propietat per elaborar diferents tècniques de compressió, especialment pensades per xarxes que continguin un nombre elevat de capes densament connectades doncs aquestes es representen a memòria com a matrius de neurones, generalment molt grans. El seu elevat cost emmagatzematge en limita el ús per certs dispositius.

S'ha proposat una sistema de compressió pensat per aquestes matrius i dut a terme durant l'entrenament de la xarxa. La base d'aquest ha consistit en dividir les matrius en blocs i intentar reduir el total de valors continguts en aquests. S'ha plantejat diferents propostes per aconseguir-ho.

El pas inicial va ser aplicar la coneguda mitjana aritmètica per cada bloc, limitant així el total de valors continguts per aquests a només un. Seguidament s'hi ha aplicat algorismes de *clustering*, donant així més llibertat als blocs a l'hora que es redueix igualment la quantitat de valors en aquests. El tercer pas va ser entrenar la xarxa lliurement un nombre fixat d'epochs i després aplicar-hi una de les tècniques dels passos anteriors. Finalment s'ha ajuntat les tres tècniques en una sola fent que hi hagi una reducció progressiva dels valors permesos per bloc al llarg de l'entrenament de la xarxa.

Els experiments han sigut favorables i mostren uns factors de compressió molt elevats sense perdre precisió, especialment en la tècnica de compressió progressiva. S'ha experimentat en diverses xarxes pels datasets MNIST i CIFAR-10.

Índex

Índex de figures

Índex de taules

Capítol 1

Context i abast del projecte

En aquest apartat s'ha introduït de manera general el tòpic tractat al llarg d'aquest treball, fent una formulació inicial del problema a resoldre, l'abast del projecte i, finalment, s'ha esmentat les diferents metodologies utilitzades per portar-lo a terme.

1.1 Context

Actualment la intel·ligència artificial és un dels tòpics més populars dins el món de la investigació en ciències de la computació. Aquesta consisteix en el desenvolupament i creació d'algorismes que permeten a una màquina prendre decisions amb cert grau d'intel·ligència, intentant aproximar-se al màxim possible a la humana. Ens podem trobar diferents aplicacions que usen aquest tipus de tecnologia en el nostre dia a dia, des d'un classificador de correus electrònics [1] passant per sistemes de recomanacions de tot tipus [2] fins a cotxes amb conducció automàtica [3], malgrat aquests últims encara estan en fase d'experimentació i no són d'ús públic.

Existeixen diferents branques derivades de la intel·ligència artificial, una d'elles és l'aprenentatge automàtic, més conegut pel seu terme angles *machine learning*. Aquesta utilitza diferents tècniques de probabilitat i estadística a partir de les quals, mitjançant una quantitat generalment molt gran de dades, ideen patrons i així poden "aprendre" a partir d'aquests, independentment, sense necessitat de programar aquest aprenentatge directament.

L'aprenentatge profund [4] és un subconjunt de l'aprenentatge automàtic, en el qual es creen algorismes que funcionen de manera similar però el qual està format per diverses capes d'aquests algorismes, d'aquí el concepte *"profund"*,

cadascuna d'elles proporciona una interpretació diferent a les dades d'entrada. Aquesta xarxa d'algorismes és el que anomenem xarxes neuronals artificials [5]. Per exemple, una xarxa que vulgui identificar el que apareix en una imatge farà que cadascuna de les seves capes tingui una funció diferent, s'aniran passant la informació a través de la xarxa, arribant a l'última capa on es donarà una predicció final.

Les xarxes neuronals artificials són sistemes computacionals que s'inspiren en les xarxes neuronals biològiques presents en els cervells dels éssers vius. Aquests sistemes tenen un procés d'aprenentatge, basat en prova i error, amb el qual a partir d'exemples aconsegueixen resoldre de manera molt encertada certs problemes concrets.

Actualment són usades en una gran varietat d'aplicacions, a continuació donarem uns quants exemples de les seves utilitats: Google va crear una xarxa neuronal especialitzada en la detecció de dibuixos [6] alhora que està generant un conjunt d'entrenament massiu gracies als usuaris que hi participin en l'enllaç `https://quickdraw.withgoogle.com/`, els anteriorment esmentats cotxes amb conducció automàtica [3], reconeixement de veu [7] i robòtica [8], entre molts d'altres.

1.1.1 Actors

En aquesta subsecció s'ha definit els actors d'aquest projecte, amés dels seus beneficiaris.

1.1.1.1 Desenvolupador

Jo he estat l'únic desenvolupador i encarregat de portar a terme la part pràctica i teòrica d'aquest projecte, incloent la redacció d'aquesta memòria. Les funcions a realitzar han sigut les de recerca, implementació, testeig, anàlisis de resultats i conclusions en forma de memòria que es presenta al final d'aquest. He estat l'encarregat de complir els terminis imposats pel director i codirector.

1.1.1.2 Director i Codirector

El director d'aquest projecte és en Josep Llosa i Espuny i el codirector és l'Eduard Ayguadé i Parra. Ells han estat els encarregats de gestionar, seguir i guiar al desenvolupador per a la correcta realització del projecte. El seu paper ha sigut essencial perquè el projecte hagi pogut acabar en èxit, ja que s'han encarregat de

supervisar periòdicament el treball realitzat pel desenvolupador i fer-li saber en tot moment si el camí escollit era el bo o n'hauria d'explorar un altre.

1.1.1.3 Beneficiaris

Degut a que aquest és un projecte d'investigació i no de creació de cap producte no sembla tenir un beneficiari directe més que realitzar pura recerca. Les persones interessades en les xarxes neuronals artificials que vulguin ampliar els coneixements en aquest tema o generar algun producte partint de les investigacions realitzades en aquest projecte tindran un punt de partida en el qual començar, per tant en seran els principals beneficiaris.

1.2 Formulació del Problema

Actualment les xarxes neuronals artificials estan limitades per la potència de hardware i emmagatzematge dels ordinadors corrents o dispositius on estigui pensat que s'executin. Aquest, entre altres motius, provoca que la seva expansió estigui en un punt d'estancament i hi hagi en marxa una recerca massiva per tal de solucionar-ho.

El problema del fet que es necessiti tanta potència de hardware rau en el seu procés d'aprenentatge, explicat en mes detall en l'apartat entrenament d'una xarxa neuronal. Aquest pot arribar a iterar milers o milions de vegades abans no n'extreu un resultat final, on per cada iteració es produeixen centenars de milers d'operacions aritmètiques. Tots aquests factors sumats fan que un ordinador normal i corrent pugui tardar dies, setmanes o inclús mesos a entrenar una xarxa neuronal artificial. Actualment s'ha comprovat que l'ús de GPUs millora notablement el temps d'entrenament gràcies a la seva elevada paral·lelització en comparació amb la CPU, malgrat això representa una despesa energètica massa elevada i encara existeix la necessitat d'utilitzar supercomputadors de clústers de GPUs, com és el cas del MinoTaure [9] o CTE-POWER [10] que s'han usat per realitzar d'aquest projecte.

L'emmagatzematge de la xarxa neuronal també és un problema actual, ja que una vegada acabat el seu entrenament s'hauran de guardar els valors que han quedat en totes les matrius de pesos durant l'últim epoch doncs s'usarà aquest estat final de la xarxa per resoldre els problemes que es plantegin. En cas de voler usar aquesta xarxa neuronal en un dispositiu petit o molt petit, com per exemple un mòbil o un microxip, es necessitarà que l'espai d'emmagatzematge que ocupi la xarxa sigui mínim, d'haver-n'hi una amb diverses capes "fully-connected"

denses es podria arribar a ocupar a l'alçada de diversos Gigabytes (GB), per tant es requereix minimitzar l'espai que d'emmagatzematge que ocuparan a fi de poder usar-se en qualsevol classe de dispositiu, sinó es corre el risc que no pugui cabre-hi, com passa actualment.

Malgrat en l'actualitat la gran majoria d'investigacions es centren en les xarxes neuronals convolucionals el cert és que segons un article publicat per Google [11], a juliol de 2016, només un 5% d'aquest tipus de xarxes estava corrent en els seus servidors respecte un 61% que eren perceptrons multicapa (multilayer perceptron) on totes les seves capes són *"fully-connected"*, les quals ocupen més en memòria i són més costoses d'executar motiu pel qual existeix aquesta necessitat de millorar el seu emmagatzematge i compressió.

La finalitat d'aquest projecte ha estat buscar una forma de comprimir l'esmentada matriu de pesos de forma que es pugui executar en un dispositiu hardware pensat específicament per aquest tipus de compressió i així s'aconsegueixi augmentar el rendiment a l'hora de córrer la xarxa neuronal i així intentar pal·liar els problemes de rendiment i excés d'emmagatzematge esmentats anteriorment. En aquesta memòria es parlarà d'aquest hardware d'una manera hipotètica doncs no s'ha creat i es deixarà obert a una possible extensió d'aquesta recerca.

1.2.1 Objectius del projecte

Aquest projecte ha tingut i complert un seguit d'objectius que es concretaran a continuació:

1. Cerca de noves tècniques de compressió de matrius aprofitant el hardware i centrades en xarxes neuronals profundes.

2. Aprenentatged del funcionament d'un framework complex de *deep learning* enfocat a xarxes neuronals profundes com és Caffe [12].

3. Interpretació dels resultats i extracció de conclusions vàlides sent capaç de comparar i justificar les millores de cadascuna de les diferents tècniques escollides.

4. Millorar el coneixement relacionat amb el camp del *machine learning*.

5. Aprenentatge en la creació i divulgació de projectes de recerca.

1.3 Abast

Aquesta secció serveix per marcar clarament fins on s'ha arribat amb la creació d'aquest projecte, definint els passos que s'ha de seguit i parts que contingudes en el projecte per haver aconseguit el seu correcte desenvolupament.

La principal fita del projecte ha sigut la ideació de diferents tècniques de compressió per matrius de pesos, usades en l'entrenament i execució de les xarxes neuronals, pensant executar-les en un hipotètic hardware específic per aquesta mateixa compressió. Amb aquest possible hardware, el qual no s'ha desenvolupat més que teòricament, s'aconseguiria optimitzar l'emmagatzematge i l'energia consumida a l'hora de l'execució de la xarxa comprimida, sempre intentant mantenir un tant per cent d'error de la xarxa similar o menor a l'inicial malgrat la seva compressió.

Al començar es va realitzar un estudi exhaustiu sobre el context actual començant per la part més genèrica d'intel·ligència artificial, passant per l'aprenentatge automàtic i arribant fins a entendre el funcionament de cada actor que compon una xarxa neuronal profunda.

Una vegada s'havia obtingut el coneixement sobre el context actual es va fer un estudi detallat i exhaustiu de l'estat de l'art sobre el problema que s'ha enfrontat, en aquest cas la compressió de les xarxes neuronals. El desenvolupador s'ha llegit diversos papers i articles científics, alguns d'ells adjuntats en la memòria, que li van fer saber fins a on s'havia arribat en la investigació actual, quines millores s'hi havia aplicat i si aquestes s'han pogut reutilitzar, ni que fos teòricament, en aquest problema. Es realitzà un estudi de diferents tècniques de compressió, independents a les xarxes neuronals, per explorar si es podrien utilitzar aquestes en el problema a abordar.

Un cop van estar tots els objectius marcats i els coneixements necessaris assolits va ser moment de crear un entorn de treball idoni per començar el projecte d'investigació: scripts per automatitzar processos, generació de gràfics, etc. En aquest cas es va haver de muntar inicialment dos entorns: un a l'ordinador personal i l'altre al supercomputador MinoTaure [9] però degut a manteniments en aquest últim també s'en muntà un al supercomputador CTE-POWER [10], per sort al compartir directoris amb el MinoTaure no va generar masses problemes. Una vegada hi va haver un bon entorn va ser el moment d'instal·lar el framework escollit per treballar amb les xarxes neuronals artificials, Caffe [12], i tot el software que es necessita per la seva utilització. Aquests frameworks tenen una complexitat relativament alta, així que es va requerir de temps per familiaritzar-s'hi.

Arribats en aquest punt va ser el moment de començar amb la part que va requerir de major dedicació temporal, la recerca de tècniques per la compressió per la matriu de pesos de les xarxes neuronals. Aquest va ser un procés d'investigació i implementació que es va realitzar paral·lelament, sempre documentant tot resultat i conclusions obtinguts.

Per concloure el projecte es van agrupar tots els resultats obtinguts al llarg de les diferents implementacions i es va fer una comparativa per determinar quina tècnica és la més òptima, havent estudiat els avantatges i desavantatges de cadascuna d'elles. Finalment s'escriurà aquesta memòria on s'explicarà en detall tot el que s'ha fet, aportant unes conclusions autocontingudes i justificades.

1.3.1 Possibles Problemes

Abans de la realització d'aquest projecte i, perquè pogués ser entregat en el termini acordat, es va fer una detecció dels problemes que era possible que apareguessin i formes d'afrontar-los perquè no produïssin un retard excessiu:

- **Falta de temps:** El projecte s'ha hagut de realitzar en un termini molt dens i curt durant el qual no s'hi ha pogut dedicar exclusivament gran part del temps, ja que el desenvolupador estava fent diferents assignatures de la carrera alhora. Per tal de no quedar-se sense temps es va fixar un calendari realista que es va haver de complir religiosament. Es va treballar tenint en compte una possible desviació de dos setmanes sobre el temps programat, per tenir marge de millora en cas d'anar endarrerit.

- **Bloqueig en la cerca de tècniques de compressió:** Durant aquest projecte es va haver de posar a prova diferents tècniques de compressió ideades entre el desenvolupador, director i codirector. Aquestes eren només teòriques per tant hi havia una possibilitat de que no funcionessin, d'haver sigut així s'hagués optat per buscar-ne de noves i existia la possibilitat de que es produís un bloqueig a l'hora de trobar-les. D'haver-se produït una situació així s'hagués fet una reunió de *brainstorming* [13] per tal d'encaminar de nou el projecte.

- **Bugs:** Com en tot projecte informàtic era molt probable que apareguessin diferents errors durant la implementació i desenvolupament del projecte. A fi de poder localitzar-los i arreglar-los en un temps raonable es va usar *unity testing* [14], el qual analitzava la robustesa del programa i així minimitzava l'aparició d'aquests. En cas d'haver aparegut un error que el desenvolupador no hagués sigut capaç de resoldre s'hagués programat una reunió excepcional

amb el director i codirector per tal de poder solucionar-ho el més ràpid possible, per sort no ha sigut necessari.

- **No disposició de Hardware:** El supercomputador Minotaure [9] era essencial per la realització d'aquest treball doncs sense ell i utilitzant-ne un de comú l'entrenament de les xarxes podria durar inclús setmanes. Com s'ha explicat, aquest va tenir una fase de manteniment d'unes quantes setmanes, moment en el qual es va haver de fer ús de la solució que s'havia ideat a aquest possible problema: es va acudir al director i codirector i aquests van proporcionar al desenvolupador un nou supercomputador suplent, el CTE-POWER [10] en el qual es va replicar l'entorn de treball.

- **Articles qüestionables:** Al ser un tema d'investigació es va haver de consultar molts articles i webs científiques de les que extreure informació. Es va considerar la possibilitat de que algun d'aquests contingués informació falsejada, malgrat és un cas molt poc comú el qual no es va donar, que alentís l'avenç del projecte amb experiments inútils. S'hagués intentat buscar responsables, de trobar un paper científics així.

- **Malaltia o situació personal:** En cas d'haver estat el desenvolupador, director i/o codirectors incapacitats per treballar temporalment hagués afectat molt perjudicialment en la planificació del treball. D'haver ocorregut s'hagués hagut de mantenir una comunicació excel·lent amb el director i codirector a fi de reorganitzar completament el calendari, tasques i temps dedicats a cada procés. Per sort aquesta situació mai es va donar.

1.4 Metodologia i Rigor

En aquest apartat s'han descrit la metodologies emprades per la realització del projecte i les eines de seguiment pel correcte desenvolupament d'aquest. Al llarg de tot el projecte no s'ha requerit cap modificació en les metodologies ni amb eines de seguiment utilitzades.

1.4.1 Metodologia de Treball

El projecte s'ha realitzat en un termini curt i intents, com a tal la millor opció va ser la d'utilitzar metodologies àgils [15] per a la seva realització ja que aquestes ens proporcionen una flexibilitat que permeté acabar el projecte en menor temps que usant qualsevol de les altres metodologies. L'ús d'aquestes metodologies està molt pensat per equips i seguir-les religiosament en un projecte en solitari no té

molt sentit, tot i això la filosofia dels seus mètodes, comentats a continuació, sí que es van aplicar.

1.4.1.1 Cicles de desenvolupament curts

Es van utilitzar cicles curts amb objectius setmanals que ens van permetre mantenir una millor planificació del projecte, sent conscients del seu estat actual, i van ajudar a prevenir possibles desviacions temporals.

1.4.1.2 Intensa comunicació amb els clients

Malgrat en aquest projecte no existís cap client real podem entendre com aquest als responsables del seguiment del treball, el director i codirector. Plantejats d'aquesta manera va ser molt útil aquest concepte de les metodologies àgils [15], ja que es van mantenir reunions setmanals i comunicacions periòdiques amb els responsables, fet que serví de guia per marcar un bon camí en la investigació.

1.4.2 Eines de Seguiment

En aquest apartat es tractarà d'especificar les diferents eines de seguiment que han utilitzat els actors partícips d'aquest projecte.

1.4.2.1 Desenvolupador

El desenvolupador va fer ús d'eines amb sistema de control de versions per disposar d'una fàcil recuperació d'aquest en cas d'errors en el codi. D'entre totes es va escollir Git [16] doncs el framework Caffe està disponible en ell i és de les més famoses i usades a nivell mundial.

1.4.2.2 Director i Codirector

Pel seguiment, control, planificació de reunions i comunicació amb el director i codirector es va utilitzar: Gmail i Google Drive. Al usar la plataforma Git també podien usar el seu compte propi per veure el codi font amb el que estava treballant el desenvolupador.

1.4.3 Mètodes de Validació

Aquests mètodes van ser molt importants pel correcte desenvolupament i implementació del projecte, ja que aquests van ajudar a agilitzar molt el procés de localització, tractament d'errors i fase de testeig.

Es van crear jocs de prova i, com s'ha comentat amb anterioritat, un sistema de *unity testing* [14] que va haver de passar qualsevol implementació, realitzada pel desenvolupador, abans de ser mesclats amb el codi font principal. En cas de no passar-los s'hagués utilitzat Gmail, les reunions periòdiques o el sistema de *issues* intern de Git per demanar ajut amb el director i/o codirector.

Capítol 2

Planificació Temporal

Va ser important seguir una estricta planificació temporal per tal que el projecte es pogués dur a terme en un marge de temps tan curt. En aquest apartat s'ha: justificat el temps de durada, descrit i especificat temporal cadascuna de les tasques realitzades conjuntament amb el diagrama de Gantt que s'ha seguit, descrit els diferents tipus de recursos emprats per la realització del treball i explicat el pla d'acció que es va idear per afrontar la possible aparició de problemes durant el projecte.

2.1 Duració del Projecte

La durada d'aquest projecte era inicialment de sis mesos, però finalment s'ha hagut d'ampliar a deu mesos. Va començar a principis de Juliol del 2018 i s'ha acabat a mitjans d'Abril del 2019. S'ha de tenir en compte que durant tot el mes d'Agost i diverses setmanes de Gener i Febrer el desenvolupador va estar de vacances i no es va avançar sobre el treball, a més que de Setembre a Gener el desenvolupador va haver de combinar el projecte amb quatre assignatures, conjuntament, i va alentir el projecte molt més de l'esperat, per això s'ha presentat en segona convocatòria a finals Abril de 2019.

2.2 Descripció de les Tasques

En aquest apartat s'ha anomenat cronològicament totes les tasques a realitzar durant el projecte conjuntament amb una breu descripció de cadascuna d'elles.

2.2.1 Adquisició de coneixement sobre xarxes neuronals

Aquest era un tema totalment nou per al desenvolupador, així que per començar es va haver de realitzar un estudi previ sobre els camps més bàsics tant de *Machine Learning* com *Deep Learning*, fent especial èmfasi a les xarxes neuronals profundes i el seu funcionament intern.

Inicialment el desenvolupador realitzà un curs online a Coursera sobre Aprenentatge Automàtic [17] anomenat *"Machine Learning"* impartit per Andreu Ng de la universitat de Stanford, amb aquest curs es va entendre el funcionament i definicions més generals de l'aprenentatge automàtic. Un cop finalitzat el curs, per aprofundir més sobre les xarxes neuronals, es va llegir el llibre "Deep Learning with Python" de Nikhil Ketkar [4] a més d'altres llibres més [5].

Els recursos hardware usats en aquest procés van ser l'ordinador i impressora personal, els software l'ús del Pack Office per fer esquemes i resums, com a recursos humans es va requerir únicament del desenvolupador.

2.2.2 Estudi exhaustiu de l'estat de l'art

Una vegada el desenvolupador va disposar del coneixement necessari per comprendre el funcionament de les xarxes neuronals va procedir a informar-se exhaustivament sobre l'estat de l'art actual per tal de determinar, entre altres coses: conjunts d'entrenament i proves amb xarxes neuronals profundes d'exemple per poder elaborar els estudis i comparacions, un framework ideat per tractar amb elles, quines tècniques de compressió s'havien intentat i havien funcionat les quals li van servir per agafar idees sobre quins retocs es podria fer per millorar l'idea plantejada, etc.

Els recursos hardware i software emprats en aquest procés van ser els mateixos que en l'anterior, però aquí es va necessitar al director i codirector com a recursos humans per programar-hi reunions per resoldre dubtes i facilitar-li articles interessants al desenvolupador.

2.2.3 Familiarització i preparació l'entorn de Treball

Per la realització d'aquest projecte s'ha utilitzat el framework de *Deep Learning* "Caffe" [12]. Aquest és un software complex a causa de: la seva instal·lació enrevessada, manera de tractar les dades i operar, estructura interna, etc. Per culpa d'això es va requerir d'un temps extra en aquest procés per tal de

que el desenvolupador es familiaritzés amb aquest i entengues perfectament el seu funcionament, ja que durant el desenvolupament del projecte va ser necessari modificar el codi font d'aquest. Caffe disposa d'una documentació amb explicacions del seu ús `http://caffe.berkeleyvision.org/doxygen/` i GitHub públic [12] per la seva instal·lació que van ser molt útils.

Una vegada es va haver entès l'estat de l'art actual on es troba el camp d'investigació, es va haver de preparar un entorn de treball òptim. La preparació d'aquest entorn va necessària en dos llocs diferents: tant el supercomputador Minotaure [9], com en el personal. Degut a manteniments en el Minotaure també es va haver de preparar aquest entorn en un supercomputador diferent: CTE-POWER [10]. Dins d'aquest procés es va realitzar la preparació de scripts per agilitzar els processos de testeig i generació de software.

Els recursos hardware que es van utilitzar són: MinoTaure, CTE-POWER i ordinador personal; els software van ser: Caffe, documentació sobre el seu funcionament i el software necessari per instal·lar-lo i fer-lo funcionar, a més de Git i el pack office per prendre apunts. Com a recursos humans només es necessitar al desenvolupador.

2.2.4 Implementació de diferents formats de compressió

En aquest procés es va implementar i comprovar el funcionament de les diferents tècniques de compressió que s'havien ideat. Aquesta va ser la tasca amb la durada més llarga i densa de totes.

Per la realització de les diferents implementacions es va fer ús dels diferents subprocessos que s'utilitzen per la creació de software: anàlisis, disseny, implementació i testeig, tal com es pot veure en la figura 2.1. A continuació es detallarà cadascun d'aquests subprocessos de forma cronològica:

- **Anàlisis:** Es busca noves formes de comprensió mitjançant millores d'anteriors estudis.
- **Disseny:** Es dissenya aquesta tècnica extreta de l'anàlisi i comprova si la seva implementació és viable o s'ha de tornar al pas anterior i buscar-ne una que ho sigui.
- **Implementació:** S'implementa la tècnica, modificant el framework Caffe, per tal poder comprovar l'efectivitat d'aquesta.
- **Testeig:** Finalment es realitza un testeig final de la implementació, comprovant que no hi ha hagut cap error, de ser així es tornaria un pas

enrere, a la fase d'implementació, per arreglar-ho. S'extraurà els resultats i conclusions corresponents.

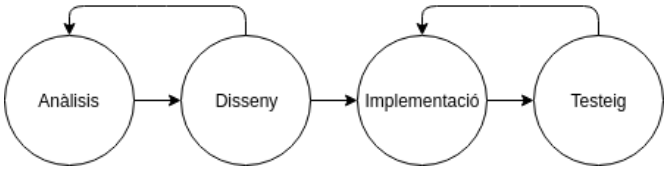

Figura 2.1: Diagrama creació de Software

Com a recursos software es va requerir: el framework de deep learning Caffe [12] conjuntament amb tot el programari perquè funcioni correctament, Git i LaTeX [18]. Els recursos hardware que es van necessitar van ser: el supercomputador MinoTaure [9], el CTE-POWER [10] i l'ordinador personal. Finalment, com a recurs humà van ser necessaris: el director, codirector i desenvolupador, ja que es va realitzar un seguiment periòdic.

2.2.5 Conclusions i Resultats

Aquest va ser l'última tasca del projecte la qual va consistir en agrupar tota la informació generada en els anteriors processos i realitzar un informe, a mode de memòria, on s'ha explicat tot el realitzat: les diferents tècniques de compressió usades, els guanys/errors que han generat, etc.

El final d'aquesta tasca va concloure amb la generació del document final del projecte. Els recursos software emprats van ser: Google Drive, Gmail, LaTeX [18] i el Pack Office per la realització de la memòria i gràfics. Com a recursos hardware es va necessitar: la impressora i ordinador personals i com a recurs humà al desenvolupador i al director i codirector per garantir la claredat i correcte redacció del treball.

2.3 Estimació Temporal

Segons la normativa de treballs de final de grau de la Facultat d'Informàtica de Catalunya (FIB) [19], aprovada el 21/10/2015, aquest consta de 18 crèdits "European Credit Transfer and Accumulation System" (ECTS) cadascun d'ells amb una duració estimada de 30 hores. Partint d'aquesta base es va estimar una duració aproximada de 540 hores, la qual ha acabat coincidint amb la que realment ha durat aquest.

En la taula 2.1 es pot veure la durada que va representar cada procés respecte al total de 540 hores.

Tasca	**Hores**
Adquisició de coneixement sobre DNN	60
Estudi exhaustiu del estat del art	130
Familiarització i preparació l'entorn de Treball	80
Implementació de diferents formats de compressió	200
Conclusions i Resultats	70
	Total: 540h

Taula 2.1: Estimació horaria

2.4 Diagrama de Gantt

En aquesta secció s'explica el diagrama de Gantt [20], repartit amb les diferents tasques. Per entendre la distribució horària s'ha de tenir en compte que a partir del 9 de setembre de 2018 el desenvolupador va tornar a començar la universitat i les hores de dedicació no van poder ser exclusives. S'ha especificat els 3 períodes de vacances on el desenvolupador del projecte no va estar disponible i, per tant, no es va poder avançar. S'ha hagut de modificar respecte al Gantt inicial perquè la previsió de finalització era finals de Gener i s'ha allargat fins a finals d'Abril. Degut a la mida d'aquest s'ha adjuntat als apèndix

2.5 Especificació dels recursos

En aquest apartat s'especifica els recursos que s'han utilitzat durant la realització projecte, dividint-los segons siguin: Hardware, Software o Humans.

Hardware:

- Ordinador de sobretaula personal
- Ordinador portàtil personal
- Supercomputador MinoTaure [9]
- Supercomputador CTE-POWER [10]
- Impressora personal

Software:

- Framework Caffe [12] amb el software requisit perquè funcioni correctament
- Google Drive
- Git [16]
- Gmail
- Editor de texts
- Pack Office 2010
- Linux Mint 18
- Windows 10 Pro
- LaTeX amb la plataforma online gratuïta OverLeaf [18]

Humans:

- Director i Codirector
- Professor assignat de GEP
- Desenvolupador (jo)

2.6 Alternatives i pla d'acció

Aquesta és la secció de la memòria on s'ha valorat les alternatives i plans d'acció que es van elaborar davant l'aparició de possibles desviacions i problemes durant la realització del projecte i la manera d'afrontar-los. Només es va disposar de 10 mesos per la realització d'aquest treball, per tant es va haver de ser curós a l'hora de tenir en compte els problemes que poguessin endarrerir-lo així, en cas que haguessin succeït, s'hagués pogut afrontar a temps i evitar un retard considerable.

Durant la realització del projecte hi ha hagut diferents problemes iguals o similars que han succeït i, aplicant el pla d'acció ideat, s'ha permès que l'impacte temporal no hagi sigut tan gran com per no arribar a l'entrega del projecte.

Com s'ha comentat anteriorment, les metodologies àgils [15] permeten revisar i adaptar dinàmicament el temps que s'havia pensat inicial per una tasca, així en el cas que un procés s'hagués allargat més del previst s'hauria pogut fer adaptar a la resta d'aquests per, igualment, haver acabat en el temps estipulat, sempre tenint en compte que haver fet això comportaria un augment de la previsió d'ús dels recursos.

2.6.1 Bugs

Caffe [12] és un framework complexe, per tant la modificació d'aquest hagués pogut comportar una generació alta de bugs, alguns cops difícils de trobar i/o arreglar, produint un endarreriment imprevist en la part d'implementació i testeig.

Per tal d'evitar aquest problema s'han realitzat execucions parcials del codi i reunions periòdiques amb el director i codirector, que s'haguessin incrementant en cas de que l'avanç del treball estigués bloquejat per un bug.

S'ha treballat tenint en compte una possible desviació de dos setmanes sobre el temps estimat inicialment, per tenir marge de millora en cas d'haver anat endarrerit. De trobar-se en aquesta situació s'hagués de fer un augment de l'ús dels recursos humans, en aquest cas el director i codirector, programant reunions extraordinàries.

Durant la realització del projecte no ha aparegut un bug suficientment gran com per entorpir l'avenç d'aquest.

2.6.2 Falta de Hardware

El supercomputador MinoTaure [9] era una eina imprescindible per realitzar aquest treball, doncs de no comptar amb ell l'entrenament de les xarxes neuronals seria extremadament lent i endarreria molt el procés d'implementació, tant que hagués acabat repercutint en la data d'entrega. En cas d'inhabilitació temporal del MinoTaure s'han proposat tres solucions, demanar temps de còmput en el supercomputador MareNostrum [21] per la realització dels entrenament, demanar accés a un altre supercomputador i/o replicar l'entorn de treball en el ordinador personal, sent aquesta última la menys desitjable al també ser la més lenta.

En cas d'haver fallat l'ordinador personal s'hagués demanat ajuda al director, codirector i UPC per adquirir-ne un de recanvi temporalment, mentre el personal estigués en reparació.

L'impacte en cas d'haver-hi hagut alguna fallada d'algun dels dos elements hardware hagués sigut innocu. En cas de ser el MinoTaure, malgrat que havent de reproduir l'entorn en el nostre ordinador personal faria que anés considerablement més lent, es pot assegurar que el Barcelona Supercomputing Center (BSC), entitat al que pertany, l'intentarà arreglar el màxim de ràpid possible. D'haver estat l'ordinador personal simplement es demanaria per un canvi temporal que seria simple de trobar.

D'haver-se trobat en una situació així s'haurà d'abusar de recursos hardware com ara els ordinadors personals en cas de fallada del MinoTaure o usar-ne de nous, de manera temporal, en cas de fallada d'algun dels ordinadors personals.

Aquest problema s'ha enfrontat doncs el Minotaure [9] va estar diverses setmanes en manteniment impedint al desenvolupador poder implementar i testejar el codi. Aquest es va posar en contacte amb el director i codirector per aconseguir solucionar-ho i li van facilitar un nou supercomputador, CTE-POWER [10].

2.6.3 Bloqueig en la cerca de tècniques de compressió

D'haver-se produït aquest problema hagués estat dels més crítics que podien succeir per l'avenç del projecte. Es va dedicar un temps considerable al procés d'anàlisis de noves tècniques de compressió, tal com s'especifica en el *Gantt*, tenint en compte ja aquest hipotètic cas de no trobar-ne de noves o que cap de les ideades representés millores significatives. D'haver sigut necessari s'hagués d'augmentat el temps dedicat a aquest procés, com s'ha mencionat anteriorment, s'ha treballat tenint en compte dos setmanes de marge de desviació per situacions crítiques com aquestes.

Si el desenvolupador s'hagués trobat en una situació així, per fer-li front, hi hauria hagut un augment considerable en les reunions amb el director i codirector, per tal d'evitar l'estancament del desenvolupador i fer una sessió de pluja d'idees [13] per trobar-li un nou enfocament. Haver fet això provocaria un augment dels recursos humans, entre ells el director i codirector.

Si bé no hi ha hagut el problema exacte, si que hi ha hagut moments on les tècniques de compressió inicialment plantejades no han acabat de donar el rendiment esperat, però gràcies a les reunions setmanals amb el director i codirector s'ha pogut redreçar el rumb del projecte.

2.6.4 Actualització del Framework

Caffe [12] és una plataforma en constant millora i desenvolupament per tant existia la possibilitat d'haver entrat d'una actualització suficientment gran com perquè fes trontollar els conceptes que el desenvolupador havia aprés d'aquest fins al moment, cosa que l'hagués obligat a tornar al procés de familiarització amb el framework, endarrerir així el projecte.

Per tal d'evitar aquest problema s'ha fet especial èmfasis en la tasca de familiarització amb Caffe, on el desenvolupador va prendre apunts de tot el que creia rellevant, així en cas d'una actualització d'aquest nivell no hagués sigut complicat tornar a estar al dia. Al ser un únic desenvolupador i tenir assimilats i anotats tots els conceptes relacionats amb el framework, en l'hipotètica entrada d'una actualització massiva s'hagués perdut un temps mínim a adaptar-se. S'ha de fer un bon ús dels recursos software de pressa d'apunts com és el pack Office.

Malgrat que Caffe [12] és un framework de codi obert amb constant desenvolupament, tal i com s'ha precisat, durant aquests deu mesos de projecte no hi ha hagut cap actualització massiva del codi. Tot i que el treball ha tingut deu mesos de durada, no en tots s'ha necessitat modificar Caffe, només en els últims.

2.6.5 Malaltia o situació personals

El desenvolupador ha sigut el principal encarregat de tirar endavant el projecte, en cas d'una malaltia o situació personal que l'hagués impedit seguir treballant-hi temporalment hauria modificat considerablement la planificació inicial proposada. En cas d'haver aparegut una situació així s'hauria solucionat mantinguen una comunicació excel·lent amb el director i codirector per tal de poder tornar a reordenar la planificació temporal i, conjuntament, augmentar la dedicació en aquest.

S'ha treballat amb dos setmanes de marge per si s'haguessin produït desviacions d'aquest estil, tal com s'ha comentat amb anterioritat. S'hagués fet un augment dels recursos humans, entre ells el director i codirector.

Per sort, cap dels recursos humans emprats per l'elaboració d'aquest projecte ha tingut cap malaltia o situació personal suficientment destacable com per no poder portar a terme els deures lligats al projecte.

Capítol 3

Pla Econòmic i Sostenibilitat del projecte

En aquest capítol de la memòria s'ha analitzat en profunditat el pressupost que ha representat la creació d'aquest projecte, desglossant-lo en els diferents tipus de costs: directes, indirectes i imprevistos. Afegint un tant per cent de contingència per tal de ser el màxim de precisos possibles. Finalment s'ha explicat el control de la gestió realitzada en pressupost per haver-lo mantingut a ratlla en tot moment.

S'ha mostrat la matriu de sostenibilitat d'aquest projecte i analitzat les seves tres dimensions: mediambiental, econòmica i social.

3.1 Pressupost del projecte

En aquest apartat de la memòria s'ha estimat els diferents costos generats en la creació d'aquest projecte, tenint en compte els recursos mencionats anteriorment. S'ha fet una distinció en tres tipus diferents de costos: directe, indirecte i imprevist. En el cost directe també es farà una diferenciació entre els recursos humans, software i hardware. Tots els preus esmentats en aquest capítol aniran amb impost sobre el valor afegit [22] (IVA) (21%) inclòs.

3.1.1 Costos Directes

En aquesta subsecció s'ha diferenciat tres tipus de recursos als quals avaluar-ne el seu cost: recursos humans, recursos software i recursos hardware.

3.1.1.1 Recursos Humans

Aquesta part està dedicada al anàlisis de costos humans especificats amb anterioritat. El projecte fou desenvolupat, dissenyat, analitzat i testejat per un únic desenvolupador, jo, però aquest ha precisat del suport d'un director i codirector, els quals fóren els encarregats de realitzar un seguiment i avaluació del projecte a través de reunions periòdiques per avaluar el desenvolupament del treball i suport tècnic en cas de ser necessari.

Posició	**Hores**	**€/Hora**	**Sou**
Director	50	50 [23]	2.500 €
Codirector	40	50 [23]	2.000 €
Desenvolupador	540	35 [23]	18.900 €
Total	**630**		**23.400 €**

Taula 3.1: Pressupost dels recursos humans

3.1.1.2 Recursos Hardware

Seguidament es procedí a detallar els costos dels recursos hardware emprats en la realització d'aquest projecte. S'ha dividit la taula següent especificant el cost del producte en el moment de la seva compra, el temps de vida útil estimat i l'amortització en €/hores.

L'ordinador de sobretaula personal està muntat a peces, com a tal s'ha especificat el cost individualment de cadascuna d'elles amb una estimació total de 4 anys, doncs s'estima que l'ordinador de sobretaula durarà aquests anys en total, aquest va ser comprat fa 3 anys així doncs es tindrà en compte el preu que tenia llavors. Els supercomputadors MinoTaure [9] i CTE-POWER [10] malgrat no aparèixer el seu cost no significa que siguin gratuïts, el problema és que aquestes dades no es poden adjuntar al no ser de domini públic.

Hardware	**Cost() €**	**Temps de vida (anys)**	**Amortització (€/hores)**
Asus Z97-K	133 [24]	4	0,018
GeForce GTX 970	359 [24]	4	0,050
Intel Core i5-4960K	224 [24]	4	0,031
RAM: DDR3 16GB	97 [24]	4	0,013
SSD 120GB	62 [24]	4	0,008
Disc dur: 1TB	45 [24]	4	0,006
MCPU2	33 [24]	4	0,004
Radix VII AG 700W	62 [24]	4	0,008
Ordinador portàtil	800 [24]	4	0,111
MinoTaure	-	-	-
CTE-POWER	-	-	-
HP Deskjet 2630	50 [24]	3	0,009
Total	**1865**		**0,258**

Taula 3.2: Pressupost dels recursos hardware

3.1.1.3 Recursos software

A continuació es detallà els costos dels recursos software que s'han necessitat pel correcte desenvolupament del projecte. S'utilitzà majoritàriament software gratuït i de codi obert, com és el cas de Caffe [12] o LaTeX. Per la redacció de la memòria s'usà d'aquest últim, un sistema de composició de textos, orientat a la creació de documents escrits que presentin una alta qualitat tipogràfica, s'ha utilitzat a través de la plataforma online OverLeaf [18] utilitzant la seva versió gratuïta.

Software	Cost() €	Temps de vida (anys)	Amortització (€/hores)
Caffe	-	-	-
LaTeX	-	-	-
Google Drive	-	-	-
Git	-	-	-
GMail	-	-	-
Editors de text	-	-	-
Pack Office 2010	50 [25]	3	0,019
Linux Mint 18	-	-	-
Windows 10 Pro	150 [25]	3	0,028
Total	**200**		**0,037**

Taula 3.3: Pressupost dels recursos software

3.1.2 Costos indirectes

Els costos indirectes fan referència als que han sigut conseqüència externa del desenvolupament del projecte. La investigació es realitzà en una universitat, per tant era inevitable el cost de l'electricitat, mobilitat, material acadèmic, accés a Internet i impressions, tal com queda reflectit en la taula següent.

Producte	Unitats	Cost	Total
Electricitat	250kWh[26]	0,15kWh € [27]	37.5
Targeta de mobilitat T-Jove	1	105 € [28]	105
Bolígraf	3	2 €	6
Pack 500 fulls	1	2 €	2
Impressions	150	0,03 €	4,5
Accés a Internet	4 mesos	30 €/mes	120
Total			**276**

Taula 3.4: Costos indirectes

3.1.3 Costos imprevists

En aquesta secció s'ha especificat els costos sorgits d'imprevistos durant la realització del projecte. Tenir en compte aquest factor ajudà a tenir un cert marge d'error en el pressupost que podrà ajudar en l'aparició d'esdeveniments imprevists.

Com s'ha comentat en el capítol de planificació temporal, es treballà tenint en compte una possible desviació de dos setmanes sobre el temps programat per poder afrontar l'aparició dels possibles problemes comentats amb anterioritat i, tot i això, entregar el projecte sense retards. Aquestes possible desviacions s'han tingut presents a l'hora de gestionar els costs imprevists, en total significà una addició de 80 € més del projecte. En la taula 3.5 s'especifica les hores de desviació donades pels recursos humans.

Posició	**Hores**	**€/Hora**	**Sou**
Director	10	50 [23]	500 €
Codirector	5	50 [23]	250 €
Desenvolupador	80	35 [23]	2.800 €
Total	**95**		**3.550 €**

Taula 3.5: Costos imprevists

També s'ha tingut present aquestes desviacions en l'ús dels recursos software i hardware. Per tal de calcular-los s'utilitzà la fórmula següent:

$$CostsImprevists = 0.3 \frac{€}{\text{hora}} \cdot 80\,\text{hores} = 24\,€ \tag{3.1}$$

On $0.3\frac{€}{hora}$ és la suma de l'amortització dels recursos software i hardware i les 80 hores són les possibles hores desviacions esmentades anteriorment.

En total els costs imprevists fóren **3.574 €**.

3.1.4 Contingències

En aquest apartat s'ha declarat un percentatge de contingència incremental per possibles oblits en l'estimació del pressupost.

El percentatge escollit de contingència per aquest projecte fou de 5%, s'ha triat un percentatge baix degut al gran nivell de detall pressupostari del que disposa el projecte, al tenir aquests fets en compte es considerà improbable l'aparició d'aquest recurs inesperat doncs tots els recursos essencials requerits estan correctament detallats i en cas de que aparegués el seu cost seria ínfim.

3.1.5 Pressupost Total

A continuació s'especifica el pressupost total que s'ha de destinat a la realització d'aquest projecte, tenint en compte el desglossament realitzat en els anteriors apartats a aquest.

Tipus	**Cost**
Recursos Humans	23.400 €
Recursos Hardware	1.865 €
Recursos Software	200 €
Recursos Indirectes	276 €
Recursos Imprevists	3.574 €
Subtotal	**29.290 €**
Contingència 5%	1.464,5 €
Total	**30.779,5 €**

Taula 3.6: Pressupost total

Finalment procedirem a especificar el pressupost destinat a cada tasca nombrada anteriorment en la secció 2.2. Aquest càlcul estimat es realitzà mitjançant el tant per cent d'hores dedicades segons la planificació total i el cost total del pressupost.

Tipus	Cost
Adquisició de coneixement sobre DNN	3.417,16 €
Estudi exhaustiu del estat del art	7.403,86 €
Familiarització i preparació l'entorn de Treball	4.556,22 €
Implementació de diferents formats de compressió	11.390,55 €
Conclusions i Resultats	3.986,7 €

Taula 3.7: Pressupost segons tasca

3.2 Control de gestió

Per tal de tenir un control sobre el pressupost estimat i assegurar-nos que no ens estem desviant del que s'ha marcat, al final de cada tasca s'ha realitzat una avaluació de les hores realitzades i pressupost gastat. Aquests números fóren comparats amb els estimats prèviament per marcar un control sobre el total de desviació al que s'ha vist sotmès.

Es farà ús les següents fórmules per tal d'obtenir els resultats desitjats:

$$\text{Desviació del cost} = (CostEstimat - CostReal) \cdot HoresReals \tag{3.2}$$

$$\text{Desviació del consum} = (HoresEstimades - HoresReals) \cdot CostEstimat \tag{3.3}$$

També s'ha fet servir indicadors per tenir una visió més genèrica del desenvolupament del projecte.

$$\text{Desviació total per tasca} = CostEstimatxTasca - CostFinalTasca \tag{3.4}$$

$$\text{Desviació total recursos} = CostREstimatRecurs - CostFinalRecurs \tag{3.5}$$

Desviació final projecte = PressupostEstimatProj - PressupostFinalProj (3.6)

Es realitzà un seguiment periòdic d'aquests indicadors per tenir un control exhaustiu el temps i recursos disponibles. Tenint en compte el nivell de detall del pressupost estimat, anàlisis dels imprevistos i el percentatge de contingència era molt probable que la desviació acabés essent mínima, com va resultar ser el cas.

3.3 Sostenibilitat i compromís social

Prenent per objectiu l'avaluació i identificació de la sostenibilitat del projecte s'ha fet un anàlisis del seu entorn marcat per tres dimensions: mediambiental, econòmica i social.

Aquest anàlisis es basa i queda plasmat en la matriu de sostenibilitat del projecte representada en la taula 3.8.

	PPP	**Useful life**	**Risks**
Enviromental	Design consumption 8/10	Ecological footprint 19/20	Enviromental risks 0/-20
Economical	Bill 6/10	Viability plan 17/20	Economical risks -2/-20
Social	Personal impact 8/10	Social impact 18/20	Social risks 0/-20
Sustanibility	22/30	52/60	-2/-60
	74/90		

Taula 3.8: Matriu de sostenibilitat del projecte

3.3.1 Dimensió Mediambiental

Aquest projecte precisament està pensat per ajudar al medi ambient doncs mitjançant l'ús d'aquestes tècniques de compressió en xarxes neuronals fa que es requereixi de menys espai d'emmagatzematge en el dispositiu on s'executin, de fer-ho en un hardware dissenyat específicament per aquest tipus de compressió faria que, a més, es requerís de menor potència de còmput i per tant menor consum d'energia comparat amb la que es requereix actualment per l'execució d'una xarxa neuronal. Aquesta millora ajudarà a reduir la contaminació residual (menor emmagatzematge) i elèctrica (menor potència de còmput).

L'empremta ecològica ha estat mínima, ja que desmantellar el projecte no ha produït cap cost addicional, simplement s'ha de desinstal·lat l'entorn i retornat el hardware als respectius propietaris. Una vegada finalitzat el projecte i penjat a Internet es possible continuar-lo molt més a fons, doncs al ser un treball de final de grau no es disposa de més temps per suficient com per explorar l'idea completament ja que existeixen moltes extensions d'aquest treball que, investigades correctament, podrien millorar els resultats obtinguts i, en conseqüència, la seva eficiència mediambiental.

No existeix cap risc mediambiental respecte aquest projecte ja que en l'hipotètic cas de que no hagués produït resultats suficientment bons com per millorar les xarxes neuronals actuals, aquestes produirien com a mínim el mateix efecte mediambiental i almenys s'hagués vist perquè aquest no és el camí a seguir.

3.3.2 Dimensió Econòmica

S'ha realitzat un quantificació detallada de tots els costos que hi haurà, diferenciant-los segons cada tipus de cost i afegint possibles desviacions i imprevists d'aquests. Després d'aquest estudi es creu que el projecte disposa de suficient grau de viabilitat, sempre tenint en compte possibles riscs que poden alterar la planificació estimada.

El projecte pot resultar costós doncs s'ha de tenir en compte que dins al total d'hores requerides per el desenvolupador hi va haver hores dedicades al estudi del tema a tractar, en cas d'usar un professional en aquest àmbit les hores es veurien reduïdes i el pressupost molt més rebaixat. També s'ha d'afegir que el MinoTaure [9] i el CTE-POWER [10] són un recursos molt cars i prescindibles, en segons quins casos.

3.3.3 Dimensió Social

La creació d'aquest projecte ha suposat una millora a nivell professional del desenvolupador doncs com s'ha mencionat amb anterioritat aquest és un tema molt actual i demanat per les empreses, per tant l'experiència adquirida en la realització d'aquesta recerca serà molt útil en la vida laboral del desenvolupador.

L'implementació d'aquesta tecnologia roman invisible pels usuaris malgrat els podria facilitar la vida ajudant a que puguin disposar de xarxes neuronals de manera més econòmica. Al ser un projecte d'investigació, en el cas de que una empresa volgués usar-lo com a punt de partida per la seva idea de negoci, aquesta, ajudaria a la creació de llocs de feina, el mateix per si un investigador volgués aprofundir sobre aquest camp, els resultats d'aquest projecte podrien servir-l'hi com a punt inicial.

No existeix cap col·lectiu que pugui beneficiar-se negativament d'aquesta investigació, doncs en el cas d'haver anat malament i les idees proposades no funcionessin de la forma desitjada, en el pitjor dels casos ens haguéssim quedat com estem ara, sense cap afectació negativa. Com la implementació funciona significa una millora respecte l'eficiència de les xarxes neuronals, així que tampoc té cap afectació negativa.

Capítol 4

Lleis i Regulacions

En aquest capítol s'ha tractat les diferents lleis i regulacions que afecten o estan directament implicades amb el projecte.

4.1 Codi obert

Tal com diu el títol de la secció, Caffe és un framework de codi obert situat a la plataforma de Github [16]. El simple fet d'estar penjat públicament en una plataforma de codi no ens dóna dret automàtic sobre aquest i ens eximeix de copyright a l'hora d'usar-lo per al nostre benefici, com per exemple en la realització d'aquest projecte. Per poder utilitzar-lo amb garantia i sense problemes de copyright s'ha necessitat una llicència de codi obert[29] la qual ens l'haurà de proporcionar l'individu o col·lectiu responsable d'aquest.

Caffe és propietat de l'universitat de California, Berkeley (EUA). Ells mateixos ens proporcionen l'esmentada llicència de codi obert en el mateix github, la qual ens fan acceptar per tal de poder usar-lo. Aquesta llicència diu:

- Qualsevol usuari anònim que faci una modificació al software i vulgui que aquesta tingui un tracte diferent en l'àmbit de copyright ho haurà d'especificar concretament a l'hora de pujar la modificació, si no automàticament tindrà el mateix tracte que la resta de codi.

- Només es pot redistribuir, amb o sense modificacions, sempre i quan mantingui l'avís de copyright adjuntat per ells en el github.

- Es protegeixen sobre possibles mal usos o mal funcionament del seu producte per particulars i/o empreses, eximint-se de qualsevol responsabilitat que això pugui comportar.

4.2 Conjunts públics d'entrenament

Per tal d'entrenar les xarxes neuronals es requerirà dos conjunts públics d'entrenament, CIFAR-10 i MNIST. Per poder fer-ne ús, ambdós, et fan acceptar una llicència d'investigació on s'asseguren que no usaràs aquests conjunts per propòsits comercials, únicament per fer recerca o propòsits educacionals.

CIFAR-10 pertany a *Canadian Institute For Advanced Research* (CIFAR són les sigles d'aquest institut de recerca) i MNIST pertany al *Modified National Institute of Standards and Technology*, que alhora són les seves sigles. Ambdós són col·leccions d'imatges, unes per entrenament i altres per testejar, categoritzades en 10 classes per CIFAR per ambdós.

4.3 Xarxes neuronals

Entrenar una xarxa neuronal per jugar a Go! [30], per exemple, no té cap implicació legal aparent més que pura investigació al respecte, però quan es comença a fer servir al dia a dia de les persones, afectant en la seva vida, comença a ser un problema legal doncs el/s dissenyadors/s de la xarxa han d'acceptar prendre responsabilitat per l'actuació d'aquesta.

La *EU General Data Protection Regulation*[31] (GDPR) és un reglament europeu relatiu a la protecció de les persones físiques en el que respecta al tractament de les seves dades personals i a la lliure circulació d'aquests. L'article 13(2)[32] punt f i 14(2)[33] punt g recullen lleis que afecten directament a les xarxes neuronals, concretament obliguen a la persona o grup de persones que apliquin intel·ligència artificial en individus a explicar la lògica que s'ha aplicat per arribar a la decisió que ha pres el programa en un tribunal, de ser necessari. De totes les tècniques d'aprenentatge automàtic, les xarxes neuronals artificials són de les més costoses de depurar o explicar, resulta realment complex dir com aprèn o justificar la forma com ho fa, per aquest motiu les empreses són bastant reticents a incorporar-les en els seus productes com a única tècnica d'aprenentatge automàtic.

Aquest problema afecta directament al projecte que s'està desenvolupant doncs la compressió de la xarxa afecta directament a la seva precisió, així que si es fes ús de les tècniques proposades i, a causa d'aquesta pèrdua de precisió, hi hagués algun problema que fes perillar la vida d'una persona (conducció autònoma, per exemple) els resultats serien molt problemàtics. Per tant s'haurà de remarcar en la memòria final que només es podrà fer ús d'aquestes tècniques de manera experimental i sense propòsits de comercialització.

Capítol 5

Introducció i entorn de treball

Aquest apartat serveix d'introducció a la part més teòrica que es tracta al llarg de la memòria, les xarxes neuronals artificials, a les quals s'hi podrà referir amb els acrònims NN (*Neural network*), ANN (*Artificial Neural Networks*), DNN (*Deep Neural Networks*) o CNN (*Convolutional Neural Networks*). En aquest capítol s'ha explicat en detall els diferents components d'aquestes: neurones, tipus de capes, tipus de xarxes i com s'entrenen.

A més també s'ha descrit l'entorn de treball que s'utilitzà per fer el projecte, descrivint els diferents conjunts de dades d'entrenament i les xarxes utilitzades per extreure els resultats. S'ha dedicat una secció d'aquest capítol per tractar l'estat de l'art actual.

5.1 Introducció a les Xarxes Neuronals Profundes

Com s'ha introduït anteriorment, les xarxes neuronals artificials són sistemes computacionals que s'inspiren en les xarxes neuronals biològiques presents en els cervells dels éssers vius, com a tals intenten emular la forma en com el nostre cervell aprèn perquè aquesta intel·ligència artificial pugui arribar a fer-ho també sense necessitat de programar-ho directament.

5.1.1 Neurones Artificials

Una xarxa neuronal artificial consta d'una col·lecció de neurones artificials, també anomenades nodes, interconnectades entre si. En la fórmula (5.1) es mostra el càlcul de la sortida d'una neurona, tal com s'observa aquestes reben un conjunt d'entrades, que tant poden venir donades per la sortida d'altres nodes com per fonts externes. Cadascuna d'aquestes entrades té associat un pes (*weight*),

inicialitzat aleatòriament, el qual es va adaptant a mesura que s'entrena la xarxa perquè obtingui millor precisió. Aquest pes marca la importància que té el node d'on prové, a major és el seu valor major importància respecte als altres nodes amb menors valors. Els pesos poden ser positius o negatius, en el primer cas farà que la sortida de la neurona tingui més rellevància i en segon produirà l'efecte contrari, els pesos amb valors propers a zero signifiquen que l'entrada associada a aquest no modificarà la sortida de la neurona.

Hi ha un paràmetre anomenat bias en l'equació, aquest sempre tindrà valor d'entrada 1 i el seu pes propi associat, el qual serà el mateix per tots els elements de la matriu de pesos, generalment. Serveix com a valor extra d'entrada pel càlcul de la sortida del node i s'assegura que, malgrat que la resta d'inputs donats siguin zero, hi hagi almenys un valor constant per l'entrenament.

$$Y = \sum(weight \cdot input) + bias \tag{5.1}$$

Al resultat obtingut en la fórmula (5.1) s'hi aplicarà una funció (no lineal) d'activació. El seu propòsit és introduir no linealitat a la sortida de la neurona, és important fer-ho doncs la majoria de dades en el món real no hosón i volem que la xarxa neuronal artificial les pugui representar amb la millor precisió possible. Existeixen diferents funcions d'activació, entre les més usades tenim: Sigmoid, ReLU i TanH, exemplificades en la figura 5.1.

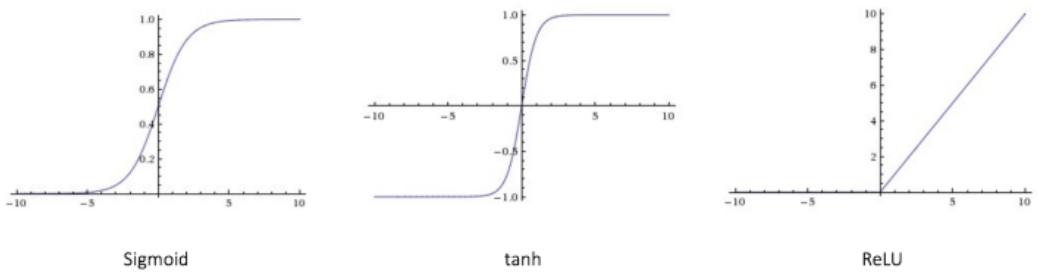

Figura 5.1: Exemples funció d'activació

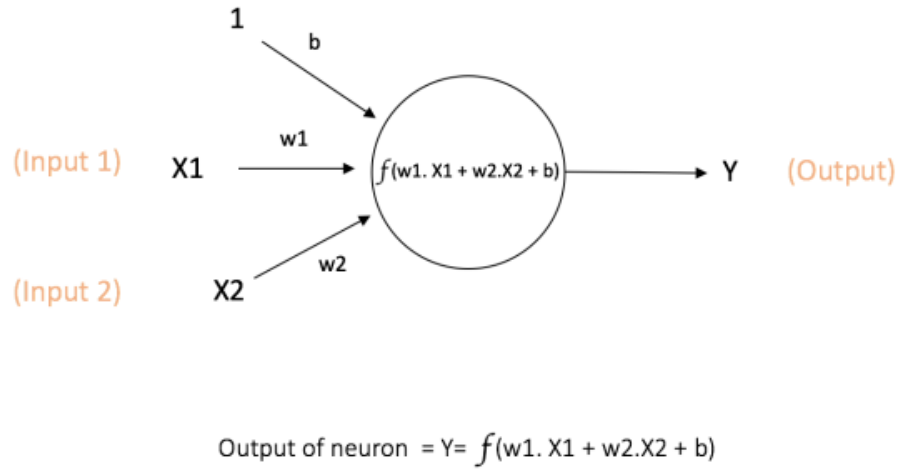

Figura 5.2: Sortida d'una neurona amb funció d'activació

5.1.2 Tipus de Capes

Les neurones, o nodes, estan organitzades en diferents tipus de capes que poden estar connectades entre si. Els tres tipus més genèrics de capes són: *input layer*, *hidden layer* i *output layer*. Les dades travessen la xarxa del *input layer* fins al *output layer*, passant per les *hidden layer* d'haver-n'hi alguna.

- L'**input layer** és, generalment, la capa inicial d'una xarxa neuronal artificial. Són les encarregades de portar les dades d'entrada inicials al seu processament dins la xarxa.

- La **hidden layer** són el conjunt de capes entre la d'entrada i sortida on es simula el conjunt d'activitats que succeeix al cervell per tal de produir el resultat esperat. Les connexions entre els nodes d'aquesta capa pot ser de molt divers.

- L'**output layer** és l'última capa de neurones encarregades de produir el resultat de la xarxa neuronal artificial i avaluar la seva precisió i error d'entrenament.

Existeixen molts tipus de connexions entre les diferents capes d'una xarxa neuronal, aquesta combinació entre les neurones fa que es pugui aproximar funcions que d'altra manera serien molt complexes (o impossibles) de programar, com el reconeixement d'imatges. A continuació es farà una breu introducció dels tipus que es tractaran durant la realització d'aquest projecte. És important remarcar que existeixen altres tipus de connexions que no es mencionaran en aquest document

i no han sigut provades en el projecte, però en alguns casos es podrien arribar a beneficiar d'ell.

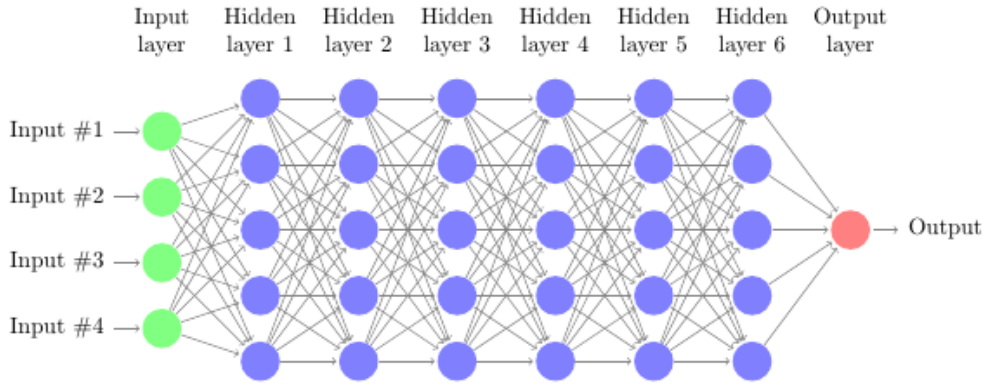

Figura 5.3: Exemple de *input*, *hidden* i *output* layer en una ANN

5.1.2.1 Capa *Fully-Connected*

En les capes *fully-connected*, o densament connectades, cada neurona està associada amb la resta de les anteriors capes, després de que s'hagi aplicat la funció d'activació en aquestes últimes. Un exemple el podem trobar en la figura 5.3 en la qual totes les neurones estan densament connectades.

Tal com s'ha vist en la fórmula (5.1), el càlcul de les neurones d'aquesta capa es pot entendre com un producte.

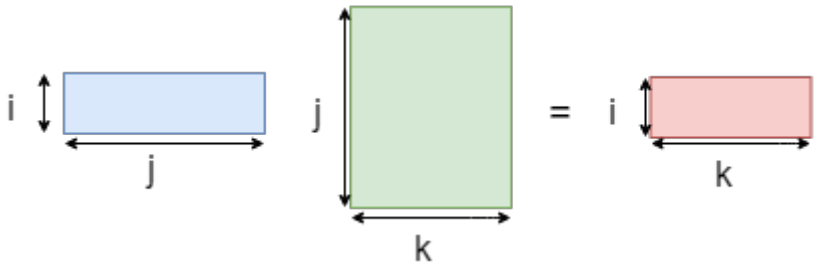

Figura 5.4: Exemple del producte entre dos capes *fully-connected*

La i de la figura 5.4 és el nombre total d'inputs agrupats amb els que es recorre conjuntament la xarxa, el que es coneix en termes de *deep learning* com batch i la k el nombre de nodes que conté la capa. En el cas de que el batch sigui 1 el que acabarà havent-hi essencialment és un producte de vector per matriu, on

el coll d'ampolla clarament està marcat per aquesta matriu de pesos de j x k, que generalment sol ser de mida molt gran. Bé és cert que en alguns casos la mida del batch no és 1 sinó un nombre superior, fent que la i ja no sigui un únic valor i convertint aquesta operació en un producte de matriu per matriu, però majoritàriament aquest fet es produeix durant l'entrenament de la matriu i no una vegada aquesta ha sigut entrenada, que és on nosaltres buscarem l'optimització.

Aquesta és la matriu de pesos a la que es fa referència al llarg de tota la memòria i a la que s'hi ha aplicat les diferents tècniques de compressió durant la realització d'aquest projecte.

5.1.2.2 Capa convolucional

Aquest tipus de capa està present en les xarxes neuronals convolucionals, també nombrades amb l'acrònim CNN de la paraula anglesa *Convolucional Neural Network*, les quals estan pensades essencialment pel reconeixement d'imatges. A diferència de la capa densament connectada, els nodes d'aquesta només es connectaran a una regió de la següent capa, donant per sortida el que s'anomena mapa de característiques o imatge convolucionada.

Parlarem d'aquesta capa en el cas d'estar sent usada amb imatges, reben com a entrada una matriu n_w x n_h x d, on n_w és la mida de l'amplada i n_h altura de l'imatge en píxels i d la seva profunditat. Aquesta contindrà k nuclis, també nombrats *kernels*, de mida m_w x m_h x $d2$, on aquestes mida m_w i m_h són l'amplada i alçada d'aquest nucli respectivament. S'haurà de respectar que $m_w \leq n_w$, $m_h \leq n_h$ i $d2$, essent la profunditat del nucli, també haurà de respectar que $d2 \leq d$. A continuació s'adjuntarà la fórmula pel càlcul entre l'imatge I (primera matriu) i el kernel K (en color vermell), obtenint l'imatge convolucionada $I * K$, representada gràficament en la figura següent.

$$(I \cdot K)_{xy} = \sum_{i=0}^{m_w} \sum_{j=0}^{m_h} K_{ij} \cdot I_{x+i-1,y+j-1} \tag{5.2}$$

S'ha explicat molt generalment aquest tipus de capa, això és degut a que només ens interessa saber de la seva existència perquè s'utilitzaran xarxes neuronals convolucionals per aquest projecte, però en cap cas es modificarà aquestes capes ni el seu contingut.

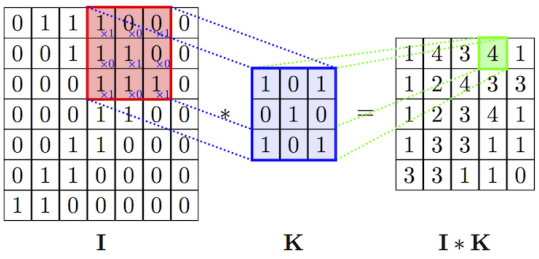

Figura 5.5

5.1.2.3 *Pooling layer*

La seva traducció al català seria capa d'agrupació, però és mantindrà el seu nom en anglès doncs generalment és molt més usat. Aquest tipus de capes també són presents en les xarxes neuronals convolucionals.

Aquesta capa s'insereix entre successives capes convolucionals i la seva funcionalitat és la d'anar reduint periòdicament la mida de la representació per així reduir la quantitat de paràmetres en la xarxa, fer el seu còmput més ràpid i prevenir la sobrecàrrega.

En aquesta ens succeeix el mateix que en l'anterior, al no realitzar-li cap modificació s'ha fet una explicació breu per entendre-les, ja que s'utilitzaran en les CNN.

5.1.3 Tipus de Xarxes Neuronals

A continuació es detallen diferents tipus de xarxes neuronals enterament relacionades amb aquest projecte, partint de les més antigues, i en alguns casos bàsiques, les quals no s'ha tractat en aquest projecte doncs en alguns casos estan obsoletes per acabar amb els més "complexos", dels quals únicament s'ha explicat els que s'utilitzaran per a aquest treball. Cal remarcar que existeixen molts més tipus de capes que no s'han explicat en aquest projecte doncs no es tractaren ni hi estan estretament vinculades.

5.1.3.1 Xarxa neuronal directa

Molt més coneguda per el seu nom anglès *Feed Forward Neural Network* (FFN), són tipus de xarxes neuronals bastant antigues, daten de la dècada dels anys 50, van ser les primeres i més senzilles xarxes creades. Les neurones es connecten consecutivament d'una capa a una altre sense crear cicles. Existeixen diferents tipus de *Feed Forward Neural Network*:

- **Perceptró d'una capa:** També conegut com *Single-Layer Perceptron* (SLP). Aquesta és considerada com la forma més senzilla i antiga de xarxa neuronal, format únicament per una neurona d'una sola capa, sense hidden layers. El seu funcionament també és molt bàsic, suma el conjunt d'inputs, aplica una funció d'activació i ho passa a l'output layer.

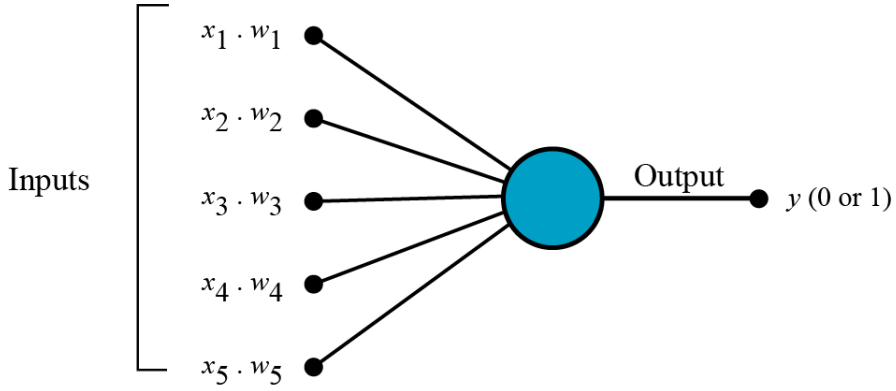

Figura 5.6: Perceptró

- **Perceptró multicapa:** Conegut també com a *Multi-Layer Perceptron* (MLP) o *Deep Feed Forwad Network* (DFF). En aquest model pot haver-hi d'una a moltes *hidden layers* entre la capa d'entrada i la de sortida. Inicialment es va crear amb una única capa oculta, però en la dècada dels anys 90 es van començar a realitzar les primeres investigacions afegint-hi indefinides *hidden layers* i comprovant que, efectivament, augmentava en gran número la precisió de la xarxa. Els nodes estan totalment densament connectats (*fully-connected*) entre si entre l'input i la hidden layer.

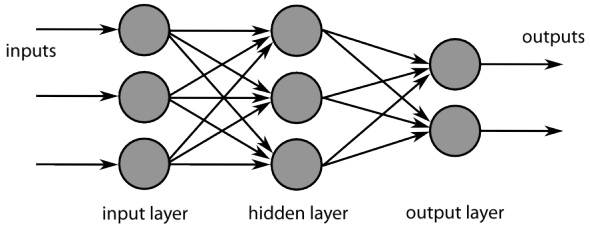

Figura 5.7: *Single-Layer Perceptron (SLP)*

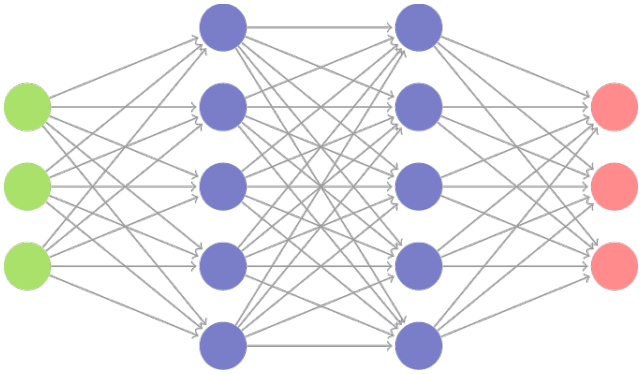

Figura 5.8: *Multi-Layer Perceptron (MLP)* amb més d'una capa

5.1.3.2 Deep Convolutional Networks

Actualment aquestes xarxes són les més usades per investigació relacionada amb *"Deep Learning"*, bona prova n'és el fet que la gran majoria de xarxes publiques per fer recerca són d'aquest tipus, fet que ha provocat que per la realització del projecte s'hagin hagut d'utilitzar malgrat no siguin les que més se'n poden beneficiar ja que únicament disposen d'una petita quantitat de capes *fully-connected*.

Els nodes convolucionals processen les dades d'entrada i els *pooling layers* la simplifiquen, reduint així les característiques innecessàries, tal com s'ha explicat en els anteriors apartats. El seu principal ús és el reconeixement d'imatges. La xarxa acaba amb un MLP com el que podem veure en la figura 5.8.

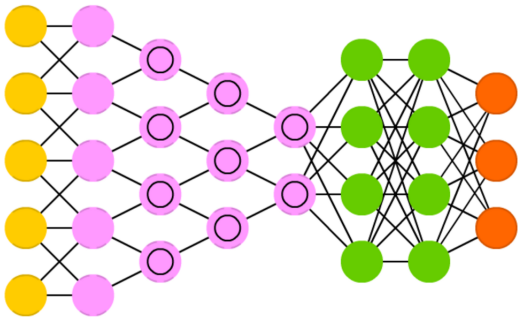

Figura 5.9: Deep Convolutional neural network (DCN)

5.1.4 Entrenament d'una xarxa neuronal

A l'hora de parlar sobre com s'entrena una xarxa s'ha de fer una breu descripció sobre diferents conceptes relacionats amb el tractament del conjunt de dades d'entrada, el qual sol ser molt gran generalment.

- **Mostra:** Es genera a partir del conjunt d'entrada i conté les dades que s'utilitzaran per entrenar la xarxa i el resultat que s'espera obtenir amb aquestes dades. Un conjunt d'entrenament està format per moltes mostres.
- **Batch:** És un paràmetre que marca el total de mostres amb les quals s'entrena la xarxa abans d'actualitzar els seus valors interns. El conjunt d'entrenament pot estar dividit per un o més batch.
- **Epoch:** El número d'epoch's defineix el total de vegades que la xarxa neuronal recorrerà tot el conjunt d'entrenament. Es marca aquesta mesura temporal per anar provant la precisió que va aconseguint durant la xarxa en l'entrenament.

Una vegada estan definits aquests conceptes podem entrar en com funciona l'entrenament en les xarxes neuronals, tal com es mostra en la figura d'exemple. L'entrenament està dividit en dos fases: el *forward propagation* i el *backward propagation*, les quals s'aniran repetint per cada epoch.

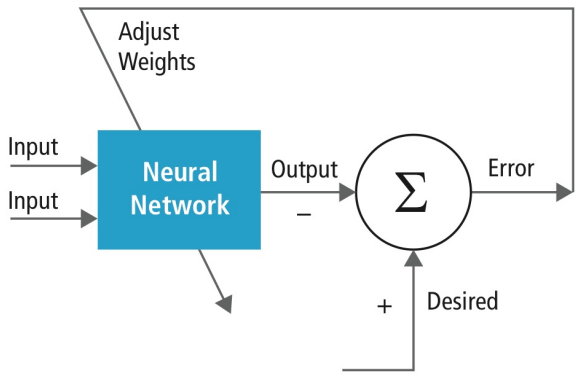

Figura 5.10: Entrenament de la xarxa neuronal

En la primera fase la xarxa rep un batch de mostres del conjunt d'entrenament, aquestes la recorreran i per cada neurona de cada capa s'aplica la fórmula (5.1) i una funció d'activació, tal com s'ha explicat amb anterioritat. La xarxa generarà una sortida a cada batch que s'utilitzarà en la fase de *backward propagation*.

En la fase de *backward proparation* disposarem del output generat per la nostra xarxa neuronal artificial durant la fase de *forward propagation* i, al conèixer el conjunt d'entrenament, la sortida que desitjaríem que hagués tingut. Per la combinació d'aquests dos es crearà una funció nombrada de pèrdua o cost, la qual bàsicament ens ajuda a saber com de bé està funcionant la xarxa a l'hora de predir els resultats esperats. Es buscarà minimitzar aquesta funció de pèrdua, doncs com menor sigui menys diferència hi haurà entre els resultats obtinguts i els esperats i per tant millor funcionarà la xarxa neuronal. Existeixen diferents tècniques per fer-ho, la més usada generalment és la que s'anomena *gradient descent*. Aquesta funciona canviant els pesos en petits increments després de cada batch, al calcular el gradient de la funció de cost en un determinat conjunt de pesos, som capaços de veure en quina direcció es troba el mínim. En la figura següent s'exemplificarà gràficament.

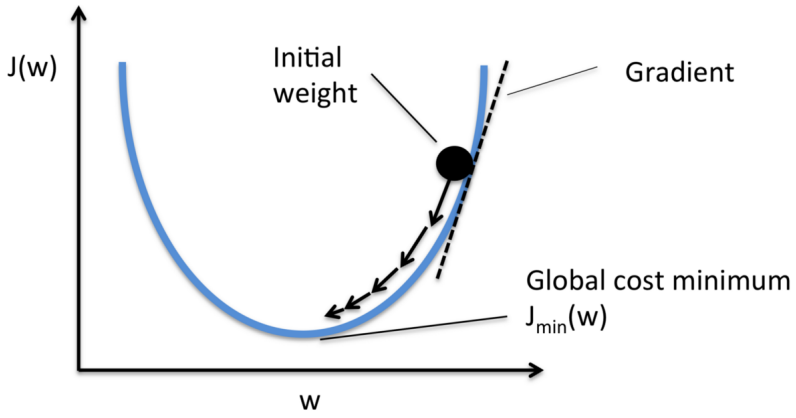

Figura 5.11: Exemple de la tècnica *Gradient Descent*

5.2 Entorn de treball i conjunt d'experimentació

Aquesta secció serveix per explicar l'entorn de treball utilitzat: els conjunts d'experimentació i xarxes neuronals fetes servir per extreure'n els resultats de la investigació.

5.2.1 Conjunts de dades utilitzats

Tal com s'ha explicat anteriorment per tal de realitzar l'entrenament d'una xarxa neuronal es necessari disposar d'un conjunt de dades que alhora contingui dos subconjunts: un per entrenament i l'altre per validació. Quan al llarg del treball es faci referència, sobretot als resultats, a les precisions obtingudes per les xarxes d'exemple, sempre serà l'extreta del conjunt de validació doncs és la que ens interessa per provar la utilitat de les tècniques de compressió implementades

Al estar tractant amb xarxes neuronals convolucionals s'ha buscat utilitzar dos famosos conjunts d'imatges: MNIST i CIFAR-10.

5.2.1.1 MNIST

MNIST, acrònim per *Modified National Institute of Standards and Technology*, és un conjunt de dades d'imatges de dígits escrits a mà utilitzat per l'entrenament de programes amb patrons de reconeixements d'imatge, com és el cas d'aquest projecte doncs s'usarà per entrenar xarxes neuronals artificials.

La seva base de dades està composta per 60.000 imatges, de *28x28* píxels, d'entrenament i 10.000 imatges, també de *28x28* píxels, de validació, aquesta forma part d'un subconjunt de la base de dades NIST [34]. Cadascuna d'aquestes imatges es pot classificar en 10 classes diferents corresponents als dígits del 0 al 9.

La pàgina web oficial on es pot descarregar aquesta base de dades d'imatges és `http://yann.lecun.com/exdb/mnist/`, d'on s'ha extret per la realització del projecte.

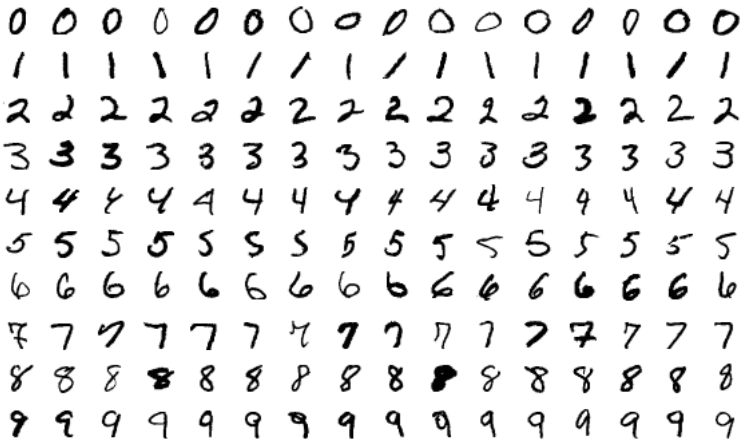

Figura 5.12: Exemple del conjunt de dades MNIST

5.2.1.2 CIFAR-10

La base de dades CIFAR-10 és una col·lecció d'imatges pensada per utilitzar durant l'entrenament de programes d'aprenentatge automàtic i reconeixement d'imatges. CIFAR és l'acrònim de *Canadian Institute For Advanced Research*, institut al que pertany aquesta base de dades. El número 10 final significa el total de classes amb les quals es poden dividir les imatges, aquestes engloben tant vehicles de transport com animals: avió, cotxe, ocell, gat, ren, gos, granota, cavall, vaixell i camió.

Aquesta base de dades consisteix en 60.000 imatges en color de mida 32x32 píxels dividides en 10 classes amb 6.000 imatges per cada classe. Del total d'imatges 50.000 es fan servir per entrenament i 10.000 per validació.

La pàgina web oficial on es pot descarregar aquesta base de dades d'imatges és https://www.cs.toronto.edu/~kriz/cifar.html, d'on s'ha extret per la realització del projecte.

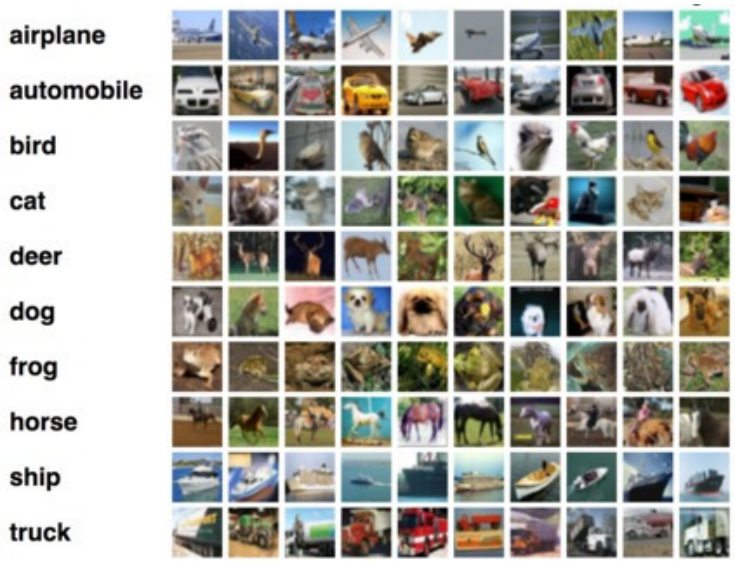

Figura 5.13: Exemple del conjunt de dades CIFAR-10

5.2.2 Xarxes neuronals utilitzades

En aquesta part s'especifica les xarxes neuronals que s'han utilitzat per provar la funcionalitat de l'idea i la consegüent extracció de resultats. Tots els resultats obtinguts amb aqueses s'han provat amb la mateixa quantitat d'epochs fixada des d'un inici, 120.

S'ha utilitzat tres xarxes neuronals d'exemple, una pel conjunt de dades MNIST i dos per CIFAR-10. Se les ha ordenat per mida de matriu de pesos.

5.2.2.1 LeNet

Aquesta xarxa neuronal està basada en l'explicada en el paper citat a continuació [35]. S'ha adjuntat un exemple gràfic de cada capa continguda en la xarxa i les seves connexions entre si. Només s'utilitzarà en el conjunt MNIST.

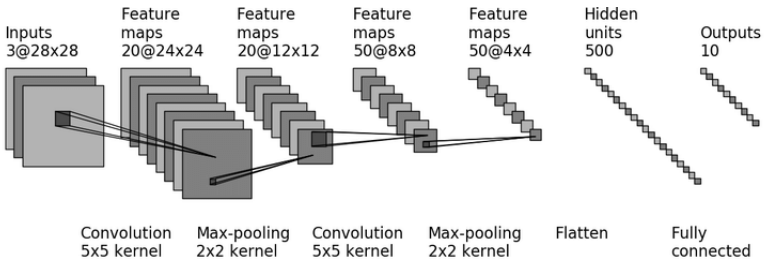

Figura 5.14: Xarxa neuronal convolucional LeNet

La part a comprimir d'aquesta matriu serà la seva penúltima capa, formada per una matriu de neurones de mida *800x500*, ocupant aproximadament 1,5 megabytes d'emmagatzematge, la més petita usada en el projecte. Com es pot observar la xarxa conté 6 capes ocultes: 2 convolucionals, 2 *pooling* i 2 densament connectades.

1. Convolucional de 20 amb kernels de *5x5*.
2. Seguida d'una de *pooling* amb un kernel de *2x2*.
3. També convolucional de 20 amb un kernel de *5x5*.
4. *Pooling* amb un kernel de *2x2*.
5. Densament connectada amb una mida de matriu de pesos de *800x500* sent aquesta la que tractarem de comprimir.
6. Última densament connectada de *500x10* sent aquest 10 el total de classes de MNIST.

Aquesta xarxa ha estat entrenada amb una taxa d'aprenentatge de *0.01* fixe, un *momentum* de *0.9* i una mida de batch de 100.

La precisió obtinguda per aquesta xarxa, sense haver aplicat cap modificació, després de 120 epochs s'adjuntarà seguidament. Com es pot comprovar la seva precisió augmenta de manera molt ràpida i és molt alta, entre el 100% i el 99%.

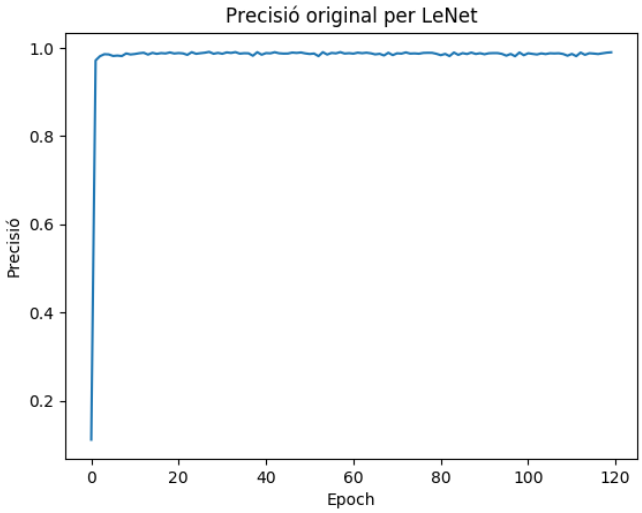

Figura 5.15: Precisió de LeNet Original

5.2.2.2 CIFAR-10 quick

Aquesta xarxa ha sigut extreta dins les mostres d'exemples que et proporcional Caffe [36], s'ha modificat lleugerament perquè tingués un capa més que fos densament connectada de *500x1024* intentant alterar molt poc la precisió que tenia l'original.

La capa que ens hem centrar a comprimir és la penúltima, de mida *500x1024*, ocupant aproximadament un 2 megabytes d'emmagatzematge. Contindrà 8 capes ocultes: 3 convolucionals, 3 *pooling* (les quals totes seran amb un kernel de *3x3*) i 2 densament connectades.

1. És convolucional de 32 amb un kernel de *5x5*.
2. Capa de *pooling*.
3. Convolucional amb els mateixos paràmetres que la primera.
4. De *pooling* igual que la segona.
5. Convolucional de 64 amb un kernel de *5x5*.
6. Capa de *pooling* com la segona i quarta.
7. Capa densament connectada amb una matriu de pesos de *500x1024* que es buscarà comprimir.
8. Densament connectada de *500x10* on el 10 és el total de classes de CIFAR-10 per fer la classificació.

Aquesta xarxa ha estat entrenada amb una taxa d'aprenentatge de *0.001* fixe, un *momentum* de *0.9* i una mida de batch de 100.

La precisió obtinguda per aquesta xarxa, sense haver aplicat cap modificació, després de 120 epochs és casi al 80%.

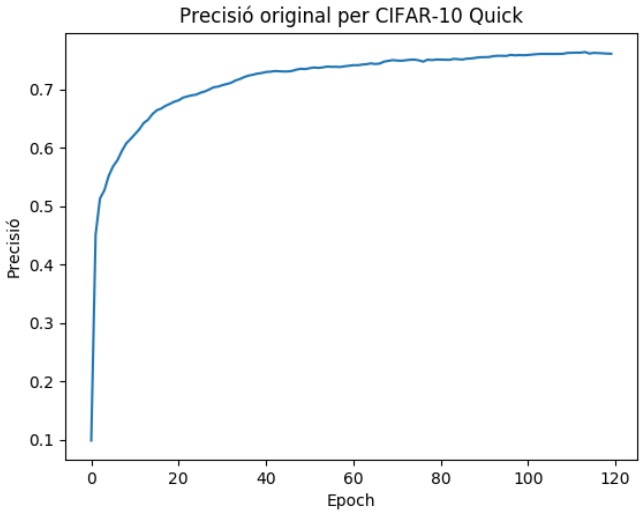

Figura 5.16: Precisió de CIFAR-10 Quick original

5.2.2.3 AlexNet

Aquesta xarxa neuronal ha volgut estar basada en l'explicada en el paper [37] però l'original estava pensada per utilitzar-se en el conjunt d'imatges *ImageNet* [38], que són de mida *256x256* píxels mentre que, com s'ha explicat, les de CIFAR-10 són de *32x32*. Partint d'aquest punt se li ha hagut d'aplicar diferents modificacions per poder usar-la correctament ja que de no ser així entre les capes convolucionals i *pooling* acabarien reduint les dades més del compte i no funcionaria.

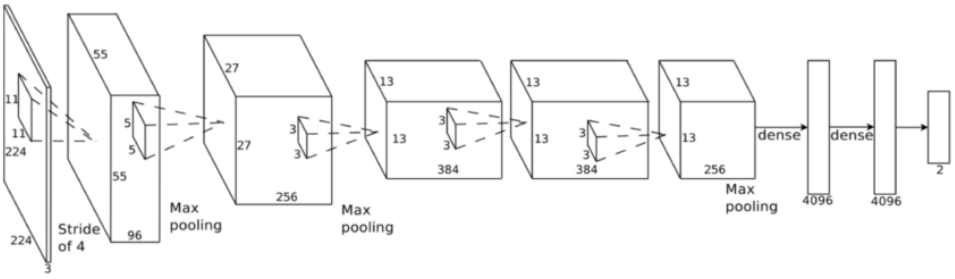

Figura 5.17: Xarxa neuronal convolucional AlexNet (**Original**)

S'ha comprimit la segona i tercera capa començant pel final, de mides 4096x2048 i 2048x1048, respectivament, i ocupant entre les dos un total aproximat de 40 megabytes d'emmagatzematge. La xarxa contindrà 10 capes ocultes: 4 convolucionals, 3 *pooling* (les quals totes seran amb un kernel de *3x3*) i 3 densament connectades, s'ha eliminat una capa convolucional respecte l'original.

1. Convolucional de 32 amb un kernel de *11x11*.
2. Capa de *pooling*.
3. Capa convolucional de 32 amb un kernel de *5x5*.
4. Capa de *pooling*.
5. Capa convolucional de 64 amb un kernel de *3x3*.
6. Idèntica a la cinquena.
7. Capa de *pooling*.
8. Capa densament connectada amb una matriu de *4096x2048*, ideal per aplicar els mètodes d'aquest projecte.
9. Capa densament connectada, una mica menor a l'anterior però igualment vàlida per aplicar-hi tècniques de compressió. Matriu de pesos de *2048x1024*.
10. Capa densament connectada final utilitzada per classificació. Matriu de pesos de mida *1024x10* on el 10 és el total de classes de CIFAR-10.

Aquesta xarxa ha estat entrenada amb una taxa d'aprenentatge de *0.001* fixe, un *momentum* de *0.9* i una mida de batch de 200 per optimitzar millor al temps ja que al ser la xarxa més grossa tardava significativament més a entrenar. La precisió obtinguda per aquesta xarxa, sense haver aplicat cap modificació, després de 120 epochs, sobrepassarà per poc el 80%.

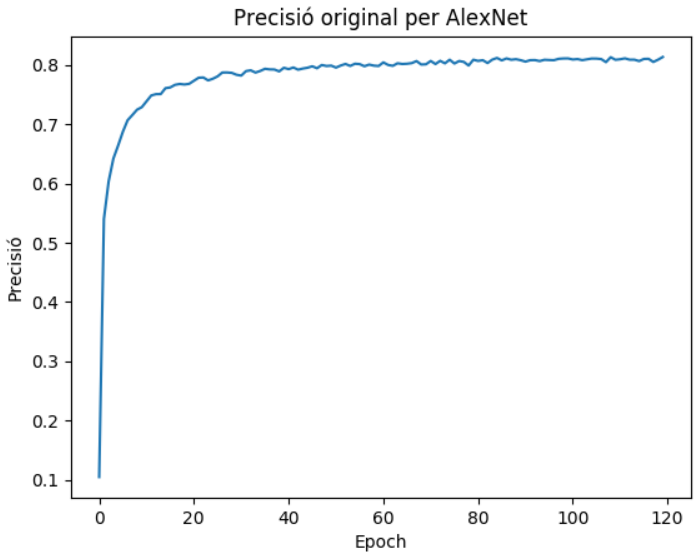

Figura 5.18: Precisió de AlexNet Original

5.3 Estat de l'Art

Com s'ha comentat en el context, aquest és un tòpic molt popular en investigació actualment, per tant existeixen molts articles al respecte, molts d'aquests estan centrats a afrontar el mateix tòpic que s'ha investigat, però des de punts de vista una mica diferents. En la gran majoria d'ells, al ser les xarxes neuronals artificials de forma error-tolerant [39], acaben aconseguint una millora significativa de rendiment sacrificant molt poc augment de l'error d'entrenament, el qual en el seu moment ens va motivar a pensar que l'idea inicial podria funcionar, com ha acabat passant.

Alguns d'ells [40] se centren a intentar reduir la mida bits de les operacions i passar-los de l'aritmètica de punt de flotant de 32 bits usada habitualment en xarxes neuronals a aritmètiques de: punt fix de 12 bits i punt flotant de 12 bits. Aquest article no acaba comprimint la matriu d'una forma similar a com s'ha fet en aquesta memòria sinó que ho fa modificant la representació dels tipus de dades, malgrat això no són excloents i ambdós projectes es podrien intentar fusionar a fi d'aconseguir una compressió encara més alta.

D'altres articles [41] opten per *"podar"* (eliminar) connexions entre les diferents matrius de pesos que tinguin valor zero o molt pròxims a ell, ja que no modifiquen la sortida de la xarxa. D'aquesta manera es pot aconseguir reduir-ne la matriu

resultant i permetre així una compressió d'aquesta.

Aplicar aquesta *"poda"* conjuntament amb altres factors pot ajudar encara més augmentar la compressió, tal com passa en l'article [42] el qual aplica una compressió de tres passos: primer poda, segon quantifica els pesos i, finalment, hi aplica la codificació de *Huffman* [43]. D'aquesta manera aconsegueix factors de compressió molt alts en xarxes neuronals convolucionals sense alterar-ne la precisió. En el cas d'aquest projecte no s'ha aplicat cap mena de *"poda"* en la matriu, malgrat això s'hi podria aplicar per possibles estudis ja que segons suggereixen aquestes investigacions podria millorar-ne notablement la compressió sense pagar-ho amb la precisió de la xarxa.

S'han proposat un nou estil de xarxes basades en l'ús de funcions de hash [44] a l'hora d'agrupar les connexions entre els diferents pesos. Les *HasedNets*, tal com les anomena el paper [45], utilitzen funcions de hash poc costoses per tal d'agrupar aleatòriament les connexions entre els pesos en el mateix grup, totes les connexions de pesos en un mateix grup es representaran amb un mateix paràmetre, millorant així la representació de la matriu i el cost d'emmagatzematge. En principi aquesta forma seria més complicada d'ajuntar-la amb l'idea que s'ha portat a terme amb el projecte doncs significaria un canvi molt radical en l'estructura de la xarxa i es vol comprimir.

Aquests només han sigut uns exemples de diferents formes d'afrontar el problema plantejat que s'estan intentant actualment, ajuntant els coneixements extrets de la recerca d'aquests articles es pot determinar que el problema, malgrat que s'intenta enfocar de moltes maneres, encara no té una resolució clara. S'ha perseguit un mateix objectiu però des d'un enfocament diferent del s'ha vist en els articles consultats, malgrat que en algun moment s'ha pogut requerir de l'extracció algun coneixement teòric esmentat en algun d'ells, com l'error tolerància de les xarxes [39]. En un futur es podria plantejar l'idea de fusionar alguna d'aquestes recerques amb la portada a terme en aquesta memòria per millorar encara més la compressió aconseguida.

Capítol 6

Proposta i Resultats

En aquest apartat s'explicarà les diferents tècniques de compressió que s'ha aplicat a la matriu de pesos, explicada en la secció 5.1.2.1, durant les diferents etapes d'aquest projecte. Es justificarà: les seves eleccions, beneficis i contres, a més d'adjuntar exemples gràfics d'alguns resultats obtinguts per avaluar-los.

La idea principal fou utilitzar la propietat error-tolerant [39] que tenen les xarxes neuronals artificials per ajudar-nos a comprimir-les. Aquesta propietat ens permet "forçar" que les matrius de pesos tinguin els valors que ens siguin més convenients per comprimir-la, obligant a la xarxa a entrenar-se d'aquesta forma, ja que ella mateixa s'autoregularà i farà que no es generi una pèrdua d'encert notable, si s'aplica correctament. S'ha buscat la manera de minimitzar aquesta possible pèrdua d'encert.

6.1 Compressió durant l'entrenament

En aquest apartat s'explica en quin moment de l'entrenament es realitzarà la compressió i justifica el per què d'aquesta decisió.

Com s'ha explicat es vol aprofitar la propietat error-tolerant [39] de les xarxes neuronals "forçant" que s'entrenin comprimides doncs es creu que, havent proporcionant un mínim de valors per bloc que sigui raonable, no es veurà necessàriament afectada la precisió de la xarxa malgrat la compressió.

Es decidí comprimir la matriu al final del *backward propagation*, just abans d'actualitzar els pesos. Al arribar al últim batch d'entrenament simplement no els actualitza i es deixa la xarxa entrenada amb la matriu comprimida. Les precisions obtingudes per provar la precisió en cada epoch han sigut considerant la matriu

totalment comprimida.

Com la compressió es farà al *backward*, el primer *forward* es fa sense haver aplicat cap modificació, però cal recordar que les xarxes neuronals artificials utilitzades inicialitzen els valors dels seus pesos aleatòriament i a partir de l'entrenament aquests es van ajustant per aconseguir cada vegada millors resultats. Tenint això en consideració no és important que durant primer batch no estigui comprimida.

A continuació s'adjuntarà el codi amb la funció d'entrenament per la xarxa neuronal. La resta es podrà localitzar en l'apèndix.

```
1  def train(iters,print_interval,solver):
2
3    train_loss = [0]*(iters//print_interval)
4    test_acc = [0]*(iters//print_interval)
5
6    for it in range(int(iters)):
7
8      correct = 0
9      loss = 0
10
11     #forward propagation
12     solver.net.forward()
13
14     #backward propagation funcio de cost + compresio matricial
15     solver.net.backward()
16
17     if it % print_interval == 0:
18       for test_it in range(TestSamples/batch_size):
19         solver.test_nets[0].share_with(solver.net)
20         solver.test_nets[0].forward()
21         correct += solver.test_nets[0].blobs['accuracy'].data
22
23       test_acc[it // print_interva] = correct / (TestSamples/batch_size)
24       train_loss[it // print_interval] += solver.net.blobs['loss'].data
25
26     #backward propagation actualitza els pesos
27     solver.apply_update()
28
29   return [train_loss, test_acc]
```

Figura 6.1: Codi per l'entrenament de la xarxa neuronal

Tal com indica el codi adjuntat, per cada batch:

1. Es fa el *forward propagation*, indicat en la línia 12, amb la matriu comprimida, menys el primer batch, i els pesos actualitzats amb el gradient

2. Seguidament es fa el *backward propagation* per actualitzar els valors del gradient i, s'haurà modificat Caffe [12] internament, perquè al final de la mateixa funció es pugui comprimir la matriu. Aquesta funció està situada en la línia 15.

3. Aquest és un pas opcional, indicat dins el condicional que comença a la línia 17, ja que de ser una iteració en l'entrenament coincident amb epoch es farà una prova de la precisió amb la matriu encara comprimida.

4. La funció *applyupdate* de la línia 27 s'ha afegit també internament a Caffe [12] per actualitzar els pesos, un cop acabat aquest pas es tornarà al primer fins fer tots els epoch's especificats.

D'aquesta manera l'entrenament es farà amb un grau de llibertat més alt doncs durant el *forward propagation* la matriu no estarà comprimida en la seva totalitat, ja que s'hauran actualitzat els pesos, però si que forçarà a que aquests siguin significativament similars i al acabar l'execució o a l'hora de provar les precisions per cada epoch si que ho estarà.

6.2 Representació i compressió de la matriu

En aquest apartat s'explica cronològicament les diferents representacions que s'ha aplicat a la matriu per aconseguir comprimir-la, alguns resultats obtinguts i comparacions gràfiques entre ells.

També s'explica la representació en binari de la compressió que s'hauria d'utilitzar en un hipotètic hardware especialitzat per entrenar les xarxes neuronals amb aquestes tècniques i als diferents graus de compressió que es podria arribar amb aquesta.

6.2.1 Divisió per blocs

Es divideix la matriu de pesos, explicada en la secció 5.1.2.1, en blocs quadrats de mides variables n x n, essent aquesta n qualsevol valor $\{2..32\} \in \mathbb{N}$. S'ha de tenir en compte que com més gran sigui el bloc millor compressió es podrà aconseguir.

Un cop s'ha escollit una mida de bloc es mantindrà exclusivament aquesta durant tot l'entrenament. En l'exemple següent mostrarem una matriu de mida 9x9 dividida en blocs de 3x3, tal i com s'ha explicat fins ara.

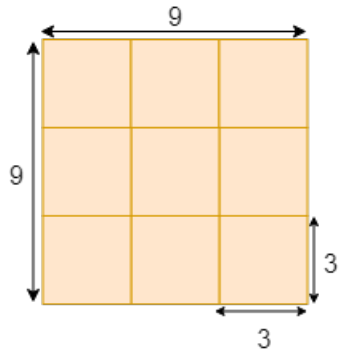

Figura 6.2: Exemple de divisió per blocs de la matriu

Només existeix una excepció per la qual la dimensió dels blocs pot ser diferent en la mateixa matriu, aquest és el cas en que l'alçada i/o amplada de la matriu no sigui múltiple amb la dimensió del bloc. S'exemplifica aquest cas en la figura següent, on hi ha una matriu de *9x10* dividida en blocs de *3x3*. Al tenir un número de columnes (10) no divisible per la mida de bloc (3) s'haurà d'afegir una última columna de blocs de mida 3x1 pintats de color lila en l'exemple.

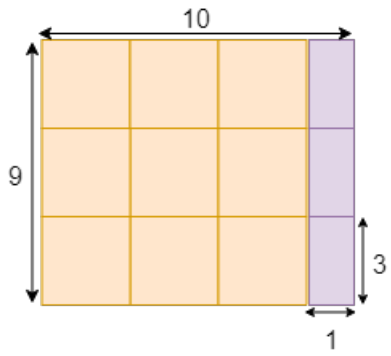

Figura 6.3: Exemple de divisió per blocs no múltiples amb la matriu

Permetre diferents mesures de bloc i no exclusivament la mateixa mida per tots ells podria ser una extensió del projecte que ens permetria donar molta més llibertat a la matriu i, possiblement, obtenir millors resultats.

Caffe utilitza la tecnologia *"Google protobuf"* [46] de manera interna per la declaració de les xarxes neuronals, s'ha hagut de modificar internament per passar

com a paràmetres les diferents mides de bloc, indicant el nombre de files com al paràmetre *rows* i columnes com *cols*, tal i com mostra l'exemple de continuació.

```
net_param{
name: "Alexnet"
      .....
  layer {
    name: "ip1"
    type: "InnerProduct"
    .....
    cols: 16
    rows: 16
    ....
}
```

Figura 6.4: Exemple de d'inicialització per la xarxa AlexNet on es comprimeix la matriu de pesos en blocs de *16x16*

6.2.2 Valors permesos per bloc

En aquesta subsecció s'explica els diferents mètodes que s'ha utilitzat per limitar el total de números i escollir quins deixar-hi en un mateix bloc.

Per tal de poder explicar les diferents tècniques correctament suposarem que s'ha extret d'una matriu de pesos inicialitzats aleatòriament, subdividida per blocs de mida *4x4*, un bloc que s'utilitzarà com a exemple gràfic per cada mètode. Tots els altres blocs pertanyents a aquesta matriu de pesos adoptaran el mateix comportament que l'explicat d'exemple.

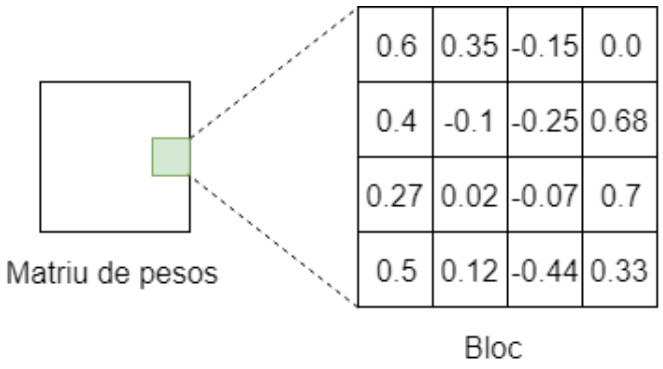

Figura 6.5: Matriu d'exemple

Com a norma s'ha establert que en cadascun d'aquests esmentats blocs només se'ls

permetrà tenir 1, 2, 4, 8, 16 o 32 valors i/o combinacions entre ells; sempre valors múltiples de dos, sense valors intermedis doncs alhora de fer la seva representació en binari no té gaire sentit permetre'n que no sigui cap d'aquests, ja que no s'aconseguiria una compressió notable.

Una altre norma establerta és que cada bloc podrà tenir un màxim de valors diferents igual a la seva mida de files i/o columnes. Donat el cas de la matriu d'exemple, al ser de *4x4* només podrà tenir 1, 2 o 4 valors diferents. Idíl·licament es voldrà que tinguin com menys valors possibles doncs així la compressió també serà més alta.

Cada bloc té un total de valors inicials iguals a $NF \cdot NC$ on NF i NC són el número de files i columnes d'aquest, respectivament. En la matriu d'exemple hi ha un total de 16 valors fruit del producte entre $4 \cdot 4$.

6.2.2.1 Mitja aritmètica

Una forma dràstica de reduir els valors dels blocs és aplicar la mitjana aritmètica, per la qual també ens podrem referir en el seu acrònim "MA" al llarg de la memòria, en cadascun d'ells. La fórmula per un conjunt de valors X de mida N seria l'adjuntada a continuació.

$$MitjanaAritmtica(X) = \frac{\sum_{i=0}^{N} X_i}{N} \tag{6.1}$$

Aquesta és considerada dràstica doncs es passaria a obtenir un únic valor per bloc limitant molt la "llibertat" dels pesos i, conseqüentment, la precisió final obtinguda en la xarxa. Per exemple una matriu de *2048x2048* dividida en blocs de *8x8* passaria a tenir 64 vegades menys valors, de 4194304 inicialment a 65536 amb aquest mètode.

Malgrat això s'obtindrien factors de compressió molt grans, precisament gràcies a aquesta reducció de valors. En el cas dels blocs de *32x32*, els més grans, hi hauria un factor de compressió de 1024.

Tot l'explicat fins ara es pot comprovar gràficament veient com quedaria la matriu d'exemple després d'haver-hi aplicat aquesta tècnica, s'ha adjuntat a continuació.

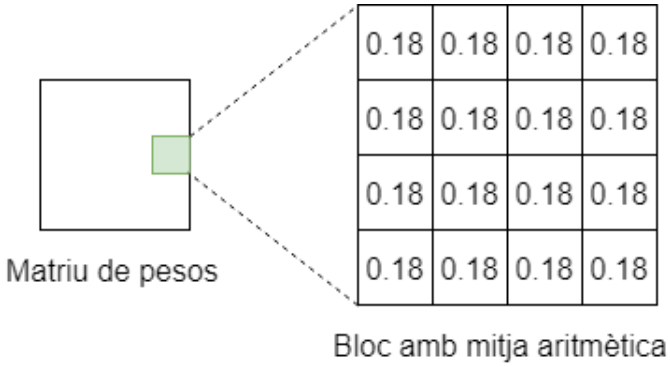

Figura 6.6: Matriu d'exemple amb mitja aritmètica

La complexitat computacional de calcular la mitja aritmètica per cada bloc és $O(n)$, sent n la mida del bloc on, el cas de la matriu d'exemple, seria $n = 4 \cdot 4 = 16$. S'ha adjuntat als apendixos el codi resultant d'haver modificat internament Caffe perquè es fes el càlcul de la mitjana aritmètica per cada bloc de la matriu de pesos.

Resultats obtinguts

La figura que s'adjunta a continuació és el resultat d'haver aplicat la mitjana aritmètica per blocs de diferents mides en la xarxa neuronal CIFAR-10 Quick.

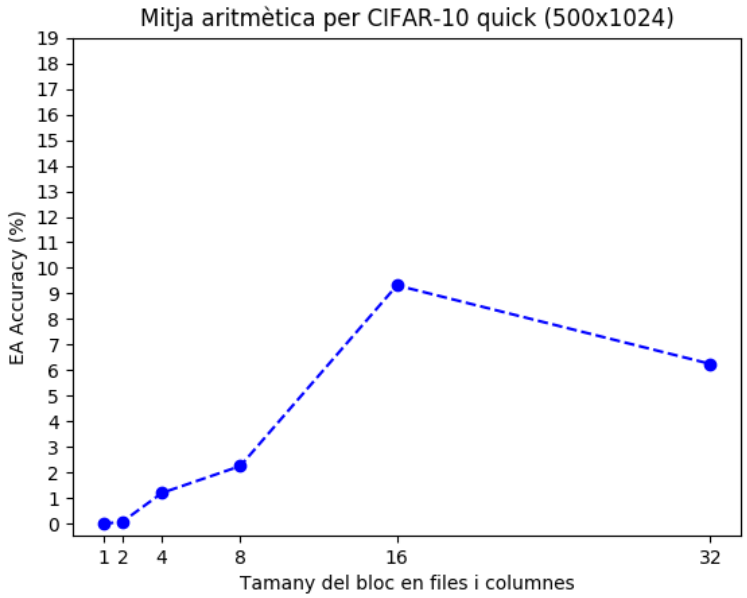

Figura 6.7: Precisions obtingudes en CIFAR-10 Quick amb la mitja aritmètica

En l'eix d'abscisses s'hi ha especificat la mida del bloc, considerant-lo quadrat. D'aquesta forma s'entén que el número 1 fa referència a la seva precisió original, 2 a mides de bloc de *2x2*, 4 a *4x4*, etc.

En l'eix d'ordenades hi ha la precisió en forma d'error absolut entre la precisió de la xarxa sense cap modificació i l'obtinguda aplicant-hi el mètode de la mitjan aritmètica.

L'error absolut es calcularà mitjançant la següent fórmula:

$$\epsilon_0 = |PrecisioModificacio - PrecisioOriginal| \tag{6.2}$$

Es pot comprovar que, tal i com és lògic, a major és el bloc més tendència té a disminuir la precisió obtinguda doncs es limita la llibertat dels blocs condicionant-los en excés. Aquest punt s'agreuja de maneres insostenibles en el cas dels blocs de *16x16* on hi ha un 10% de diferència entre l'original i aquesta modificació.

Malgrat hi hagi hagut bons resultats amb blocs petits, com ara els de mida *2x2*, no ens soluciona el problema de compressió doncs, tal i com s'especifica en la taula de calor, a menor bloc menor compressió, tal és que per la mitja aritmètica en blocs de *2x2* únicament s'aconsegueix un factor de compressió de 4, poc significatiu.

En la figura següent s'observa el desenvolupament de la precisió per epoch que ha anat seguint aquesta tècnica aplicada a les diferents mides de bloc seleccionades.

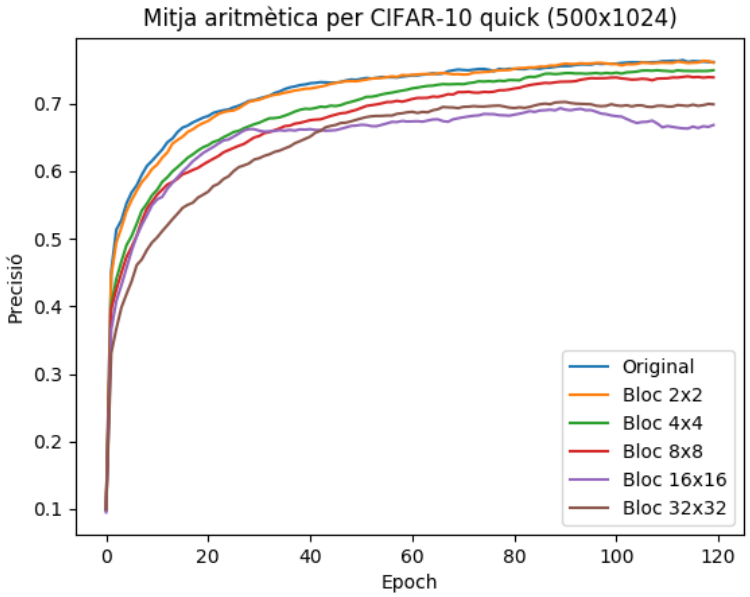

Figura 6.8: Precisions per epoch obtingudes en CIFAR-10 Quick aplicant la mitja aritmètica

Menys els blocs de mida *2x2* i, molt just, *4x4* la resta d'ells no aconsegueixen entrenar-se d'una forma que la precisió es pugui assimilar al original. Després de fer un anàlisis exhaustiu dels resultats obtinguts en les dos gràfiques detectat dos problemes majors que podrien estar produint aquest comportament:

- **Pocs valors per bloc**, com s'ha comentat el fet de limitar els blocs a exclusivament un valor pot estar condicionant excessivament la xarxa i fent que les precisions s'hi vegin afectades.

- **Aleatorietat inicial** dels pesos i partir d'aquests per calcular la mitjana aritmètica. Al ser aleatori existeix la possibilitat que un bloc s'hagi inicialitzat amb valors molt perjudicials per obtenir una bona precisió i amb aquest mètode limita molt el deixar fluir els pesos per arreglar aquest error. Aquest es creu que es el problema pel que el bloc de *16x16* obté pitjor precisió que el de *32x32*, ja que la seva precisió arriba un moment, al voltant del epoch 20, on es queda estancada i no augmenta o inclús disminueix.

Tots aquests resultats i més poden ser consultats en l'apartat d'apèndix, on s'ha adjuntat una taula de resultats per cada mètode i xarxa.

6.2.2.2 Algorismes d'agrupament

Els algorismes d'agrupament [47], més coneguts pel seu terme en anglès *clustering algorithms*, són procediments mitjançant els quals s'agrupa un conjunt dades qualsevol en diferents grups a través d'uns criteris marcats, cadascuna d'aquestes agrupacions s'anomena *cluster*. En l'exemple següent es veu com diferents punts dins d'un espai euclidià de dos dimensions s'han agrupat en clústers, representats en colors diferents segons la seva proximitat, propietat la qual els fa similars en aquest cas.

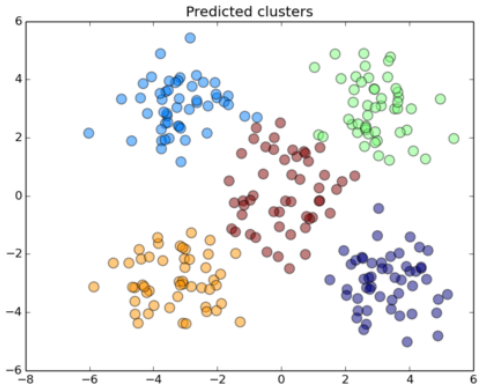

Figura 6.9: Exemple d'algorisme de clustering

En el cas d'aquest projecte aquest tipus algorismes ens van molt bé doncs ens permet igualment reduir el total de valors permesos per bloc, però sense ser tant limitant com la mitja aritmètica, ja que en aquest cas podrem tenir tants valors com diferents grups es permetin, recordant la norma on sempre seran números múltiples de dos, explicada anteriorment.

Existeixen diferents mètodes d'agrupació dels quals s'ha escollit fer servir l'anomenat *K-Means* [48] perquè el desenvolupador ja estava familiaritzat amb el seu funcionament i el problema era adequat per fer-lo servir.

L'objectiu d'aquest algorisme és dividir un conjunt de dades en un nombre de grups representat per la variable K. Cada dada es situarà al grup on la mitjana de les dades d'aquest grup estigui més proper a aquesta, la mitjana representativa de cada *cluster* s'anomenarà *centroides* i n'hi haurà un per cada grup.

Per aquesta investigació es calcula aquest algorisme a cada bloc en els que s'ha dividit la matriu de pesos, per tant el conjunt de dades seran el total de

valors dins de cada bloc, un conjunt de nombres $\mathbb{Z}$, els quals s'entendrà com si estiguessin representats en un espai euclidià unidimensional a l'hora d'entendre la proximitat entre ells. A continuació s'enumera cada pas a fer en aquest algorisme:

1. S'inicialitza de manera aleatòria cada *centroide* representat amb la lletra grega μ_i on aquest subíndex indica el número de *cluster* al que pertany el *centroide*.

$$\mu_1, \mu_2, ..., \mu_i, ..., \mu_k \in \mathbb{Z} \text{ Aleatoris} \tag{6.3}$$

2. Cada valor pertanyent al conjunt $X \in x_1, x_2, ..., x_j, ..., x_N$ (essent X el conjunt de valors de cada bloc i N el nombre total d'aquests valors) se'l situa en el *cluster* c_i més proper segons la seva distància euclidiana amb el *centroide* μ_i corresponent.

$$\text{Per cada valor } j\text{: } x_j \in c_i := \min_i |x_j - \mu_i| \tag{6.4}$$

3. Una vegada recol·locats tots els punts x_j al *cluster* c_i corresponent es procedeix a actualitzar cada *centroide* μ_i fent una mitja aritmètica de tots els valors continguts al grup representat per aquest.

$$\text{Per cada valor } i\text{: } \mu_i = \frac{\sum_{j=0}^{N} x_j \in c_i}{N} \tag{6.5}$$

4. Es va repetint en bucle el pas 2 i 3 fins arribar al punt on no s'actualitzi cap valor a un *cluster* diferent del que està situat, en aquest moment es diu que l'algorisme ha "convergit".

El nombre de grups, K, vindrà marcat per el desenvolupador i serà el mateix que el número de valors permesos per bloc. Al final d'aplicar *K-Means* es substituirà cada valor per el *centroide* representatiu del grup on ha convergit. Al donar més llibertat de valors dins de cada bloc s'ha aconseguit millors precisions sense sacrificar excessiu factor de compressió.

Tot l'explicat fins ara es pot comprovar gràficament a traves de la matriu d'exemple adjuntada a continuació, on s'hi ha aplicat *K-Means*, ja convergit, permetent quatre valors diferents i cadascun pintat amb un color diferent.

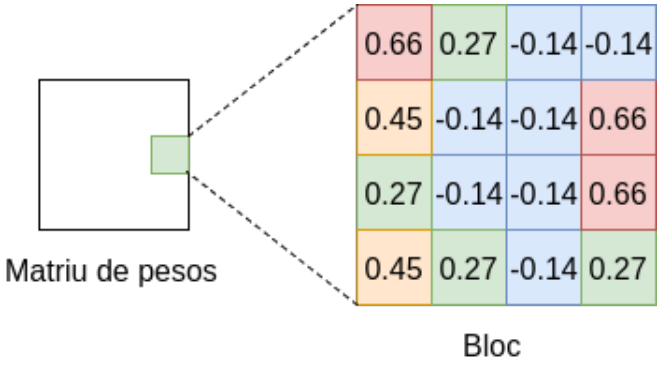

Figura 6.10: Matriu d'exemple amb K-Means de 4 valors diferents

La complexitat computacional de calcular *K-Means* per cada bloc és $O(n^{dk+1})$, sent n la mida del bloc, k el total de valors permesos i d el total de valors que conté un bloc, aquests valors per la matriu d'exemple serien on per la matriu d'exemple serien $n = 4 \cdot 4 = 16$ i $d = n \cdot n$. De no tenir k i d fixades es convertiria en un problema *NP-Hard*. Al tenir un codi suficientment gran s'ha decidit adjuntar-lo als apèndix en la seva totalitat, per poder consultar-se de ser necessari.

Caffe utilitza la tecnologia *"Google protobuf"* [46] de manera interna per la declaració de les xarxes neuronals, s'ha hagut de modificar internament per passar com a paràmetres el màxim de valors permesos en un bloc indicat amb el paràmetre *kas*, tal i com mostra l'exemple de continuació.

```
net_param{
name: "Alexnet"
     .....
  layer {
    name: "ip1"
    type: "InnerProduct"
    .....
    cols: 16
    rows: 16
    kas: 4
    ....
}
```

Figura 6.11: Exemple de d'inicialització per la xarxa AlexNet on es comprimeix la matriu de pesos en blocs de *16x16* permetent un màxim de 4 valors per cadascun d'ells

Resultats obtinguts

La figura que s'adjuntarà a continuació és el resultat d'haver aplicat *K-Means* per blocs de mida *8x8* en la xarxa neuronal CIFAR-10 Quick.

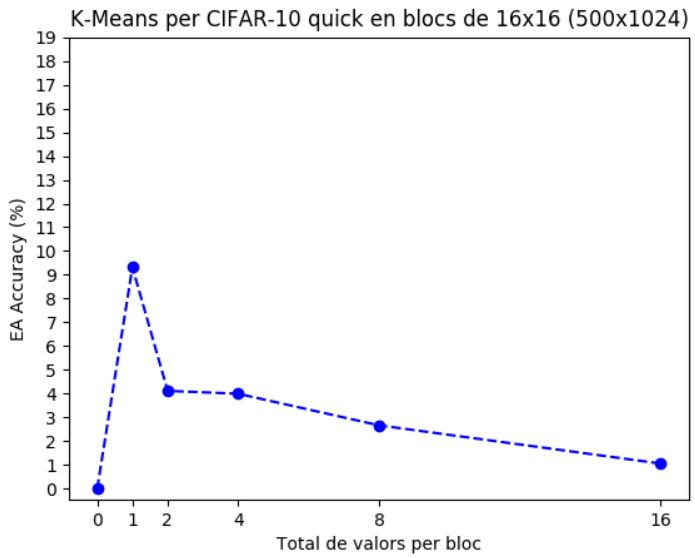

Figura 6.12: Precisions en CIFAR-10 quick per blocs de *16x16* amb *K-Means*

En l'eix d'abscisses s'hi ha especificat el total de valors permesos per bloc. D'aquesta forma s'entén que el número 0 fa referència a la precisió original, el número equivaldria a la seva mitjana aritmètica, 2 a un total de dos valors i a partir d'aquí es procediria consecutivament.

En l'eix d'ordenades hi ha la precisió en forma d'error absolut entre la precisió de la xarxa sense cap modificació i l'obtinguda aplicant-hi el mètode de la mitjan aritmètica. El càlcul d'aquest error absolut s'ha especificat anteriorment.

Es pot observar que a mesura que es permet més valors dins del mateix bloc aquest manté una tendència a millorar la precisió de la xarxa, tal i com s'havia apuntat anteriorment, fins a arribar a uns escassos 2% d'error per 16 valors, on s'aconseguiria reduir dels per setze el total de valors dins d'aquest bloc passant dels 256 de forma original als 16 permesos.

Malgrat els resultats hagin acompanyat a l'hora de permetre valors més grans dins del mateix bloc, al fer això s'ha reduït significativament la compressió que es podria aconseguir, tal i com s'especifica en la taula de calor on s'observa que es passaria d'obtenir un factor de compressió de 256 usant la mitjana aritmètica a únicament 5.33 permetent 16 valors.

En la figura següent es veu el desenvolupament de la precisió per epoch que ha anat seguint aquesta tècnica permetent valors diferents dins del mateix bloc de *16x16*.

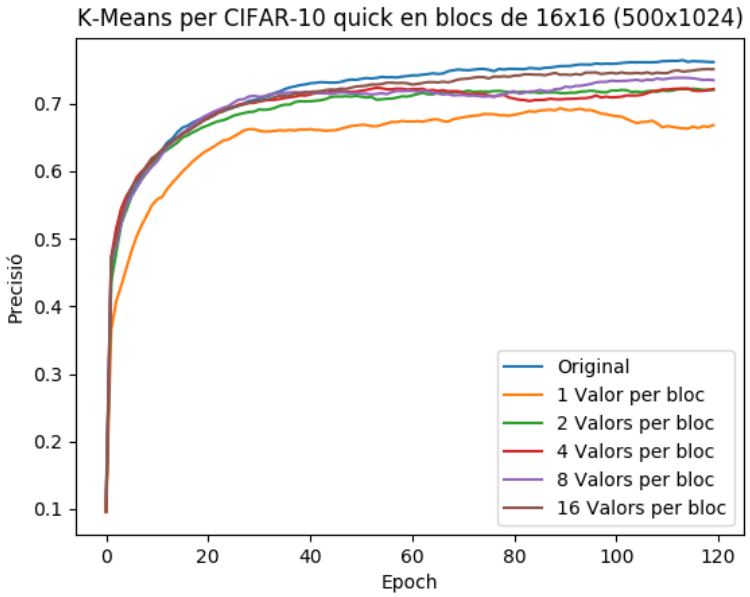

Figura 6.13: Precisió per epoch en CIFAR-10 quick i blocs de 16x16 amb *K-Means*

Es pot notar la diferència en la millors respecte haver passat de la mitjana aritmètica, representada com *K-Means* d'un valor, a algorismes d'agrupament que n'utilitzin més d'un. Les precisions aconseguides són bastant raonables si es té en consideració els factors de compressió als que es pot arribar amb aquestes, malgrat hi hagi pèrdues mínimes. Després d'un anàlisis dels resultats obtinguts s'ha comprovat que segueix existint el problema ja esmentat anteriorment d'inicialització aleatòria dels pesos doncs aquesta tècnica limita molt el deixar fluir els pesos lliurement al aplica-se *K-Means* des de l'inici, aquest fet pot estar limitant la precisió que s'aconsegueix. S'ha afrontat aquest problema en les seccions que vénen a continuació.

Tots aquests resultats i més poden ser consultats en l'apartat d'apèndix, on s'ha adjuntat una taula de resultats per cada mètode i xarxa.

6.2.3 Compressió progressiva

Aquesta subsecció explica l'últim mètode de compressió investigat, pensat especialment per pal·liar un del majors problema comentats amb anterioritat: la inicialització aleatòria dels pesos de la matriu. Aquest afectava tant a la tècnica de la mitjana aritmètica com a la de *K-means*. S'esperava aconseguir factors de compressió considerables sense sacrificar precisió i es compliren aquestes prediccions.

Tal com s'explica en la secció on s'introdueix al lector de la memòria a les xarxes neuronals, aquestes inicialitzen els valors dels seus pesos de manera completament aleatòria i, a mesura que es va entrenant la xarxa, es van adaptant per aconseguir precisions cada vegada més altes. Si des de la primera iteració de l'entrenament s'aplica les tècniques de reducció de valors per bloc anteriors, per molt error-tolerants [39] que siguin les xarxes, s'està restringint en excés la conducta d'aquests pesos, provocant una pèrdua en la precisió final.

La tècnica que s'ha ideat per solucionar aquest problema ha sigut la que anomenarem "compressió progressiva", aquesta dóna completa llibertat d'entrenament a la xarxa durant un número d'epochs suficients com perquè hagi pogut adaptar els seus pesos per aconseguir precisions més elevades, dit d'una altre forma, la xarxa hagi "convergit" i la seva precisió comenci a augmentar cada vegada més lenta. Un cop s'ha arribat a aquest punt s'hi aplica les tècniques vistes anteriorment: mitjana aritmètica i/o *K-means*.

El més significatiu a escollir en aquest mètode és el número d'epochs a partir del qual començar a aplicar les tècniques de reducció de valors per bloc, s'ha escollit aquest valor tenint en compte les tres xarxes neuronals d'experiment usades en aquesta memòria: LeNet, CIFAR-10 quick i AlexNet. La figura adjuntada a continuació mostra el desenvolupament de la precisió per epoch, fins al 120, d'aquestes tres xarxes sense haver-hi aplicat cap de les modificacions comentades al llarg de la memòria. S'ha marcat amb una línia vermella on es pot observar que les tres arriben a "convergir" en el moment coincident en l'epoch 25, conseqüentment, aquest ha sigut el valor que s'ha utilitzat.

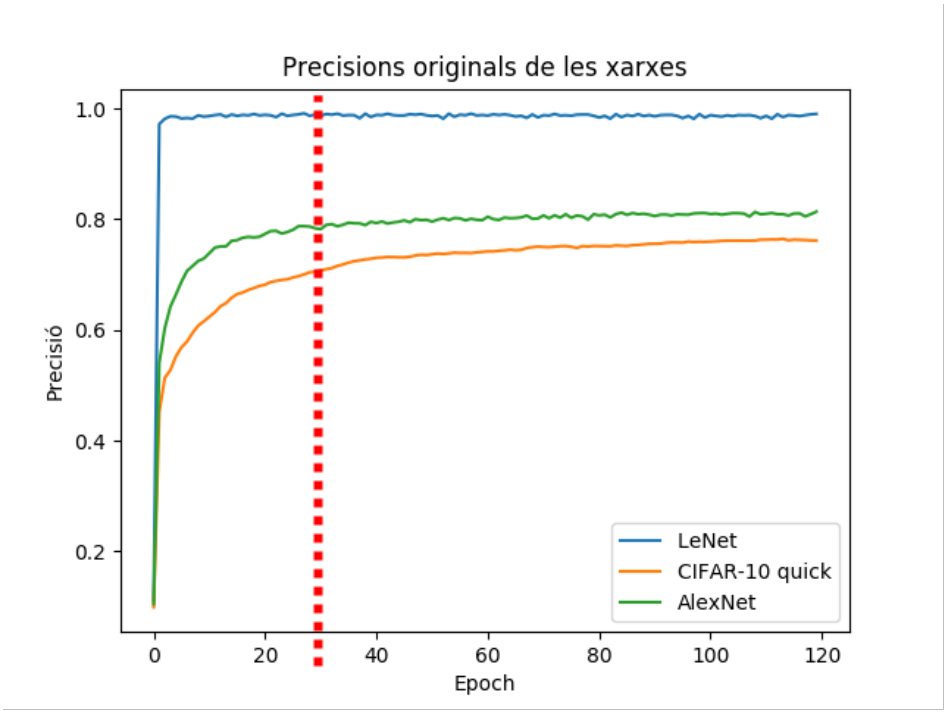

Figura 6.14: Punt de "convergència" entre les tres xarxes neuronals d'expermient

Seguidament s'ha aplicat aquesta tècnica amb les de reducció de valors per bloc vistes anteriorment: la mitjana aritmètica i l'algorisme d'agrupament *K-means*. En les dos següents subseccions s'ha adjuntat resultats per cadascuna d'elles en les xarxes d'experiment: LeNet i CIFAR-10 quick.

6.2.3.1 Mitjana aritmètica

S'ha provat de la xarxa més petita a la més gran, així doncs primerament s'ha experimentat amb LeNet i, després, CIFAR-10 quick. La dos figures següents mostren com s'ha desenvolupat la precisió en aquestes dos xarxes per mides de bloc diferents i aplicant-hi la mitjana aritmètica a partir de l'epoch 25 en cadascun d'aquests. Apareixen línies de colors varis en el gràfic, cadascuna d'elles fent referència a mides de bloc diferent, correctament indicades en la llegenda. Tal i com es pot observar a major és el bloc, major caiguda de precisió es produeix al arribar a l'epoch 25, però ràpidament es recupera i acaba donant resultats bons.

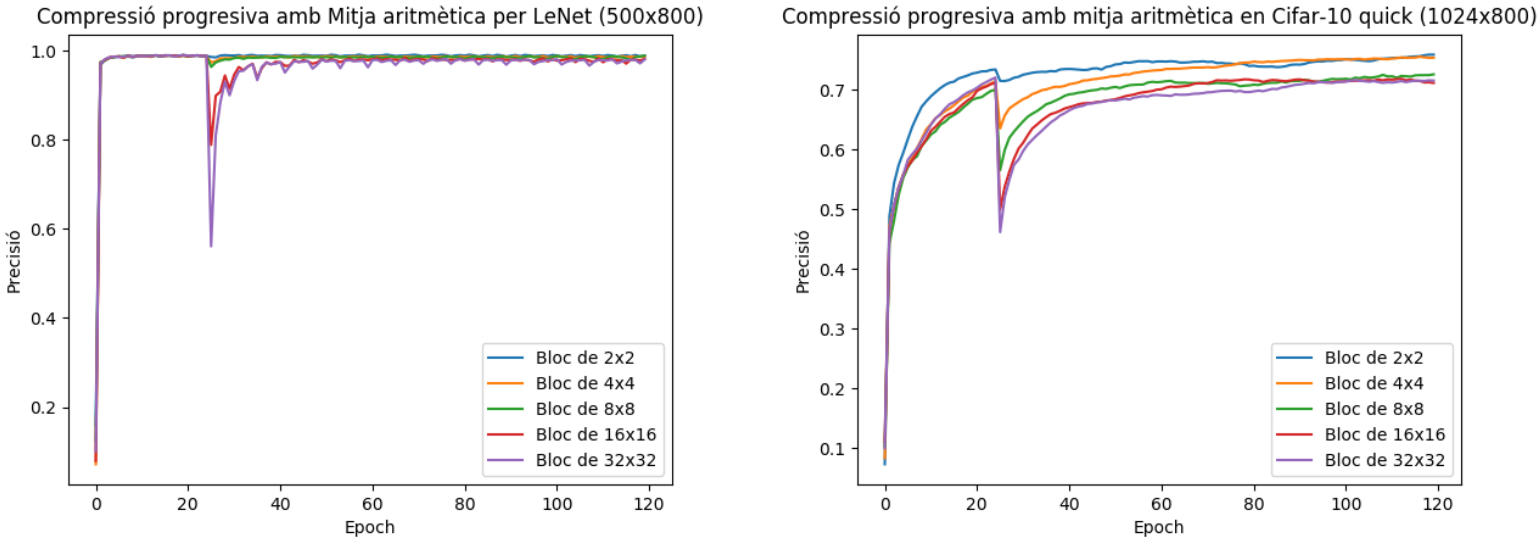

Figura 6.15: Precisió per epoch en LeNet i CIFAR-10 quick usant compressió progressiva amb MA

Per fer una comparació respecte la millora de la precisió aplicant "compressió progressiva" o comprimint des d'un inici s'ha adjuntat dos gràfics on es mostra aquests resultats per mides de bloc diferents. En l'eix d'abscisses s'hi ha especificat la mida del bloc, considerat quadrat, d'aquesta forma s'entén que el número 0 fa referència a la seva precisió original, 2 a mides de bloc de *2x2*, 4 a *4x4*, etc. En les ordenades s'hi mostra la precisió en forma d'error absolut entre la xarxa sense cap modificació i l'obtinguda aplicant-hi el mètode especificat en la llegenda del gràfic. Com s'hi pot observar la "compressió progressiva" generalment sempre acaba produint millors precisions que fent-ho des d'un inici, aquesta diferència és més pronunciada en blocs de mides més grans, els quals són encara més rellevants perquè ens permetran obtenir factors de compressió més alts.

Tots aquests resultats i més poden ser consultats en l'apartat d'apèndix, on s'ha adjuntat una taula de resultats per cada mètode i xarxa.

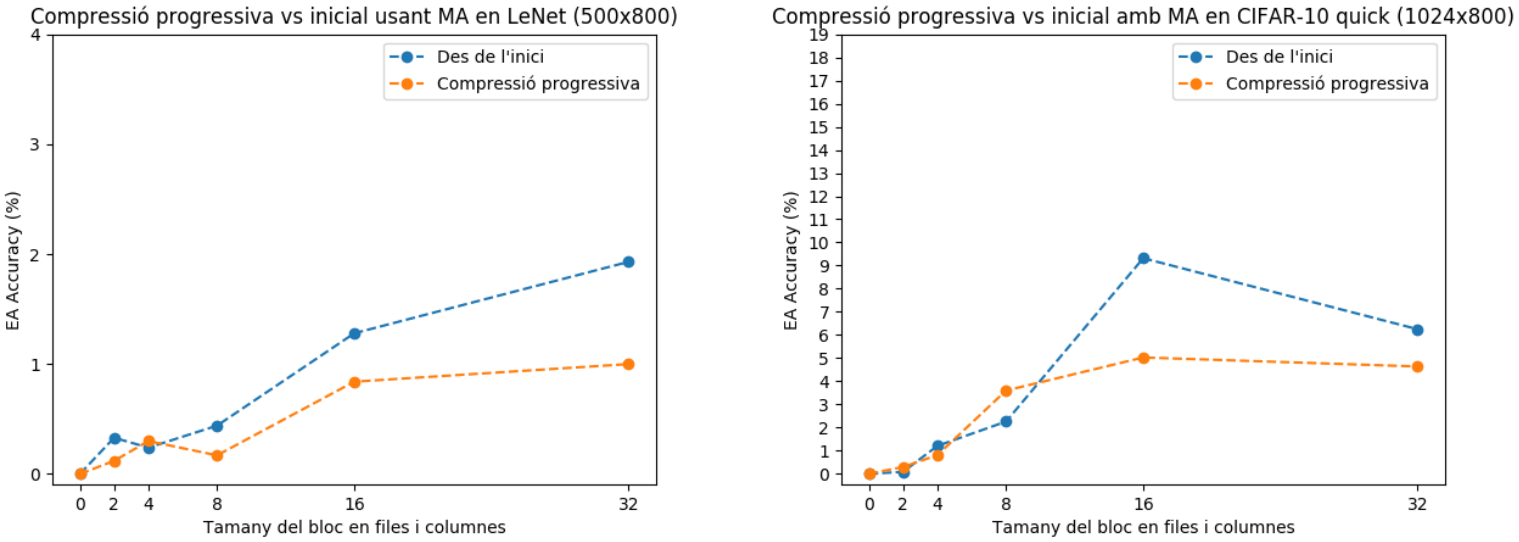

Figura 6.16: Comparació entre les precisions obtingudes amb "compressió progresiva" o des d'un inici, ambdós usant MA, en LeNet i CIFAR-10 quick

6.2.3.2 K-Means

A fi de no saturar al lector amb moltes gràfiques s'ha optat per únicament mostrar aquest cas per blocs de *32x32* en CIFAR-10 quick, la resta de resultats englobant més mides estan situats l'apèndix en forma de taula de resultats. La figura següents mostren com s'ha desenvolupat la precisió en aquesta xarxes per quantitats de valors diferents i aplicant *K-Means* a partir de l'epoch 25 en cadascun d'aquests. A major número de valors permesos per bloc major és la caiguda en la precisió però ràpidament es recupera i acaba donant bons resultats, tal i com succeïa en l'apartat anterior però en aquell cas per mides de bloc.

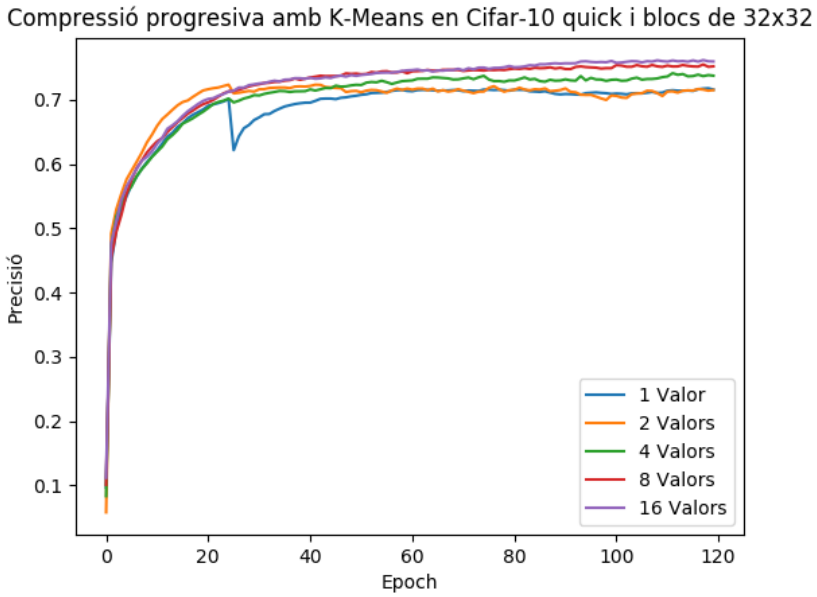

Figura 6.17: Precisió per epoch en CIFAR-10 quick dividit en blocs de *32x32* usant compressió progressiva amb K-Means

Per fer una comparació respecte la millora de la precisió aplicant "compressió progressiva" o comprimint des d'un inici s'ha adjuntat un gràfics on es mostra aquests resultats per diferents valors permesos dins del mateix bloc de mida *32x32* en CIFAR-10 quick. En l'eix d'abscisses s'hi ha especificat el nombre de valors que s'ha permès per bloc. D'aquesta forma s'entén que el número 0 fa referència a la precisió original, el número 1 equivaldria a la seva mitjana aritmètica, 2 a un total de dos valors i a partir d'aquí es procediria consecutivament. En les ordenades s'hi mostra la precisió en forma d'error absolut entre la xarxa sense cap modificació i l'obtinguda aplicant-hi el mètode especificat en la llegenda del gràfic. Tal i com succeïa en l'anterior apartat la "compressió progressiva" acaba produint millors resultats que fent-ho des d'un inici però les diferències encara són més pronunciades a més valors permesos arribant inclús a igualar, o superar lleugerament, la precisió original.

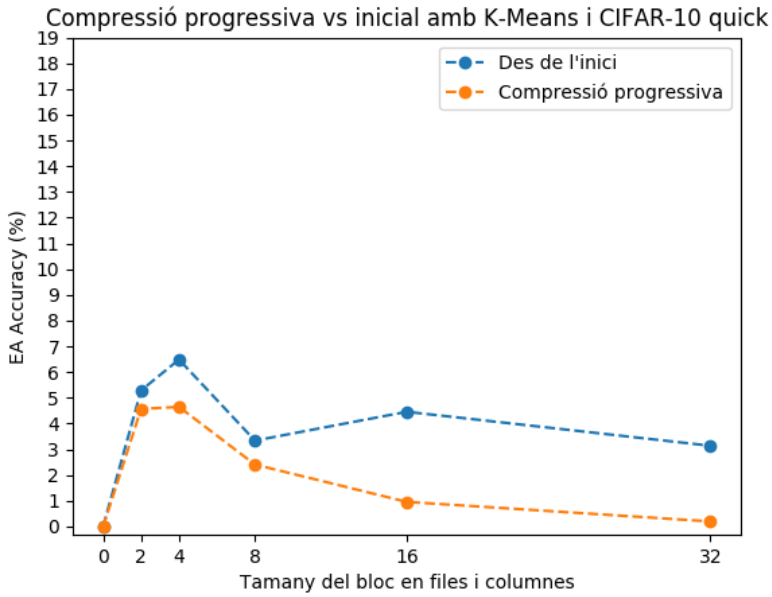

Figura 6.18: Comparació entre les precisions obtingudes amb "compressió progresiva" o des d'un inici, ambdós usant K-Means i blocs de *32x32* i CIFAR-10 quick

6.2.4 Compressió progressiva per percentils

Fins ara s'havia adoptat tècniques de reducció de valors per bloc que han servit per millorar notablement la compressió però perdent precisió, com és el cas de la mitjana aritmètica o per millorar precisió sacrificant-ne compressió, com ha passat amb l'algorisme d'agrupació adoptat: *K-means*. El mètode que s'ha proposat en aquest apartat ha volgut englobar les dos permetent que diferents blocs puguin tenir diferent quantitat de valors i no exclusivament el mateix nombre, com s'havia estat fent fins ara. Aquesta tècnica es fusionarà amb la compressió progressiva.

El principal problema que es va afrontar arribats a aquesta idea va ser la de marcar el filtre a utilitzar per decidir quins blocs tindrien una quantitat de valors o una altre. Després de diverses consideracions es va optar per classificar els blocs segons la seva desviació típica doncs a major sigui aquesta més dispersos seran els valors dels seus pesos i, per tant, més necessitats hi haurà de permetre tenir més valors en aquests blocs en concret.

La forma de calcular la desviació típica per cada bloc es va realitzar utilitzant la següent fórmula, on N seria el total de valors dins un bloc, x_i cada pes en aquest i $\bar{x}$ la mitja aritmètica del total de pesos en el bloc.

$$\sqrt{\frac{\sum_{i=0}^{N}(x_i - \bar{x})^2}{N-1}} \tag{6.6}$$

Aquesta es va implementar internament a Caffe de la manera següent:

```
1  for (int ii = 0; ii < ShapeRows; ii+=BlocRows){
2   for (int jj = 0; jj < ShapeCols; jj+=BlocCols) {
3    double mitja_elems = 0;
4    double total_elems = 0;
5    double desv_quadrat = 0;
6    for (int i = ii; i<min(ShapeRows,ii+BlocRows); ++i) {
7     for (int j = jj; j<min(ShapeCols,jj+BlocCols); ++j) {
8      mitja_elems += tmp[i][j];
9      total_elems += 1;
10    }
11   }
12   mitja_elems /= total_elems;
13   for (int i = ii; i<min(ShapeRows,ii+BlocRows); ++i) {
14    for (int j = jj; j<min(ShapeCols,jj+BlocCols); ++j) {
15
16     desv_quadrat += (Copy[i][j]-mitja_elems)*(Copy[i][j]-mitja_elems);
17    }
18   }
19   double desv = sqrt((desv_quadrat)/total_elems);
20   Desviacions.push_back(make_pair(desv,,make_pair(ii,jj)));
21
22  }
23 }
```

Figura 6.19: Càlcul de la desviació típica en cada bloc de la matriu de pesos

En aquest punt ja s'havia decidit el filtre a utilitzar, ara només era necessària saber la proporció de valors per bloc que es permetria. Es van pensar diferents formes de marcar-la però finalment la utilitzada va ser la de divisió en percentils. Es volia obtenir la major quantitat de blocs amb menors valors possibles així que es va decidir ordenar de menor a major les desviacions obtingudes en cada bloc, després es va dividir aquest conjunt ordenat en percentils cada cop més petits, fent que els primers i de major mida fossin els que continguin els blocs amb menor desviació i se'ls permeti menor quantitat de valors. Aquest càlcul es veu representat en la fórmula següent, on s_i és el total de blocs que aniran al percentil i, p és el percentil

escollit dividit entre 100, k el total de valors que es permetrà tenir a un bloc i N el total de blocs que hi ha en la matriu de pesos.

$$s_1 = N - N \cdot p \rightarrow ... \rightarrow s_{i-1} = s_i - s_i \cdot p \rightarrow ... \rightarrow s_{k-1} = s_k - s_k \cdot p \tag{6.7}$$

Es van explorar percentils amb diferents valors, sempre intentant no baixar excessivament del 40% doncs de fer-ho es trencaria la norma de major quantitat de blocs amb menors valors possibles segons s'ha explicat en l'anterior paràgraf. El que generalment va obtenir millors resultats en totes les xarxes i mides de bloc va ser el 50%, per tant és el que es va adoptar en tots els casos. En la figura següent exemplifica com quedaria la repartició en una matriu que contingués 400 blocs i es permetés un com a màxim 8 valors, obtenint d'aquesta manera 200 blocs amb 1 valor, 100 amb 2 valors i 50 amb 4 i 8.

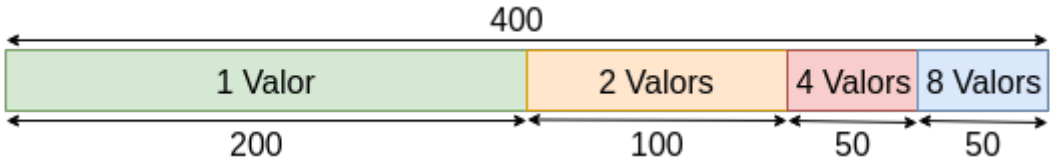

Figura 6.20: Exemple de divisió per percentils al 50% de 400 blocs en, màxim, 8 valors

Arribats a aquest punt es va idear una forma de compressió progressiva que fos més suau a la utilitzada anteriorment. Aquesta començarà igual, entrenant-se lliurement durant els 25 epochs que s'havien marcat, però la diferència és que en intervals de 15 va limitant cada cop més els valors continguts en els blocs, començant pel màxim que es permetin i anant-lo reduint en potències de 2 fins arribar a 1, que serà el mínim i calcularà utilitzat la ja anomenada mitjana aritmètica, en tots els casos s'aplicara la divisió per percentils explicada anteriorment.

En la figura següent es veurà dos exemples de la tècnica que s'ha explicat per comprovar utilitzant màxims de valors permesos per bloc de 8 (figura superior) i 32 (figura inferior) valors. Cada línia vertical marca una fase de com estarà dividit aquest entrenament i valors que es permetran en cadascuna d'elles.

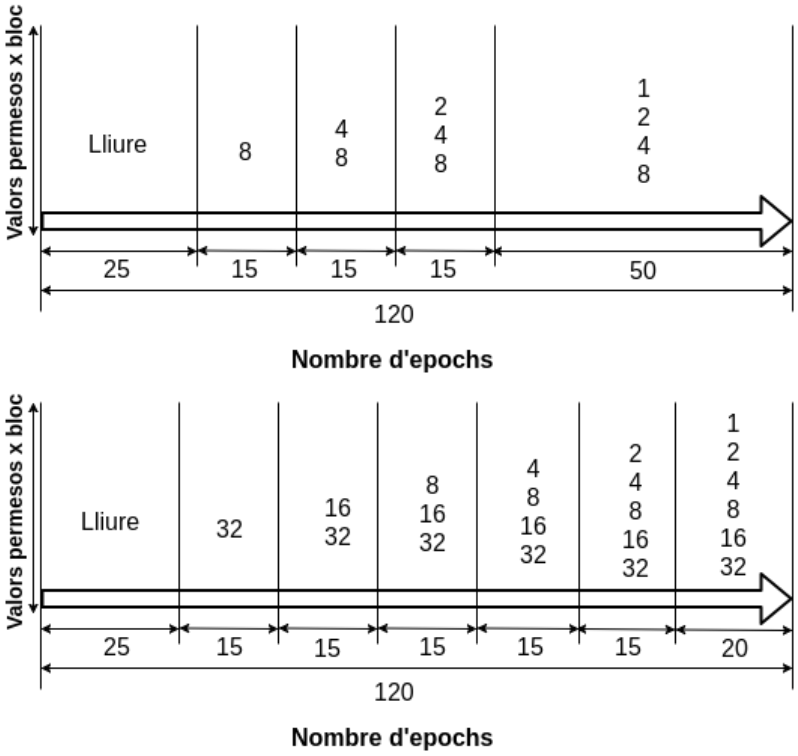

Figura 6.21: Entrenament amb compressió progressiva per percentils permetent un màxim de 8 valors (figura superior) i 32 valors (figura inferior)

Resultats obtinguts

A continuació s'ha adjuntat resultats obtinguts amb les dos xarxes d'exemple: AlexNet i CIFAR-10 quick en el cas de blocs de *32x32*, la resta de resultats englobant les diferents mides i xarxes esta situat a l'apèndix en forma de taula de resultats. Les figures següents mostren com s'ha desenvolupat la precisió d'aquestes xarxes aplicant la compressió progressiva a base de percentils que s'ha explicat. Com es pot comprovar, a partir del epoch 25, es produeix també una caiguda de la precisió que ràpidament es recupera i acaba produint bons resultats .

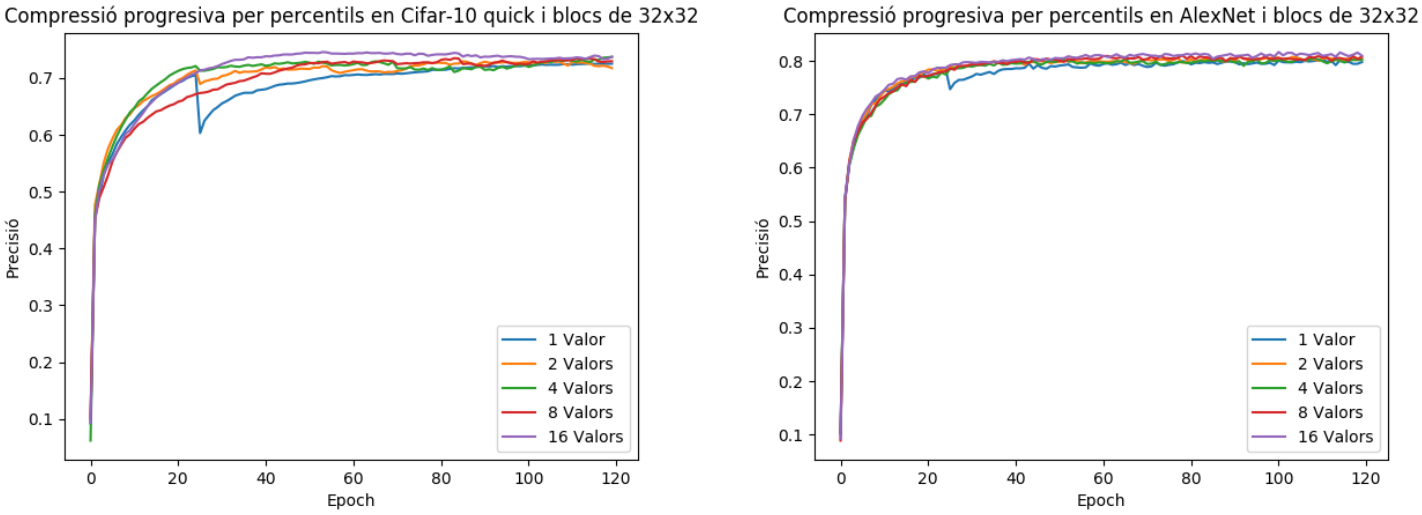

Figura 6.22: Precisió per epoch en CIFAR-10 quick i AlexNet dividits en blocs de *32x32* usant compressió progressiva en percentils

Per fer una millor comparació entre els diferents mètodes experimentats s'ha adjuntat un gràfic on es mostra les tècniques extretes d'aplicar aquests resultats per diferents valors permesos dins de la mateixa mida de bloc de *32x32* en CIFAR-10 quick. En l'eix d'abscisses s'hi ha especificat el nombre de valors que s'ha permès per bloc. D'aquesta forma s'entén que el número 0 fa referència a la precisió original, el número 1 equivaldria a la seva mitjana aritmètica, 2 a un total de dos valors i a partir d'aquí es procediria consecutivament, en el cas de la tècnica que usa percentils el número equival al màxim de valors que es permet. En les ordenades s'hi mostra la precisió en forma d'error absolut entre la xarxa sense cap modificació i l'obtinguda aplicant-hi el mètode especificat en la llegenda del gràfic.

Tal i com s'observa en el gràfic la tècnica que generalment aconsegueix millors resultats és *K-Means* progresiu, però aquest sacrifica molta compressió i la diferència en precisió de l'última proposada, la compressió progressiva a percentils és molt baixa respecte la millora del factor de compressió aconseguit ja que aquesta última garanteix un mínim del 50% de blocs amb mitjana aritmètica i únicament hi haurà l'últim percentil, i menor, amb blocs que continguin el màxim de valors possibles.

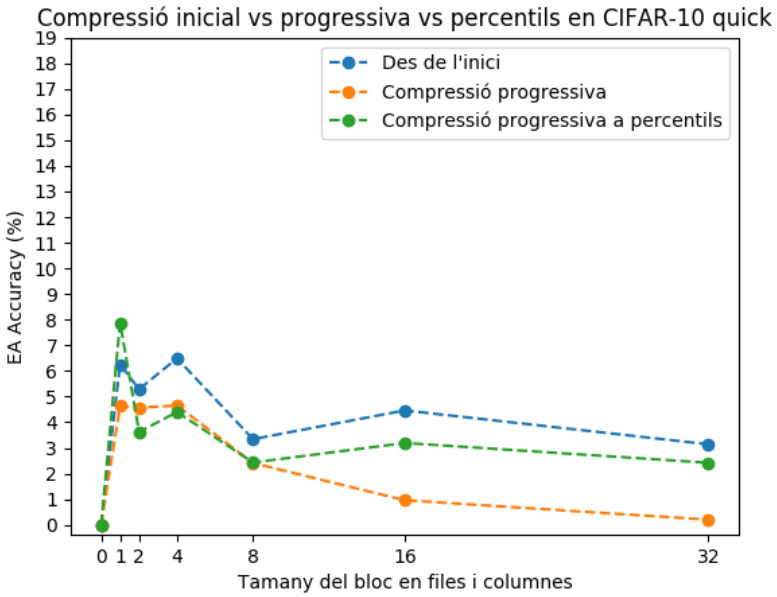

Figura 6.23: Comparació entre les precisions obtingudes amb els diferents mètodes tractats: compressió des d'un inici, compressió progresiva" o progressiva amb percentils, les dos primeres usaran K-Means i, les tres, blocs de *32x32* en CIFAR-10 quick

També s'adjuntarà el mateix gràfic que l'anterior però aplicant exclusivament la tècnica de compressió progressiva en percentils per AlexNet i blocs de *32x32*. Aquesta xarxa és la més gran de totes, arribant a comprimir dos matrius de *4096x2048* i *2048x1024*, malgrat aquestes mides s'ha aconseguit que la diferència de precisió amb l'original sigui 2% en tots els casos, aconseguint així factors de compressió enormes.

Tots aquests resultats i més poden ser consultats en l'apartat d'apèndix, on s'ha adjuntat una taula de resultats per cada mètode i xarxa.

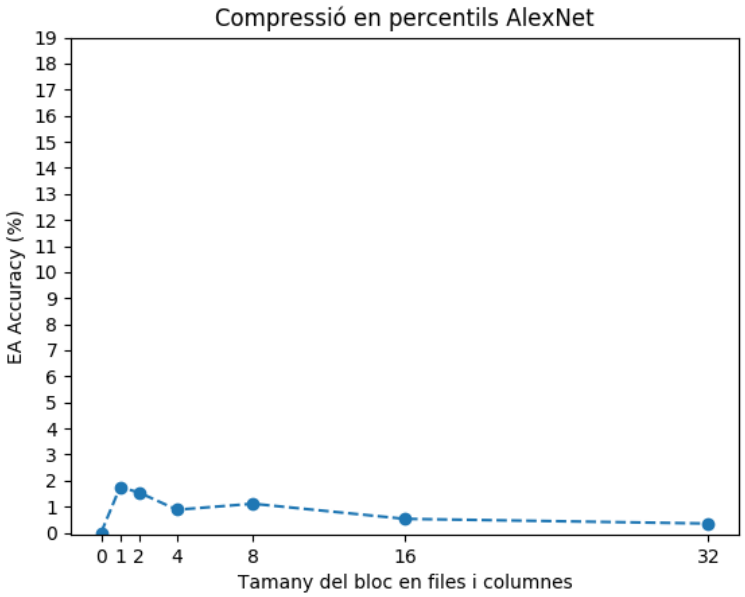

Figura 6.24

6.2.5 Representació binària dels blocs

Un cop establerta la divisió en blocs i la forma de distribuir la quantitat de valors en aquests cal fixar una representació binària a fi de poder avaluar els factors de compressió aconseguits en cada mètode.

En aquesta subsecció s'explicarà la representació en binari de la compressió que s'ha escollit. S'ha exemplificat mitjançant el dibuix següent:

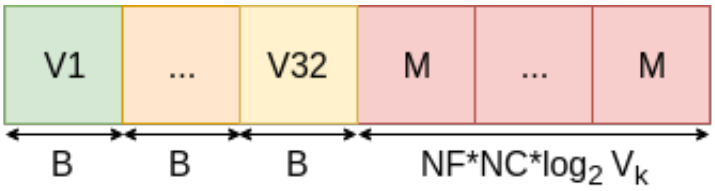

Figura 6.25: Representació en binari de la compressió proposada

En la representació binària apareixeran Vk valors on $k \in \{1..32\}$. Cadascun d'aquests ocuparà B bits, per defecte les xarxes neuronals treballen amb aritmètica de punt flotant de 32 bits, una optimització podria ser reduir aquesta mida de B bits a aritmètiques més petites tal com proposa el paper [40] i així aconseguir millors factors de compressió. Els blocs als que es fa referència en aquest paràgraf

surten pintats de color verd, taronja i groc en l'exemple.

Els blocs representats en color vermell en la figura d'exemple serviran de màscara per tal de saber les posicions on es situarà cada valor dins el bloc que s'estigui codificant, d'haver-hi únicament un valor aquesta màscara no existirà ja que serà innecessària. La mida de la màscara haurà de ser mínim el nombre total de valors per bloc, d'aquí s'extreu el producte entre el nombre de files (NF) i columnes (NC), al estar en binari es necessitarà multiplicar el resultat anterior per $\log_2 V_k$, on $k \in \{1..32\}$ és el total de valors del bloc representat, doncs es passara el número d'aquests a binari i s'afegirà a la seva corresponent posició de la màscara.

La figura següent exemplificarà una màscara d'un bloc de *3x3* amb quatre valors diferents, cadascun d'ells es representarà en binari usant: 00_2 en la posició on es situí el valor V_0, 01_2 per V_1, 10_2 per V_2 i 11_2 per V_3. En total hi haurà $3 \cdot 3 \cdot log_2 4 = 18$ bits en la màscara, acolorits en funció del valor que codifiquen.

Figura 6.26: Exemple de màscara per bloc de *3x3* i 4 valors

6.2.6 Factors de compressió aconseguits

Una vegada en tenim la representació en binari se'n pot extreure el factor de compressió respecte el que suposaria no aplicar-hi cap modificació. S'està calculant aquest factor en funció d'un bloc, però al estar la matriu dividida en ells el resultat és extrapolable per aquesta.

La representació binària sense modificacions serà un producte entre: el total de valors del bloc, calculat amb el producte entre el nombre de files (NF) i columnes (NC) d'aquest, i els bits que ocuparà la seva aritmètica (B) que al ser de punt flotant n'ocuparà 32 per defecte.

$$NF \cdot NP \cdot B \tag{6.8}$$

De l'apartat de la representació binària per blocs en podem extreure la fórmula per calcular la mesura en bits que ocuparà aquesta. Hi haurà K blocs de mida B bits (32 per defecte al ser aritmètica de punt flotant) a més d'una màscara de mida $NF \cdot NC \cdot log_2 K$.

$$K \cdot B + NF \cdot NC \cdot \log_2 K \tag{6.9}$$

Per calcular el factor caldrà fer una divisió entre el que ocupa la codificació del bloc sense cap compressió, situat al dividend, i el que ocupa amb ella, localitzat al divisor.

$$\frac{NF \cdot NC \cdot B}{K \cdot B + NF \cdot NC \cdot \log_2 K} \tag{6.10}$$

6.2.6.1 Taula de calor dels factors de compressió

S'ha creat una taula de calor, adjuntada als apèndix, aquesta expressa la fórmula anterior donant valor a les diferents variables: nombre de files (NF), nombre de columnes (NC) i total de valors permesos (K). Es tindrà en compte que els blocs són quadrats per tant que el nombre de files i columnes sempre serà el mateix, representat com a N.

Aquesta forma un mapa de calor sobre els diferents factors de compressió obtinguts: marcant en vermell els majors, groc els intermedis i blau els menors, també farà una mescla entre algun d'aquests colors si el valor es troba en un punt entremig.

En el cas d'aquesta taula s'ha ampliat el nombre de valors permesos fins a 512 per pura experimentació, però, tal i com es pot observar, les compressions aconseguides per més de 32 valors no són destacables, a través d'aquest experiment és d'on s'ha extret la norma aplicada a la secció dels valors permesos per bloc on no es permet més que el nombre de valors permesos per bloc sigui superior a la mida d'alçada i/o amplada d'aquest.

Capítol 7

Conclusions

En aquest apartat s'hi ha especificat les conclusions extretes a través de la realització d'aquest projecte. S'ha diferenciat aquestes conclusions en: personals (que han afectat, o podran fer-ho, directament al desenvolupador) i del projecte (extretes a partir dels resultats obtinguts amb les diferents tècniques proposadas al llarg de la memòrial.

7.1 Personals

En aquest projecte he aprés a diferents tòpics molt populars i demanats per les empreses actualment com són l'aprenentatge automàtic i les xarxes neuronals profundes els quals abans desconeixia. Si bé és cert que en certs punts de la carrera es mencionen, no s'arriben mai a tractar a fons degut al seu caràcter tan especialitzat. Personalment estava considerant la realització d'un màster en *data science* i aquest treball de fi de grau m'ha ajudat a valorar-ho positivament.

He après a usar el framework per tractament d'aprenentatge profund Caffe i no únicament això sinó que l'he hagut de modificar internament per aconseguir portar a terme aquest projecte.

El camp que més s'ha tocat és el de les xarxes neuronals convolucionals, sobre les quals he aprés el seu funcionament complet, entrenament, testeig, etc. tal i com està documentat en aquest apartat.

Al ser aquest un projecte d'investigació, al primer que he pres part, m'ha ajudat a veure com funciona internament tot el seu procés i com actuar en cada part.

7.2 Resultats del projecte

En aquest projecte s'han proposat un seguit de metodologies per intentar pal·liar el cost computacional i en emmagatzematge que tenen les xarxes neuronals actualment. S'ha treballat amb la hipòtesis d'exportar aquestes tècniques a un futur treball que serà la creació d'un hardware especialitzat per tractar les compressions proposades, donant així una solució els problemes que s'ha comentat

Prèviament a aquest projecte existien diferents estudis, explicats en l'estat de l'art, que s'havien adonat del mateix problema que s'intenta abordar en aquesta investigació i havien proposat diferents idees per solucionar-lo. Analitzant-les i, en algun cas, reproduint-les, es comprovar que les xarxes neuronals tenen un propietat error-tolerant [39] considerablement a partir de la qual es pot partir.

La idea principal era comprimir la matriu per parts, més tard desenvolupats com a divisió per blocs. Per cadascuna d'aquestes parts s'ha buscat reduir el nombre de valors continguts en ella.

Inicialment es va aplicar una mitjana aritmètica, deixant un únic valor per cada bloc per més tard comprovar que malgrat la compressió era alta els resultats de la precisió no acompanyaven doncs es restringia excessivament la quantitat de valors i al estar els pesos inicialitzats aleatòriament i aplicar-hi la mitja aritmètica des del primer epoch no se'ls deixava suficient llibertat per adaptar-se.

Per arreglar aquests problemes es va canviar l'ús de la mitjana aritmètica per algorismes d'agrupació on, d'entre tots, es va escollir *K-Means*. Aquest ens va permetre tenir diferents valors per cadascun dels blocs però només donava resultats acceptables si el nombre de valors era suficientment alt com perquè el factor de compressió no ho fos. Encara s'arrossegava el problema d'inicialització aleatòria dels pesos.

Com el problema venia derivat de la inicialització es va decidir donar llibertat uns quants epochs a la xarxa i després aplicar-hi les tècniques de restricció de valors que s'han comentat, esperant així que es solucionaria el problema, com va acabar passant.

Finalment s'ha volgut encara afinar encara més, sacrificant poca precisió aconseguir una millora de compressió molt més significativa, per fer-ho s'ha permès que dins d'una mateixa matriu hi hagi blocs amb diferents valors per cadascun d'ells, escollint aquest número de valors màxims permesos en funció de la desviació típica entre els pesos continguts en el bloc i la mitjana aritmètica

d'aquest. Per tal de no perdre precisió amb aquesta tècnica s'ha dividit el total de blocs, ordenats amb la desviació típica, per el 50%, agafant contínuament la meitat menor d'aquests blocs i situant-lo en la meitat, així garantint que hi haurà sempre un mínim de blocs suficientment petits com perquè la compressió funcioni. Aquesta tècnica ha donat lleugerament pitjors resultats que simplement aplicant *K-Means* però la diferència mínima que es produeix en la precisió es compensa pel guany de compressió respecte usar una tècnica o altre. En el cas de la xarxa més gran provada, AlexNet amb dos matrius de *4096x2048* i *2048x1024*, usant la tècnica de compressió progressiva per percentils en la mida de blocs més grossa (*32x32*) s'ha aconseguit no tenir una diferència més alta que el 2% respecte la precisió original de la xarxa.

Capítol 8

Extensions del projecte

Al haver-se tractat d'un projecte de tan curta durada no ha permès que es pogués explorar en profunditat completament aquesta idea ja que aquesta és molt versàtil i es podria investigar per moltes variables diferents. Durant la memòria s'ha fet algunes puntualitzacions quan s'han trobat altre possibles camins d'investigació que s'haguessin pogut explorar, però en aquest apartat s'han recollit tots ells per si algun lector interessat en el tema volgués portar-ne algun d'ells a terme.

Creació d'un hardware específic per aquest tipus de compressió

Aquest ha sigut un tema recurrent al llarg de la memòria, doncs s'està treballant en tot moment partint de que les tècniques explicades en aquesta memòria serà possible adaptar-les en un hardware (idealment d'una mida petita) que, al ser específic, permetrà minimitzar el seu cost computacional una vegada entrenades les xarxes i el cost en memòria que representarien. Seria necessària la creació d'un prototip hardware que complís aquests requisits i es demostrés el seu augment de rendiment.

Reducció en l'aritmètica de punt flotant

Tal i com ja s'ha apuntat en la secció de la codificació en binari de les tècniques de compressió usades en aquest projecte, es pot observar que una de les variables per al càlcul del factor de compressió és el nombre de bits que ocuparà cadascun dels valors permesos en cada bloc a memòria. Existeixen alguns papers que afronten aquest tema [40] i que han aconseguit uns bons resultats, si s'aconseguís fusionar les tècniques de compressió esmentades en aquest document amb una reducció de l'aritmètica de punt flotant de 32 bits a menys, 8, 12, 16, etc. els factors de compressió serien significativament més alts.

Blocs no uniformes

Al llarg d'aquesta memòria s'ha especificat que una matriu només es podria dividir en blocs quadrats, de la mateixa mida per files i columnes, amb l'única excepció que aquesta no fos múltiple amb mesura de files o columnes de la matriu de pesos. Però si s'aconseguís idear un mecanisme per permetre que els blocs fossin no únicament rectangles, sinó figures no uniformes i es formessin de pesos similars al seu voltant tècnicament es podria aconseguir millors resultats tant en compressió com en precisió doncs podrien existir blocs molt més grans que es codifiquessin conjuntament i, a l'hora, no es restringirien en extrem els valors dels pesos.

Mida de blocs modificables durant l'entrenament

En aquesta memòria, per falta de temps, només s'ha explorat la possibilitat de permetre valors diferents en el mateix bloc i, inclús, blocs amb diferents valors dins d'una mateixa matriu. En cap moment s'ha considerat la possibilitat de permetre blocs de mides diferents i que aquestes també fossin canviants durant l'entrenament per tal que els pesos s'adaptessin correctament i no de forma restrictiva. Fer aquest mètode en combinació amb el proposat en aquesta memòria podria aportar molts bons resultats, però bé és cert que s'hauria d'idear una nova codificació en binari, ja que la proposada en aquest projecte no funcionaria.

Blocs amb pruning

Tal com s'ha explicat durant l'estat de l'art, existeix un mètode d'alliberament de la xarxa neuronal anomenat pruning [41]. Aquesta elimina connexions entre les diferents matriu de pesos que tinguin valors zero o molt pròxims a ells, ja que aquestes no modifiquen la sortida final de la xarxa. Es podria exportar aquesta idea, en comptes de fer-ho per pesos, fer-ho per blocs iguals als explicats en aquesta memòria, d'aquesta forma s'aconseguiria una reducció del cost computacional que requereix l'execució i entrenament d'una xarxa neuronal conjuntament amb un factor de compressió encara més alt, doncs segon la codificació binària que s'ha proposat en aquest projecte aquests blocs ocuparien 0 bits en memòria.

Apèndixs

Apèndixs A

Taules de resultats per cada mètode especificat en la memòria

A.1 Taula de resultats per LeNet

Mètodes (valors x bloc)	2x2	4x4	8x8	16x16	32x32	Original
MA	*98,70%*	*98,79%*	*98,59%*	*97,75%*	*97,10%*	***99.03%***
MA progressiu	*98,90%*	*98,73%*	*98,86%*	*98,19%*	*98,03%*	***99.03%***
MA percentils	*98,97%*	*98,69%*	*98,51%*	*98,10%*	*97.59%*	***99.03%***
K-Means (2)	*98,92%*	*98,72%*	*98,80%*	*98,76%*	*98,72%*	***99.03%***
K-Means (2) progresiu	*98,89%*	*98,80%*	*98,76%*	*98,97%*	*98,71%*	***99.03%***
K-Means (2) percentils	*98,83%*	*98,96%*	*98,83%*	*98,87%*	*99,01%*	***99.03%***
K-Means (4)	-	*98.98%*	*98.78%*	*98.70%*	*98.75%*	***99.03%***
K-Means (4) progresiu	-	*98.80%*	*98.83%*	*98.64%*	*98.86%*	***99.03%***
K-Means (4) percentils	-	*98.80%*	*98.85%*	*99.03%*	*99.02%*	***99.03%***
K-Means (8)	-	-	*98,46%*	*98,72%*	*98,80%*	***99.03%***
K-Means (8) progresiu	-	-	*98,96%*	*98,78%*	*98,64%*	***99.03%***
K-Means (8) percentils	-	-	*98,84%*	*98,91%*	*98,90%*	***99.03%***
K-Means (16)	-	-	-	*98,43%*	*98,85%*	***99.03%***
K-Means (16) progresiu	-	-	-	*98,91%*	*98,56%*	***99.03%***
K-Means (16) percentils	-	-	-	*98,90%*	*98,96%*	***99.03%***
K-Means (32)	-	-	-	-	*95,57%*	***99.03%***
K-Means (32) progresiu	-	-	-	-	*98,71%*	***99.03%***
K-Means (32) percentils	-	-	-	-	*98,97%*	***99.03%***

Taula A.1: Taula de resultats per LeNet

A.2 Taula de resultats per CIFAR-10 quick

Mètodes	2x2	4x4	8x8	16x16	32x32	Original
MA	*76,06%*	*74,95%*	*73,9%*	*66,84%*	*69,91%*	***76,16%***
MA progressiu	*76,86%*	*75,34%*	*72,56%*	*71,13%*	*71,52%*	***76,16%***
MA percentils	*76,41%*	*73,93%*	*74,50%*	*71,69%*	*68,28%*	***76,16%***
K-Means (2)	*75,46%*	*75,02%*	*74,78%*	*72.03%*	*70,87%*	***76,16%***
K-Means (2) progresiu	*76,63%*	*72,54%*	*74,14%*	*71,53%*	*71,59%*	***76,16%***
K-Means (2) percentils	*75,17%*	*75,29%*	*74,59%*	*70,44%*	*72,54%*	***76,16%***
K-Means (4)	-	*75,92%*	*75,14%*	*72,16%*	*71,51%*	***76,16%***
K-Means (4) progresiu	-	*74,88%*	*75,30%*	*73,70%*	*71,51%*	***76,16%***
K-Means (4) percentils	-	*75,86%*	*74,93%*	*74,13%*	*71,76%*	***76,16%***
K-Means (8)	-	-	*74,98%*	*73,49%*	*72,81%*	***76,16%***
K-Means (8) progressiu	-	-	*75,37%*	*73,07%*	*73,75%*	***76,16%***
K-Means (8) percentils	-	-	*74,91%*	*73,16%*	*73,73%*	***76,16%***
K-Means (16)	-	-	-	*75,09%*	*71,70%*	***76,16%***
K-Means (16) progresiu	-	-	-	*75,26%*	*75,13%*	***76,16%***
K-Means (16) percentils	-	-	-	*74,56%*	*72,97%*	***76,16%***
K-Means (32)	-	-	-	-	*72,85%*	***76,16%***
K-Means (32) progresiu	-	-	-	-	*75,96%*	***76,16%***
K-Means (32) percentils	-	-	-	-	*73,74%*	***76,16%***

Taula A.2: Taula de resultats per CIFAR-10 quick

A.3 Taula de resultats per AlexNet

Mètodes (valors x bloc)	2x2	4x4	8x8	16x16	32x32	Original
MA percentils	*80,88%*	*80,25%*	*80,10%*	*79,98%*	*79.60%*	***81.36%***
K-Means (2) percentils	*81,33%*	*80,40%*	*80,72%*	*80,59%*	*79,81%*	***81.36%***
K-Means (4) percentils	-	*80.75%*	*81.28%*	*80.47%*	*80.47%*	***81.36%***
K-Means (8) percentils	-	-	*81,17%*	*81,14%*	*80,24%*	***81.36%***
K-Means (16) percentils	-	-	-	*80,67*	*80,37%*	***81.36%***
K-Means (32) percentils	-	-	-	-	*81,01%*	***81.36%***

Taula A.3: Taula de resultats per AlexNet

Apèndixs B

Diagrama de Gantt del projecte

B.1 Diagrama de Gantt del projecte

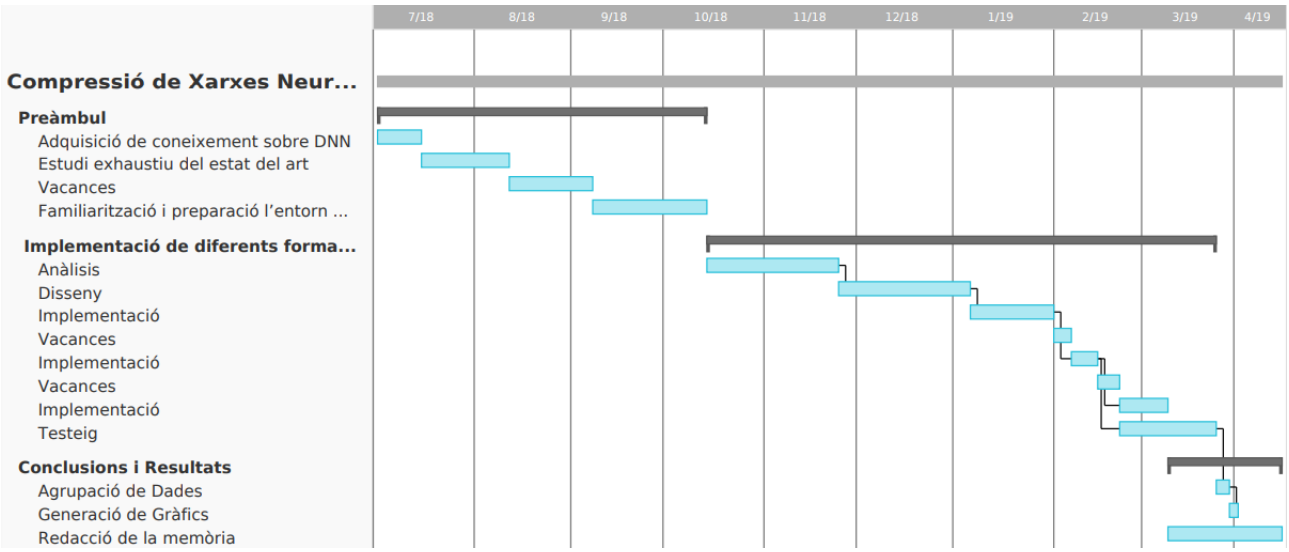

Figura B.1: Diagrama de Gantt del projecte

Apèndixs C

Codis usats per la realització d'aquest projecte

```
1  from pylab import *
2  import sys
3  import os
4  caffe_root = '../../../'
5  sys.path.insert(0, caffe_root + 'python')
6  import caffe
7  from caffe import layers as L, params as P
8  import matplotlib.pyplot as plt
9  import numpy as np
10
11 #Train on GPU
12 caffe.set_device(0);
13 caffe.set_mode_gpu()
14
15 TrainingSamples = 50000
16 TestSamples = 10000
17 batch_size = 100
18
19
20 #Entrena la xarxa
21 def train(iters,print_interval,solver):
22
23   train_loss = [0]*(int(iters//print_interval))
24   test_acc = [0]*(int(iters//print_interval))
25
26
27   # the main solver loop
28   for it in range(int(iters)):
29
30     correct = 0
31     loss = 0
32
33     solver.net.forward()
34
35     solver.net.backward()
```

Figura C.1(1): Codi del fitxer "texting.py" usat per entrenar les xarxes neuronals en aquest projecte

```
    if it % print_interval == 0:

     for test_it in range(int(TestSamples/batch_size)):

      solver.test_nets[0].share_with(solver.net)
      solver.test_nets[0].forward()
      correct += solver.test_nets[0].blobs['accuracy'].data

     test_acc[int(it // print_interval)] = correct / (TestSamples/batch_size)
     train_loss[int(it // print_interval)] += solver.net.blobs['loss'].data

    solver.apply_update()

  return [train_loss, test_acc]

#Initialize solver (SGD)
solver = None
solver = caffe.SGDSolver(sys.argv[1])

#Normal Training
epoch = TrainingSamples/batch_size
[loss,acc] = train(120*epoch, epoch, solver) #SC 70 epoch's & BC 70 epoch

#Prints
a_imprimir = ','.join(map(str,loss))
print(a_imprimir)
a_imprimir = ','.join(map(str,acc))
print(a_imprimir)
```

Figura C.1(2): Codi del fitxer "texting.py" usat per entrenar les xarxes neuronals en aquest projecte

```
1  int shapeRows = learnableParams[0]->shape(0);
2  int shapeCols = learnableParams[0]->shape(1);
3
4  for (int ii = 0; ii < shapeRows; ii+=rows) {
5      for (int jj = 0; jj < shapeCols; jj+=cols) {
6          double mitja_elems = 0;
7          double total_elems = 0;
8          for (int i=ii; i<min(shapeRows,ii+rows); ++i) {
9           for (int j=jj; j<min(shapeCols,jj+cols); ++j) {
10           mitja_elems += tmp[i][j];
11           total_elems += 1;
12          }
13         }
14         mitja_elems /= total_elems;
15         for (int i=ii; i<min(shapeRows,ii+rows); ++i) {
16          for (int j=jj; j<min(shapeCols,jj+cols); ++j) {
17           tmp[i][j] = mitja_elems;
18          }
19         }
20     }
21 }
```

Figura C.3: Codi modificat de Caffe pel càlcul de *K-Means* en cada bloc de la matriu de pesos

```
for (int ii = 0; ii < shapeRows; ii+=rows){
    for (int jj = 0; jj < shapeCols; jj+=cols) {
     //Pas 1
     vector< pair<double,int> > ElemsBloc;
     vector<double> KCentroids(K);
     int count0 = 0;

        for (int i = ii; i<min(shapeRows,ii+rows); ++i) {
      for (int j = jj; j<min(shapeCols,jj+cols); ++j) {

       ElemsBloc.push_back(make_pair(tmp[i][j],0));

       if (count0 < K) { //Inicialitza "Random" els centroides

              KCentroids[count0] = tmp[i][j];
              ++count0;
             }
            }
      }
      //Pas 2
      bool finish = false;
      while(!finish){
      finish = true;
          for (int i = 0; i<ElemsBloc.size(); ++i){
           double min =
               sqrt((ElemsBloc[i].first - KCentroids[ElemsBloc[i].second])
                  * (ElemsBloc[i].first - KCentroids[ElemsBloc[i].second]));
           for (int j = 0; j<KCentroids.size(); ++j) {
            if (j != ElemsBloc[i].second) {

             double min_test = sqrt((ElemsBloc[i].first - KCentroids[j])
                             * (ElemsBloc[i].first - KCentroids[j]));

             if (min_test < min){

              min = min_test;
              ElemsBloc[i].second = j;
              finish = false;
             }
            }
           }
          }
```

Figura C.3(1): Codi modificat de Caffe pel càlcul de *K-Means* en cada bloc de la matriu de pesos

```
1
2                if (!finish){
3                 vector< pair<double,double> > sumKCentroids(K,make_pair(0.0,0.0));
4
5                 for (int i = 0; i<ElemsBloc.size(); ++i) {
6
7                  sumKCentroids[ElemsBloc[i].second].first += ElemsBloc[i].first;
8                  sumKCentroids[ElemsBloc[i].second].second += 1;
9                 }
10                for (int i = 0; i<KCentroids.size(); ++i) {
11                 KCentroids[i] = sumKCentroids[i].first / sumKCentroids[i].second;
12                }
13
14               }
15          }
16
17          int next = 0;
18          for (int i = ii; i<min(shapeRows,ii+rows); ++i) {
19               for (int j = jj; j<min(shapeCols,jj+cols); ++j) {
20
21                tmp[i][j] = KCentroids[ElemsBloc[next].second];
22                ++next;
23               }
24          }
25     }
26 }
```

Figura C.3(2): Codi modificat de Caffe pel càlcul de *K-Means* en cada bloc de la matriu de pesos

Apèndixs D

Taula de calor amb els factors de compressió

D.1 Taula de calor amb els factors de compressió

	K									
N	1	2	4	8	16	32	64	128	256	512
2	4,00	1,88								
4	16,00	6,40	3,20	1,68						
8	64,00	16,00	8,00	4,57	2,67	1,52				
10	100,00	19,51	9,76	5,76	3,51	2,10	1,21			
12	144,00	22,15	11,08	6,70	4,24	2,64	1,58			
14	196,00	24,12	12,06	7,43	4,84	3,13	1,95	1,15		
16	256,00	25,60	12,80	8,00	5,33	3,56	2,29	1,39		
20	400,00	27,59	13,79	8,79	6,06	4,23	2,88	1,86	1,12	
24	576,00	28,80	14,40	9,29	6,55	4,72	3,35	2,27	1,44	
28	784,00	29,58	14,79	9,62	6,88	5,07	3,72	2,62	1,73	1,07
30	900,00	29,88	14,94	9,74	7,00	5,21	3,87	2,77	1,87	1,18
32	1024,00	30,12	15,06	9,85	7,11	5,33	4,00	2,91	2,00	1,28

Taula D.1: Taula de calor amb els factors de compressió

Referències

[1] Youn S. ; McLeod D. *Advances and Innovations in Systems, Computing Sciences and Software Engineering.: A Comparative Study for Email Classification*. Springer, Dordrecht: Elleithy K, 2017, pàg. 387 - 397. URL: https://doi.org/10.1007/978-1-4020-6264-3_67.

[2] Pazzani M.J. ; Billsus D. *The Adaptive Web: Content-Based Recommendation Systems*. Vol. 4321. Springer, Berlin, Heidelberg: Brusilovsky P., Kobsa A., Nejdl W, 2007, pàg. 325 - 341. URL: https://link.springer.com/chapter/10.1007/978-3-540-72079-9_10.

[3] Lawrence D. Jackel ; Mathew Monfort ; Urs Muller ; Jiakai Zhang ; Xin Zhang ; Jake Zhao ; Karol Zieba Mariusz Bojarski ; Davide Del Testa ; Daniel Dworakowski ; Bernhard Firner ; Beat Flepp ; Prasoon Goyal. "End to End Learning for Self-Driving Cars". A: *NVIDIA Corporation* (2016). URL: https://arxiv.org/pdf/1604.07316.pdf.

[4] T.M. Mitchell R.S. Michalski J.G. Carbonell. *Machine Learning: An Artificial Intelligence Approach*. Germany, Heidelberg: Springer-Verlag, 1983. Cap. 1.

[5] Nikhil Ketkar. *Deep Learning with Python: A hands-on Introduction*. USA: Apress, 2017.

[6] David Ha ; Douglas Eck. "A Neural Representation of Sketch Drawings". A: *Google Brain* (2017). URL: https://arxiv.org/pdf/1704.03477.pdf.

[7] O. Abdel-Hamid ; A. Mohamed ; H. Jiang i L. Deng ; G. Penn ; D. Yu. "Convolutional Neural Networks for Speech Recognition". A: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 22.10 (oct. de 2014), pàg. 1533 - 1545. URL: https://ieeexplore.ieee.org/abstract/document/6857341.

[8] Sulabh Kumra ; Christopher Kanan. "Robotic Grasp Detection using Deep Convolutional Neural Networks". A: (2017). URL: https://arxiv.org/pdf/1611.08036.pdf.

[9] Barcelona Supercomputing Center. *Documentació Minotaure*. 2019. URL: https://www.bsc.es/es/marenostrum/minotauro.

[10] Barcelona Supercomputing Center. *Documentació CTE-POWER*. 2019. URL: https://www.bsc.es/user-support/power.php.

[11] Norman P Jouppi et al. "In-datacenter performance analysis of a tensor processing unit". A: *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*. IEEE. 2017, pàg. 1 - 12. URL: https://arxiv.org/ftp/arxiv/papers/1704/1704.04760.pdf.

[12] Yangqing Jia ; Evan Shelhamer. *Caffe Documentation*. 2018. URL: http://caffe.berkeleyvision.org/.

[13] *Tècniques de creativitat: Pluja d'idees "Brainstorming"*. 2018. URL: http://www.innovaforum.com/tecnica/brain_e.htm.

[14] *Proves unitàries*. 2019. URL: https://es.wikipedia.org/wiki/Prueba_unitaria.

[15] *Metodologies àgils*. 2019. URL: https://ca.wikipedia.org/wiki/Metodologia_%C3%A0gil.

[16] *Github*. 2019. URL: http://www.github.com/.

[17] Andrew Ng. *Curso de aprendizaje automático*. 2018. URL: https://www.coursera.org/learn/machine-learning.

[18] *OverLeaf*. 2019. URL: https://www.overleaf.com.

[19] Facultat d'Informàtica de Barcelona (FIB). *Normativa de Treballs de Fi de Grau*. Oct. de 2015. URL: https://www.fib.upc.edu/sites/fib/files/documents/estudis/normativa-tfg-gei-final.pdf.

[20] *Team Gantt*. 2019. URL: https://www.teamgantt.com/.

[21] Barcelona Supercomputing Center. *Documentació MareNostrum*. 2019. URL: https://www.bsc.es/es/marenostrum/marenostrum.

[22] Agencia tributaria. *Información general sobre el IVA*. 2019. URL: https://www.agenciatributaria.es/AEAT.internet/Inicio/La_Agencia_Tributaria/Campanas/IVA/_INFORMACION/Informacion_General/Informacion_General.shtml.

[23] Michael Page. *Tendencias del mercado laboral, Tecnología*. 2018. URL: https://www.michaelpage.es/sites/michaelpage.es/files/PG_ER_IT.pdf.

[24] *PcComponentes*. 2019. URL: https://www.pccomponentes.com/.

[25] *Microsoft Product*. 2019. URL: https://products.office.com/es-es/products.

[26] *¿Cuánto consume realmente nuestro ordenador?* 2018. URL: https://hardzone.es/cuanto-consume-realmente-nuestro-ordenador/.

[27] *Precio de la electricidad en España*. 2019. URL: https://tarifaluzhora.es/factura-luz/precio-electricidad-espana/.

[28] *Preu T-Jove*. 2019. URL: https://www.tmb.cat/ca/tarifes-metro-bus-barcelona/senzills-i-integrats/t-jove/.

[29] Github. *The Legal Side of Open Source*. URL: https://opensource.guide/legal/.

[30] Yuandong Tian ; Yan Zhu. "Better computer go player with neural network and long-term prediction". A: *International Conference on Learning Representations (ICLR) 2016* (2016). URL: https://arxiv.org/pdf/1511.06410.pdf%20vom%2023.8.2018.pdf.

[31] *EU General Data Protection Regulation (GDPR)*. 2018. URL: https://eugdpr.org/.

[32] *Article 13 EU GDPR*. URL: http://www.privacy-regulation.eu/en/13.htm.

[33] *Article 14 EU GDPR*. URL: http://www.privacy-regulation.eu/en/14.htm.

[34] Patrick J Grother. "NST special database 19: Handprinted Forms and Characters database". A: *National Institute of Standards and Technology* (1995). URL: https://www.nist.gov/sites/default/files/documents/srd/nistsd19.pdf.

[35] Yann LeCun ; León Bottou ; Yoshua Bengio ; Patrick Haffner. "Gradient-based learning applied to document recognition". A: *Institute of Electrical and Electronics Engineers (IEEE)* (nov. de 1998). URL: http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf.

[36] UC Berkeley. *Github Caffe: Cifar10 Quick*. 2019. URL: https://github.com/BVLC/caffe/blob/master/examples/cifar10/cifar10_quick_train_test.prototxt.

[37] Alex Krizhevsky ; Ilya Sutskever ; Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks". A: *University of Toronto* (2012). URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[38] Princeton university. *ImageNet Database*. URL: http://www.image-net.org/.

[39] Cesar Torres-Huitzil i Bernard Girau. "Fault and error tolerance in neural networks: A review". A: *IEEE Access* (2017).

[40] Marc Ortiz ; Adrián Cristal ; Eduard Ayguadé ; Marc Casas. "Low-Precision Floating-Point Schemes for Neural Network Training". A: *Barcelona Supercomputing Center* (2018). URL: https://arxiv.org/abs/1804.05267.

[41] Taiji Suzuki ; Hiroshi Abe ; Tomoya Murata ; Shingo Horiuchi ; Kotaro Ito ; Tokuma Wachi ; So Hirai ; Masatoshi Yukishima ; Tomoaki Nishimura. "Spectral-Pruning: Compressing deep neural network via spectral analysis". A: *The university of Tokyo* (2018). URL: https://arxiv.org/pdf/1808.08558v1.pdf.

[42] Song Han ; Huizi Mao ; William J. Dally. "Deep compression: compressing deep neural networks with pruning, trained quantization and huffman coding". A: *Stanford University* (2016). URL: https://arxiv.org/pdf/1510.00149.pdf.

[43] *Deinició: Codificació de Huffman*. 2019. URL: https://ca.wikipedia.org/wiki/Codificaci%C3%B3_de_Huffman.

[44] *Definició: Funció de Hash*. 2019. URL: https://ca.wikipedia.org/wiki/Funci%C3%B3_hash.

[45] Wenlin Chen ; James T. Wilson ; Stephen Tyree ; Kilian Q. Weinberger ; Yixin Chen. "Compressing Neural Networks with the Hashing Trick". A: *Department of Computer Science and Engineering, Washington University in St. Louis, St. Louis, MO, USA* (2015). URL: http://proceedings.mlr.press/v37/chenc15.pdf.

[46] Google. *Google Profobuf documentation*. 2019. URL: https://developers.google.com/protocol-buffers/.

[47] Stanford University Jure Leskovec ; Anand Rajaraman. *Clustering Algorithms*. URL: https://web.stanford.edu/class/cs345a/slides/12-clustering.pdf.

[48] Andrea Trevino. *Introduction to K-Means Algorithm*. 2016. URL: https://www.datascience.com/blog/k-means-clustering.