
This is the **published version** of the bachelor thesis:

Armenteros Garcia, José; Serra-Sagristà, Joan, dir. Learning for compression.
2022. (958 Enginyeria Informàtica)

This version is available at <https://ddd.uab.cat/record/264172>

under the terms of the  license

Learning for Compression

Jose Armenteros García

Resum— En els últims anys, la compressió ha estat un tema molt important tant en l'àmbit més tècnic, com en l'àmbit popular. El gran volum de dades que es mou arreu del món és in comptable i les imatges són una part molt rellevant d'aquest volum de dades. Per això, des de fa anys, la investigació al voltant de la compressió ha crescut i evolucionat fins al punt de crear sistemes, com els que es mostraran, capaços de comprimir les imatges a dimensions molt petites sense perdre gairebé qualitat. Aquest projecte analitza tot un seguit de tècniques que han aparegut al llarg dels anys i sobretot en les basades en xarxes neuronals, enfocades a la compressió d'imatges.

Paraules clau— Xarxes neuronals, amb pèrdua, sense pèrdua, HiFiC, RNN, CNN, autoencoders, JPEG, JPEG200, ANN, GAN.

Abstract— In recent years, compression has been a very important topic, both in the technical and in the popular domain. The sheer volume of data moving around the world is uncountable and images are a very relevant part of this volume of data. For this reason, since a few years ago, research on compression has grown and evolved to the point of creating systems, such as the ones shown here, capable of compressing images to very small dimensions, with almost no loss of quality. This project analyses a whole series of techniques that have appeared over the years, especially those based on neural networks, focused on image compression.

Keywords— Neural networks, lossy, lossless, HiFiC, RNN, CNN, autoencoders, JPEG, JPEG200, ANN, GAN.



1 INTRODUCCIÓ

AMB l'evolució tecnològica, en concret en dispositius mòbils i més recentment en l'àmbit automobilístic, la generació, transmissió i intercanvi d'imatges ha crescut de forma exponencial, donant peu a que la quantitat de dades que es transmeten arreu del món siguin in comptables. A mesura que aquests dispositius evolucionen, més capaços són de poder capturar imatges i vídeos que s'han d'emmagatzemar localment en el dispositiu o directament es comparteixen a través de la xarxa. La pregunta llavors seria; Què passaria si no existissin mètodes per comprimir les dades?

Al llarg dels anys s'han realitzat infinitud d'estudis al voltant de la compressió, per poder determinar quina és la millor forma de poder comprimir aquestes dades, per poder emmagatzemar-les de forma que no ocupin gaire espai, que siguin fàcilment transportables a nivell de xarxa i que

no perdin qualitat per l'ull humà. Existeixen dues classes fonamentals de compressió i per les quals es classifiquen, segons si són del tipus **sense pèrdua** (*lossless*) o **amb pèrdua** (*lossy*).

El concepte *compressió de dades*, és el procés de codificació, reestructuració o modificació d'unes dades per tal de reduir la seva grandària [1]. De forma simple, es tracta de tornar a recodificar la informació utilitzant menys bits que en la representació original.

2 ESTAT DE L'ART

La compressió de dades, es pot veure com una branca de la teoria de la informació, i certament ho és, ja que es defineix com l'estudi de la codificació eficient i les seves conseqüències, en forma de velocitat de transmissió i probabilitat d'error [2].

Aquesta compressió es sol dividir en dos estats diferents. La compressió sense pèrdua, que redueix la quantitat de bits a tractar en identificar i eliminar de tot el conjunt de dades, aquells que produeixen una redundància estadística ¹ [3].

¹Redundancy

Diferència fraccionària entre l'entropia d'un conjunt de dades i el seu valor

• E-mail de contacte: jose.armenteros@autonoma.cat
• Menció realitzada: Tecnologies de la Informació
• Treball tutoritzat per: Joan Serra Sagrista
• Curs 2021/2022

Un exemple és el format *PNG*, que ofereix opcions de compressió sense pèrdua 1.

En canvi, en la compressió amb pèrdua, es redueix la quantitat de bits, en eliminar informació no necessària o poc important [3]. Un exemple clar de codificació amb pèrdua, va sorgir amb la implementació del format d'arxius *.mp3* en àudio i *.jpeg* en imatges, on per exemple, en el cas de *.mp3*, les freqüències per sobre i per sota del llindar de l'oïda, s'eliminen per tal de reduir l'espai que ocupa 1.



Fig. 1: lossless vs lossy.

2.1 Començaments. Anys 1950 a 1980

L'inici del concepte de xarxes neuronal aplicat a la informàtica, no va aparèixer fins a l'any 1943, quan dos neuròlegs dels Estats Units, en *Warren McCulloch* [4] i en *Walter Pitts* [5], van crear el primer model informàtic per a xarxes neuronals, anomenat **lògica llindar** [6] basat en fórmules matemàtiques i algorismes. A finals de la dècada dels quaranta, el tan refutat psicòleg *Donald Hebb* [7], va crear una hipòtesi d'aprenentatge basat en el mecanisme de plasticitat neuronal, el qual actualment es coneix com a l'**aprenentatge de Hebb** [8]. No va ser fins a l'any 1958 quan **Frank Rosenblatt**, psicòleg de la Universitat de *Cornell* va crear el primer algorisme de reconeixement de patrons basat en una xarxa d'aprenentatge de dues capes, d'addició i sostracció. Aquest invent és l'anomenat **Perceptró** 2. Aquest va ser concebut com un model matemàtic de com funcionen les neurones en el nostre cervell. En l'aplicació d'aquesta idea és on *Rosenblatt* va diferenciar-se del model de *McCulloch-Pitts*, incapaç d'aprendre per si sol i que va assentar les bases dels models futurs.

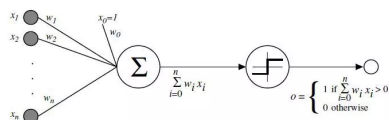


Fig. 2: Diagrama de funcionament del Perceptró.

Rosenblatt, va implementar la idea del **Perceptró** i va demostrar que podria utilitzar-se per aprendre a classificar formes simples. Aquest va ser el naixement del que avui es coneix com a intel·ligència artificial [9]. Una de les següents idees va ser l'ampliació de les capes, que més tard va donar forma a les tan conegudes *ANN* o *Xarxes neuronals artificials*, que no deixen de ser capes de *perceptrons*, avui dia anomenades neurones.

Encara els avenços, no deixaven de ser funcions d'activació lineals que només poden generar funcions lògiques, amb "1/0". La idea estava clara, connectar unitats neuronals simples per tal de formar xarxes més grans.

2.2 Renaixement. Anys 1980 a 2000

El següent avenç clau a la història, es va produir l'any 1974, quan el científic *Paul Werbos* [10], va presentar a la seva tesi doctoral, l'aplicació del procés de retropropagació per a l'ús en xarxes neuronals. El concepte de capes ocultes, prové de la idea que en *Rosenblatt* va encaminar en el seu estudi. Enviar la informació de les neurones d'entrada a moltes altres neurones de forma arbitrària, creant així una capa oculta, com es mostra a la figura 3.

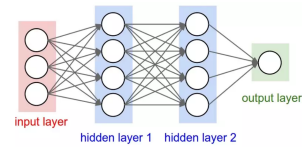


Fig. 3: Representació de capes ocultes.

La raó principal per la qual les capes ocultes són especialment bones, és perquè permeten classificar. Unes capes troben característiques dintre de les dades i permeten a les següents capes operar sobre aquestes característiques ja extretes, podent crear xarxes amb moltes capes.

Poc anys després, un grup encapçalat per *Yann LeCun*, van demostrar com una màquina mitjançant l'algorisme de retropropagació i l'ús de xarxes neuronals multicapa [11], era capaç de reconèixer correctament els díigits escrits a mà alçada. Tant va ser l'èxit que es va convertir en la base dels sistemes de lectura de xecs bancaris als Estats Units.

A més, en el mateix article, van ser els primers a definir de forma clara, la primera **xarxa neuronal convolucional**, avui conegut com a **Deep Learning**.

Tornant a les xarxes neuronals multicapa, aquestes relacionen cada píxel a una variable, dividint la imatge en tantes variables com píxels tingui i per conseqüència, traient importància a on es troba cada píxel dins la imatge global 4. Però, al contrari, en les xarxes neuronals convolucionals, s'elimina la idea d'independència de píxels, donant importància on es troba cada píxel. Per fer-ho, s'aplica una operació matemàtica anomenada **convolució**, on cada píxel de la nova imatge, es generarà, mitjançant la col·locació d'una matriu de valors, anomenada *kernel*, sobre la imatge original, i sumant i/o multiplicant els valors de cada píxel veí, aconseguirem el nou valor del píxel. Desplaçant el filtre per tota la imatge, construirem una nova imatge, anomenada mapa de característiques.

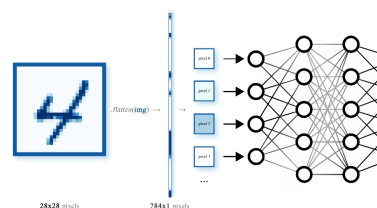


Fig. 4: Representació dades entrada en xarxes multicapa.

màxim possible. És a dir, la quantitat d'espai no aprofitat per enviar dades.

El mètode per excel·lència a l'hora de realitzar aquest entrenament és l'anomenat **descens del gradient**. Aquesta és una de les eines matemàtiques més importants del camp de la IA i és la responsable del procés d'entrenament que succeeix dintre de les xarxes neuronals. La idea bàsica és calcular el pendent mitjançant la derivada per trobar el punt que optimitzi al màxim la seva funció d'error.

Poc temps després, l'any 1992, va aparèixer el concepte de **max-pooling** o submostratge, per tal de reduir la complexitat computacional de la xarxa, mitjançant un procés de discretització, reduint la seva dimensionalitat.

La primera idea de **Codificadors Automàtics**, va sorgir en el tan conegut article de l'any 1986 sobre la retropropagació [12]. Aquest tipus xarxa no supervisada és molt útil, a l'hora de trobar patrons o característiques de la imatge. La capa de sortida, calcula aquesta sortida utilitzant únicament la sortida de la capa oculta. Donat que aquesta capa oculta té menys sortides que la capa d'entrada de la xarxa, la sortida de la capa oculta serà la representació comprimida de les dades 5.

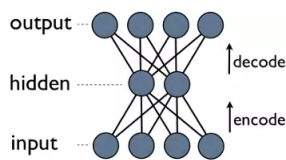


Fig. 5: Representació gràfica d'un auto-codificador.

Pel que fa a, l'*aprenentatge automàtic supervisat*, necessita un conjunt d'entrenament on les dades estiguin etiquetades *labeled data*, en canvi, l'*aprenentatge automàtic no supervisat*, només necessita un conjunt de dades d'entrada, *input*, per poder trobar alguna estructura oculta dintre d'aquestes dades. El tercer tipus d'aprenentatge, és el de **per reforç**. El seu objectiu és ser capaç de decidir una acció en funció del seu estat, sent un sistema retroalimentat [13].

Les xarxes comentades anteriorment, tenen en comú que el seu flux intern és cap endavant. Aquesta, és la característica principal de les **Xarxes Neuronals Recurrents (RNN)** 6. Pel que fa a les seves funcionalitats, són especialment bones en anàlisi, predicció i reconeixement de seqüències. El principal avantatge és poder aprendre com extreure informació rellevant i utilitzar-la com a part de les dades d'entrada. La solució per poder reduir l'error es troba en la tècnica basada en **gradients** de **retropropagació a través del temps** o **BPTT** [14] [15].

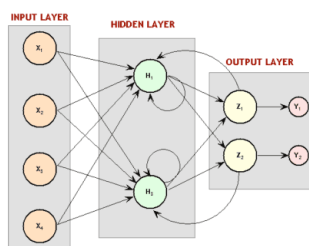


Fig. 6: Xarxa neuronal Recurrent.

Aquesta idea d'aprenentatge de les RNN, tenia algunes limitacions amb les dependències a llarg termini [16] i no

va ser fins a l'any 1997 [17], que es va introduir un concepte molt important, anomenat **memòria de llarg termini (LSTM)**. A diferència de les RNN, les LSTM tenen la capacitat per la seva estructura interna, de poder aprendre dependències llargues, 7.

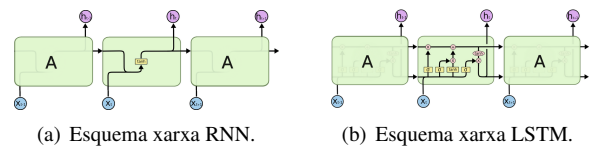


Fig. 7: Comparació d'esquemes de xarxa.

2.3 Aprenentatge profund. Anys 2000 a l'actualitat

Encara els avenços, la retropropagació no funcionava bé com a mètode d'entrenament per xarxes amb moltes capes i no va ser fins a l'any 2006, amb l'article *A fast learning algorithm for deep belief nets* [18], el qual va donar peu al que avui es coneix com a aprenentatge profund. L'estudi de Hinton, amb les màquines RBM [19], va donar com a resultat que les xarxes DBN tenien un rendiment molt superior en el conjunt de dades de reconeixement de caràcters de la base de dades MNIST², superant significativament a altres xarxes neuronals, amb tan sols poques capes.

Yoshua Bengio, va continuar el treball, i l'any 2007, va publicar un estudi [20], on proposava la utilització d'autocodificadors per suplir el preentrenament, demostrant la seva eficiència per sobre d'altres mètodes.

Els avenços en els algorismes i en els conjunts de dades van ser un pas molt important en la IA, però els resultats esperats encara no arribaven. La clau es trobava en el poder computacional pur. L'any 2009, George Dahl, estudiant en el laboratori de la Universitat de Toronto, precisament on era supervisor Hinton, va descobrir com entrenar i simular xarxes neuronals de forma eficient utilitzant les mateixes targetes gràfiques.

Abans d'aprofundir en la utilització de xarxes neuronals per la compressió d'imatges, cal diferenciar alguns termes. Actualment, els termes d'aprenentatge profund o *deep learning* i aprenentatge automàtic o *machine learning*, se solen intercanviar i confondre. El que s'anomena *deep learning* no deixa de ser un subconjunt concret del *machine learning* que a la vegada ho és de la intel·ligència artificial [21]. A la següent imatge 8 queda molt més clar l'idea.

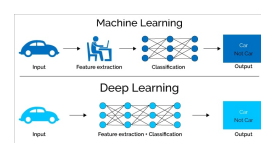


Fig. 8: Machine Learning VS Deep Learning.

²MNIST

És una gran base de dades on es troben dígits escrits manualment, que s'utilitzen per preparar sistemes com en el cas de les xarxes neuronals.

Retornant a la compressió d'imatges, la utilització de tècniques basades en xarxes neuronals, s'han utilitzat històricament per a la compressió, incloent-hi tècniques com *perceptrons multicapes*, *xarxes neuronals aleatòries*, *xarxes neuronals convolucionals*, etc. Centrant-se en les anomenades **CNN**, aquestes s'han fet servir per arquitectures de reconeixement i compressió d'imatges principalment.

Existeixen molts algoritmes capaços de poder comprimir imatges. El problema és que la reconstrucció no és tan simple, per això, el desenvolupament de xarxes neuronals convolucionals d'extrem a extrem capaces de comprimir i descomprimir, s'han convertit en la referència en l'àmbit [22]. L'any 2017, un conjunt de desenvolupadors de descendència asiàtica [23], varen proposar un model de compressió format per dues CNN i un còdec d'imatge, com es mostra a la següent imatge 9. La primera CNN anomenada, *ComCNN* o xarxa neuronal convolucional compacta, s'encarrega de fer el procés de compressió i la segona CNN, anomenada *RecCNN* o xarxa neuronal convolucional de reconstrucció, s'encarrega del procés invers.

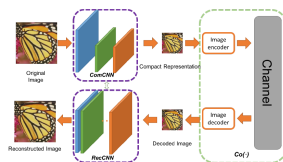


Fig. 9: Xarxa neuronal convolucional d'extrem a extrem.

3 OBJECTIUS

Per poder definir els objectius de forma correcta, primer es van definir els que es consideraven com a necessaris per a la realització del treball i que posteriorment s'han afegit de nous, i els que eren importants de tractar, per donar millor consistència al treball i oferir un punt de vista més ampli al lector.

3.1 Objectius Principals

Pel que fa als objectius principals podem descriure'ls com els següents:

1. Realització d'un estudi, sobre l'estat de l'art en tècniques de *deep learning*, *machine learning* i xarxes neuronals.
2. Recerca especialitzada en l'aplicació de les xarxes en la compressió i detecció d'imatges.
3. Recerca de tècniques neuronals de compressió amb pèrdua i sense.
4. Com aplicar aquestes tècniques esmentades en l'estat de l'art, per la compressió de dades i/o imatges.
5. Presentar comparacions com a resultat de les diferents tècniques de compressió aplicades.

3.2 Objectius Secundaris

1. Recerca d'altres tècniques de compressió no neuronals.
2. Adopció o modificació d'algun model pràctic per tal de demostrar el seu funcionament.

4 METODOLOGIA I PLANIFICACIÓ

Pel que fa a la metodologia de treball, en tractar-se d'un projecte individual, l'aplicació d'algunes metodologies àgils, com **Scrum**³, **DevOps**⁴ o **XProgramming**⁵, no s'adapten, ja que requereixen diferents rols per dur a terme de forma correcta la seva implementació i per conseqüència en aquest cas no és una metodologia apta.

Per altra banda, existeix la metodologia de treball **Kanban**, que sí que s'adapta més a la forma de treball que s'utilitzarà en aquest projecte, ja que a diferència de les altres mencionades, no cal l'existència de rols diferenciats, com podria ser un Product Owner o un Scrum Master, etc. És per això, que finalment s'ha adoptat per escollir Kanban, com a metodologia de treball en la realització del projecte.

La metodologia *Kanban*, és un mètode on es busca l'eficiència del treball delimitant els estats de saturació o de massa treball, per evitar colls d'ampolla en la realització del projecte [24], ja que per petits projectes és la que millor s'adapta.

Per tal de poder crear aquesta planificació, s'utilitzarà el programari web anomenat Trello [25], on es definiran els diferents estats 10:

- **Backlog:** Es recolliran totes les tasques que s'hauran de fer durant el projecte (degut a la forma de funcionament del model i per facilitar el treball, es podran afegir algunes tasques menys primordials a posteriori).
- **Doing:** Es definirà la tasca sobre què s'està treballant actualment.
- **Done:** Serà l'últim estat, i ens definirà que la tasca ha estat completada satisfactòriament.
- En el cas de la part pràctica, s'ha afegit l'estat de **Testing**, on es provarà el funcionament.

³Scrum

Es denomina com un marc de treball col·laboratiu entre equips. La seva planificació es basa en la divisió del projecte en petits blocs o *sprints*, que es planifiquen i revisen contínuament al transcurs del projecte.

⁴DevOps

Es caracteritzen per agrupar el desenvolupament de *software* i operacions de TI per tal de maximitzar el cicle de vida del codi o programa i la seva qualitat.

⁵XProgramming

A diferència de les altres metodologies de treball, aquesta se centra en la comunicació i sobretot en el reaprofitament de codi ja escrit o utilitzat en altres projectes, de forma que no s'hagin de crear de nou i per conseqüència s'estalvia temps.

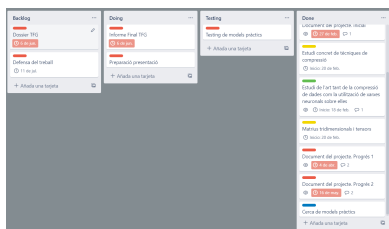


Fig. 10: Planificació del projecte

5 DESENVOLUPAMENT

5.1 Amb pèrdua - Lossy

La utilització d'algoritmes de compressió amb pèrdua, implica una eliminació d'informació de l'arxiu que es tracta i a diferència dels algoritmes sense pèrdua, aquesta informació no es pot tornar a recuperar ¹. A canvi, d'acceptar una distorsió de la realitat, es poden obtenir relacions de compressió molt més grans. En moltes aplicacions, aquesta pèrdua d'informació no és important o prou visible per donar-li importància i, per tant, la consistència de les dades no ha de ser perfecte.

Alguns exemples d'algoritme de compressió amb pèrdua són:

- **Chroma subsampling:** En disminuir informació del color (*croma*) emmagatzemat en la imatge, en comptes de la lluentor, es redueix l'espai de l'arxiu. Com que l'ull humà és més sensible a la luminància que als colors, aquests es poden sacrificar si la lluentor de la imatge s'ajusta en conseqüència ^[26].
- **Color reduction:** La quantitat de colors que es guarda en una imatge ocupa espai, però molts cops es pot reduir aquesta quantitat sense alterar la imatge, com seria el cas del cel, en el que es redueix la quantitat de tonalitats de colors als més utilitzats.
- **Fractal compression:** A causa de la semblança entre fraccions d'una mateixa imatge, s'agafen aquestes parts semblants i mitjançant l'aplicació d'un algoritme fractal, es dupliquen aquestes parts ^[27].
- **Transform coding:** Semblant al *Color reduction*, es fa una mitjana dels colors per així utilitzar menys colors. També, s'aprofita que l'ésser humà, no és tan perceptible a la majoria de colors i, per tant, eliminant-los, es redueix l'espai ocupat ^[28].

Dintre d'aquestes quatre tècniques de compressió, la més estesa i utilitzada és la denominada **Transform coding**, ja que dintre s'inclouen les conegudes com a **Transformada de cosinus discreta** o DCT i la **Transformada de Wavelet**, corresponents als formats JPEG ⁶ i JPEG2000 ⁷.

⁶JPEG

Mètode de compressió basat en la transformada discreta del cosinus.

⁷JPEG2000

Al contrari que el seu predecessor, es basa en la transformada de *wavelet*, cosa que fa suavitzar les imatges per tal de comprimir-les.

5.2 Sense pèrdua - Lossless

A diferència dels mètodes de compressió amb pèrdua, aquest no comporten pèrdua d'informació i, per tant, les dades es poden tornar a recuperar completament. Generalment, aquests mètodes s'utilitzen per aplicacions en què no es pot tolerar cap mena de diferència entre les dades originals i les reconstruïdes, com en el text o l'àmbit mèdic i radiològic o fins i tot en astrofísica per exemple.

Alguns exemples d'aquestes tècniques són:

- **Huffman Coding:** També anomenat codificació de longitud variable, utilitza un esquema de tipus arbre en què els caràcters s'emmagatzemen inicialment com a fulles de l'arbre. La freqüència de cada símbol dintre de l'arxiu es converteix en el seu pes. Un cop definit l'arbre, s'emparellen les fulles de pes més baix creant branques connectades a un nus superior amb el pes total de les fulles. Un cop agrupat, les branques de l'esquerra es defineixen amb un '1' i les de la dreta amb un '0'. D'aquesta forma cada fulla es codifica amb una sèrie de bits des del node arrel fins a la fulla específica ^[29].
- **Arithmetic Coding:** La codificació aritmètica és una forma de codificació d'entropia on una cadena entrant, es representa utilitzant un nombre fixe de bits per caràcter, semblant a *ASCII*⁸. Quan aquesta cadena entrant es codifica, els caràcters més freqüentment utilitzats s'emmagatzemaran amb menys bits i els caràcters menys freqüents, amb més bits, de forma que obtenim una disminució de l'espai ocupat per aquesta cadena ^[30].
- **Run-Length Encoding:** És probablement la forma de compressió més simple però amb bons resultats en representacions gràfiques. La idea es basa en el fet que les sèries de dades (seqüències en les que el mateix valor apareix en molts elements consecutius) s'emmagatzemen com un únic valor de dada i es conti quants cops es repeteix aquest valor. D'aquesta forma es pot arribar a reduir de forma considerable un fitxer sense perdre cap informació ^[31].
- **Lempel-Ziv-Storer-Szymanski:** Més coneguts com a **LZ77** i **LZ78**, van ser els dos primers mètodes de compressió sense pèrdua basats en codificadors de diccionari. **LZ77**, analitza el text com una seqüència de caràcters, mitjançant l'anomenada finestra lliscant, composta per dues parts (un *buffer* d'anticipació i per un *buffer* de cerca) ^[32]. Després d'algunes crítiques, els mateixos creadors, van donar pas al conegut com a **LZ78** ^[33], el qual utilitza dues variables [**Count**, **Offset**]. El seu funcionament es basa en comprimir dades seqüencials mitjançant la creació de seqüències de *tokens*. Un exemple seria, la següent cadena "AAB-BA" de la qual es crearia el següent diccionari; Entrada 0, 0,_, entrada 1, 0,A, entrada 2, 1,B, entrada 3, 0,B. Aquests algoritmes són realment d'utilitat quan la quantitat de dades és elevada.

⁸ASCII

És una codificació de caràcters utilitzada en la comunicació electrònica i és d'on es basen la majoria d'esquemes de codificació de caràcters més moderns.

D'entre les tècniques de compressió sense pèrdua exposades anteriorment, les que més s'utilitzen són la codificació de *Huffman* i tots els derivats de *LZ77* i *LZ78*. En aquest segon cas, existeixen molts derivats i un d'ells, el *LZW* [34], deriven formats d'imatge tan coneguts com, **GIF** i **TIFF** i fins i tot **PDF**. L'altre format d'arxius relacionat amb les tècniques comentades anteriorment, és el tant conegut **PNG**. El qual en la fase de compressió fa servir un model anomenat **DEFLATE** [35], que utilitza una combinació entre *LZ77* i *Huffman* per dur-la a terme.

5.3 Neuronal Lossy Compression

La base de la creació d'algoritmes de compressió amb pèrdua amb xarxes neuronals, és, poder aprofitar algunes limitacions que tenen els éssers humans en la visió. La majoria, utilitzen aquesta limitació per utilitzar mètodes basats en blocs. Segons més petits són aquests blocs millor definició tindrà, però pitjor compressió obtindrem. Alguns mètodes de compressió basats en blocs són els ja anomenats *auto-codificadors* i les *xarxes generatives antagòniques* o també anomenades *GAN*.

- **Compressió mitjançant Auto-Codificadors:** Els auto-codificadors, com ja hem comentat anteriorment, són un tipus de xarxa que intenten representar la imatge amb una disminució del senyal original. Aquest procés de reducció del senyal original, es pot aconseguir reduint la dimensionalitat, però tenint en compte, que es puguin recuperar els senyals originals, a partir de la dimensió reduïda amb un petit error. Un model de codificació automàtic, es divideix en dues subxarxes diferenciades, la xarxa de codificació i la xarxa de descodificació, com es pot veure a l'anterior figura 5. Existeixen diferents tipus de codificadors automàtics. Els anomenats, *de convolució* o **CAE**, els quals aprofiten el fet que un senyal pogués veure's com una suma d'altres senyals. Aquests, utilitzen l'operador de convolució per aprendre a codificar l'entrada amb un conjunt de senyals més simples, per a després, reconstruir-les. El problema principal dels codificadors automàtics de convolució, és causat per la xarxa de descodificació, ja que fa falta utilitzar un operador de desconvolució, però l'operador de convolució no és injectiu⁹ i, per tant, no existeix l'operador invers com a tal, encara que, sí que existeix un mètode que utilitza una funció d'interpolació seguida d'un operador de convolució per suplir-ho.

L'altre tipus d'autocodificador, és el **Deep Autoencoder** basat en capes. La primera capa, per les característiques de primer ordre en l'entrada sense processar. La segona capa, les de segon ordre, pel reconeixement de patrons sobre les característiques de primer ordre i així successivament. Una de les seves aplicacions és la variació de característiques. Això permet eliminar el soroll d'una imatge d'entrada i extreure la imatge sense soroll. També, l'aplicació de la reducció de la dimensionalitat, que és potser la més interessant,

on la imatge reconstruïda, és la mateixa que la imatge d'entrada, però amb dimensions reduïdes i aprofitant les limitacions de visió de les persones, podem reduir el valor del píxel, sense que la diferència sigui apreciable a simple vista.

- **Compressió mitjançant xarxes GAN:** Des de la seva aparició l'any 2014, per *Ian Goodfellow* i altres companys seus [36], les anomenades xarxes generatives antagòniques, han portat un fort i ràpid progrés. La principal particularitat és que són capaces de generar imatges foto-realistes amb una alta resolució. El seu funcionament es basa el joc anomenat suma zero¹⁰. Donat un conjunt d'entrenament, aquesta tècnica és capaç d'aprendre a generar noves dades amb les mateixes estadístiques que el conjunt d'entrenament. La idea és tindre una xarxa generativa (xarxes neuronals deconvolucionals), que s'encarrega de generar els candidats, mitjançant un model generador estadístic, mentre que l'altra xarxa, la discriminatòria (xarxes neuronals convolucionals), examina i avalua als candidats. D'aquesta forma, com en les dues xarxes s'aplica el mètode de retropropagació, cada cop s'igualen més a 0.

Però, no va ser fins a l'any 2017, quan la utilització de xarxes GAN va fer el gran salt. Companyies com *Samsung* o *Nvidia*, van crear sistemes basats en xarxes GAN capaços de realitzar coses abans impossibles [37] [38].

A més, existeixen uns models anomenats, *High-Fidelity Generative Image Compression*, els quals comentarem a l'apartat de resultats 6, basats en xarxes GAN i que són capaços d'obtenir amb la mateixa compressió que mètodes com *JPEG*, un nivell de detall 13 vegades millor [39]. A la següent imatge, es pot veure aquesta comparativa 11, on ocupant les dues imatges al voltant dels 75kb, la imatge en *JPEG* té una qualitat molt baixa i la comprimida amb *HiFiC* té una qualitat molt més alta.

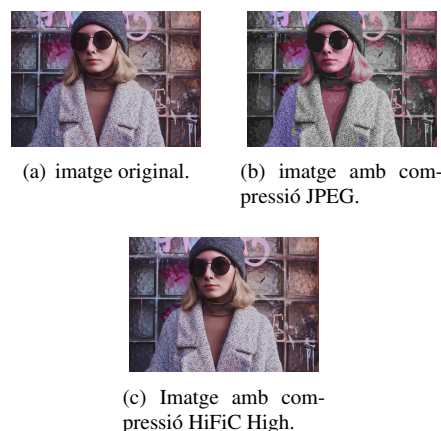


Fig. 11: Comparació d'imatges segons la compressió

⁹Injectiu

Una funció és injectiva, un a un, si a diferents elements d'un grup X , li corresponen elements diferents d'un altre conjunt Y .

¹⁰Joc de suma zero

Representació matemàtica d'una situació on dos costats lluiten per obtenir un avantatge sobre l'altre, i la finalitat és poder apropar-se el màxim possible a 0.

5.4 Neuronal Lossless Compression

Al contrari que passa amb la compressió amb pèrdua, en els mètodes de compressió sense pèrdua, l'important és poder recuperar tota la informació de la imatge original, un cop aquesta ha estat comprimida. Per això, la seva aplicació està més enfocada a àries com la medicina, per radiografies (RX), TACs o ressonàncies magnètiques RM, o, per altra banda, per l'astronomia. Alguns dels mètodes de compressió sense pèrdua són la codificació predictiva mitjançant xarxes *MLP*.

- Les xarxes *MLP*, són un model de xarxa neuronal que funciona com aproximadors universals, és a dir, poden aproximar qualsevol funció contínua, a diferència dels perceptrons unicapa, que només poden aproximar funcions lineals. Aquesta aproximació és dur a terme gràcies a la utilització de com a mínim tres capes diferents. La primera capa, és la capa d'entrada, que s'encarrega de distribuir les entitats d'entrada a la primera capa oculta de la xarxa. La segona capa, és considerada la part de les capes ocultes, on podem trobar una o més capes ocultes de perceptrons. La primera capa, rep com a entrada, les característiques distribuïdes per la capa d'entrada, i les altres capes ocultes, reben com a entrada, la sortida de la capa anterior. Finalment, l'última capa, la de sortida, que rep com a entrada, la sortida de l'última capa oculta de perceptrons. A la figura 3, es pot veure de forma esquemàtica.

Les xarxes basades en *MLP*, entren en la categoria d'algoritmes *feedforward*, perquè les entrades es combinen amb els pesos inicials en una suma ponderada i se sotmeten a una funció d'activació. La característica principal, és que cada combinació lineal, es propaga a la següent capa, donant-nos la capacitat de poder resoldre qualsevol funció continua, com hem comentat.

6 RESULTATS

Pel que fa als resultats, l'estudi s'ha centrat en la cerca de mètodes de compressió amb pèrdua i més concretament, en comparar mètodes clàssic enfront mètodes moderns on s'aplica l'ús de les xarxes neuronals.

El cas que es presenta a continuació ha estat una comparació entre la compressió d'imatges amb el codificador JPEG i amb HiFiC (compressió d'imatges d'alta fidelitat), basat en xarxes neuronals generatives, o *GAN*. A la següent taula 1, trobarem els resultats que comentarem a continuació. Les imatges que s'han utilitzat per dur a terme les proves les podem trobar en els següents enllaços, [Data set image comparison](#), [Kodak Dataset image](#) i al següent repositori [GitHub](#), codis escrits en *python* que s'han utilitzat per calcular algunes dades. A la següent taula 2, trobem les imatges que s'han fet servir per fer les proves i treure les conclusions, amb les seves especificacions. Dimensió en píxels i espai ocupat en format *RAW*. Com es tracten d'imatges amb tres canals *RGB* amb una profunditat de 8 bits per canal, és a dir, en total 24 bits per píxel. Cal destacar, que les proves han estat realitzades sota la

plataforma d'execució de [Google Colab](#) per tal de dur-les a terme de la millor forma possible i més ràpida, ja que permet l'execució de gairebé qualsevol programa, sempre i quant, els recursos ho permetin, al núvol i sense haver d'instal·lar o preocupar-se per temes de compatibilitat de llibreries en màquines locals. Ha estat escollida també, gràcies, tant al seu *hardware*, la qual ofereix una tarjeta gràfica *Tesla K80* de 12 GB de VRAM o una *Tesla T4* de 16 GB de VRAM, i un processador Intel(R) Xeon(R) CPU @ 2.30GHz amb una memòria RAM màxima utilitzable d'entorn al 13 GB en les proves realitzades i un disc dur SSD d'uns 80 GB, dels quals queden uns 40 aprofitables per poder pujar arxius, instal·lar llibreries, etc, com tant, per la facilitat de poder treballar amb la plataforma.

Com s'ha esmentat, els mètodes de compressió escollits per comparar són *JPEG* i *HiFiC*, en dues variants diferents. Aquest últim ja prèviament entrenat, per estalviar temps i recursos. Pel que fa al mètode de compressió *HiFiC*, en la següent taula 1 veiem com s'ha diferenciat en tres tipus de compressió. La compressió de tipus *low* o baixa, és el model amb la relació de compressió més alta, i per això trobem que tant la taxa de bits o *bit per píxel* com l'espai que ocupa el fitxer comprimit és el menor. Al contrari, el tipus *high*, ens ofereix una pitjor compressió, per tant, la imatge ocuparà més espai, però obtindrem una millor representació i definició de la imatge, bastant semblant a l'original i en tots els casos, amb una relació d'espai qualitat molt superior que *JPEG* i *PNG*. A més, també tenim una opció intermèdia, anomenada *middle*, la qual busca balancejar de la millor forma possible el nivell de qualitat de la imatge, per l'espai ocupat.

Si comparem la imatge amb una de menor taxa de compressió i, per tant, més *bpp*, veurem algunes diferències entre elles 12, per exemple la definició de les cares o el nom "repeat" que hi ha escrit al costat del vaixell o el fons del pla amb una menor definició. Tot això fa que l'espai ocupat sigui menor, però per contra, es perden detalls. En aquest cas en concret, en tractar-se d'una imatge no gaire gran, de 512 x 768 px, les diferències visualment no són excessivament apreciables, però si agafem una imatge més gran, sí que veurem més diferències. Si ens fixem en el següent exemple, 13, veiem com en aquest cas sí que estem tractant amb una imatge gran de 1536 x 2048 px. Si comparem la imatge original amb la imatge comprimida amb l'algoritme *HiFiC* de tipus baix, veurem com les diferències són gairebé inexistents i és que, gràcies al gran conjunt d'entrenament amb el qual s'ha entrenat aquest algoritme, la xarxa neuronal *GAN* és molt solvent a l'hora de treballar amb imatges on les formes són relativament rectes, com és el cas, ja que gairebé totes les formes són quadrades i no tenen gaires detalls, com si passava en el cas del vaixell, amb el nom, per exemple. Un últim exemple, el trobem amb les següents imatges 14. Si ens fixem en la imatge original o de compressió *HiFiC High* en comparació amb la de tipus *low*, es veu molt clar com la definició es veu afectada a favor de l'espai que ocupa, ja que, per exemple, el nom de la marca de la moto groga no es pot llegir, o tampoc el nom de la marca de les ulleres del pilot de casc vermell. En definitiva, podem dir que encara que es perdi definició de la imatge, el guany en comparació amb una imatge de la mateixa grandària, però en format

JPEG és molt alt.

Finalment, si fem una comparació entre els dos models de compressió que es mostren a la taula 1, veurem com, encara que estan relativament parells, el model *HiFiC* obté una millor definició que el model *HiFiC-pytorch*, tant en la compressió de tipus *high*, com es pot veure a la següent imatge 15, on podem apreciar com el nom del vaixell es veu amb més qualitat, els números de les veles del vaixell també i dels fons de la imatge, les formes són més visibles en el model *HiFiC*, tant en la compressió de tipus *low*, com es pot apreciar a les següents figures 16, on veiem més clarament aquesta diferenciació de definició, ja que el model implementat en *Pytorch*, suavitzava més les formes, cosa que pot anar bé quan es tracta d'imatges de relleus, com en el cas d'un paisatge 17, en què la diferència no és tan apreciable com en una imatge amb formes molt marcades. També, es pot veure de forma clara, en la següent comparació 18, on podem veure que la diferència més notable a simple vista entre la imatge original i les comprimides és la llum i la tonalitat de colors, però gairebé ni el relleu de la muntanya, ni el camp ni la tenda de campanya que surt en primer pla, es veuen afectades per aquesta suavització, que a simple vista no és apreciable.

Un cop vists els resultats obtinguts, comentarem quines són les diferències que trobem entre dues de les llibreries més populars de *Python* a l'hora de treballar en projectes de *Maching Learning*, encara que avui dia, existeix una altra anomenada *JAX* que està obtenint una molt bona acollida en la comunitat de desenvolupadors. Primerament, pel que fa a la llibreria de *Tensorflow*, aquesta va ser creada per l'empresa *Google Inc.* l'any 2015 [40] i inicialment va ser desenvolupada per poder solucionar tants problemes d'investigació, com de producció interna de *Google* i sempre s'ha associat a un ús més de producció. Aquesta primera, és la que utilitza el model que hem anomenat a la taula 1 com **HiFiC**. Per l'altra part, *Pytorch* va ser desenvolupat per l'empresa *Facebook* i tret al servei de la comunitat l'any 2017 [41] i encara que també és utilitzada per grans empreses com *Tesla* pel seu *software Autopilot*, s'ha associat més a treballs d'investigació, i l'utilitza l'altre model de compressió **HiFiC-pytorch**.

Pel que fa a la funcionalitat i els marcs de treball tenen diferenciacions per exemple, en la utilització dels recursos de la *GPU*. En el cas d'utilitzar les llibreries de *Pytorch*, sobretot a l'hora d'entrenar un model, es dona el cas que es pot millorar aquest procés, utilitzant la *GPU* a l'hora que s'executa codi en *CUDA*¹¹, a més, també ofereix una altra característica diferenciadora, i és l'optimització del rendiment aprofitant la compatibilitat nativa amb l'execució asíncrona de *Python*, en canvi, a *Tensorflow*, aquesta compatibilitat s'haurà d'ajustar i codificar manualment, cosa que dificultarà el treball i, pel que fa a l'ús de la *GPU*, es pot accedir a les *GPUs*, però no permet modificar aquesta acceleració, ja que ve definida. És per això, que tant el temps d'execució, com el d'entrenament, com els resultats finals poden variar i diferir uns dels altres, encara

que el producte final sigui i realitzi la mateixa funció, com es dona en el cas anterior.

Per poder quantificar les diferències que s'han remarcat anteriorment, existeixen diferents mètriques. Aquestes ajuden a donar un valor numèric a la percepció, ja que aquesta és molt relativa. Una de les mètriques més interessants a l'hora de poder quantificar diferències entre imatges és l'anomenada **PSNR** o *Peak Signal to Noise Ratio*. Aquesta mètrica és comunament utilitzada en l'àmbit de l'enginyeria per definir la relació entre la potència màxima possible d'un senyal i el soroll que afecta la seva representació fidedigna.

L'ús més habitual del **PSNR**, és com a mesura quantitativa de la qualitat de reconstrucció i ens dona una solució matemàtica a com de bé o malament està la imatge reconstruïda sobre la imatge original. En el cas que estem tractant, el senyal entrant, serà la imatge original que introduïm a l'algoritme de compressió, i el soroll, és l'error introduït a la imatge a causa d'aquesta compressió. Els valors típics per les mesures del *PSNR* amb pèrdua, oscil·len entre 30 i 50 dB, sempre que la profunditat de cada canal sigui de 8 bits, com en aquest cas i com més alt sigui aquest valor, la diferència entre l'original i la comprimida serà menor. El problema d'aquest tipus de mètrica, és que la seva comparació és a baix nivell, és a dir, es pot donar el cas que encara que el **PSNR** tingui un valor baix, com algun dels nostres exemples, la impressió a simple vista per l'ull humà, no concordi amb aquesta mala reconstrucció, ja que en aquest cas, entren en joc, les limitacions dels éssers humans, a l'hora de percebre les imatges i com les compressions es poden adaptar per trobar una representació que per l'ull humà sigui bona, però que tingui un dolent **PSNR** [42].

Com hem vist, els tipus de mètriques com **PSNR**, ofereixen una relació senyal - soroll que ens pot donar una referència, però no té en compte la percepció, al contrari que el mètode **SSIM** o *mètode d'índex de similitud d'estructura*. Aquesta nova eina de mesura va sorgir l'any 2004 [43] de la mà dels investigadors *Hamid Sheikh* i *Eero Simoncelli*. **SSIM**, és un model basat en la percepció que considera la degradació de la imatge, com un canvi percebut en la seva informació estructural, alhora que també incorpora l'emascament de luminància, on les distorsions de les imatges produïdes per la compressió, tendeixen a ser menys visibles per l'ull humà en les regions més brillants i més fosques i l'emascament de contrast, on les distorsions es tornen menys visibles en zones o regions de la imatge on hi ha una alta activitat o textura en la imatge.

Aquesta informació estructural és la idea que els píxels tenen una forta relació o interdependència, especialment quan es troben a prop uns dels altres i això ens dona informació important de l'estructura dels objectes, les escenes, etc., cosa que es pot aprofitar i que els algoritmes de compressió aprofiten per reduir l'espai ocupat per la imatge, sense afectar gaire a la seva definició. Com en el cas que s'ha presentat, quan s'aplicava una compressió de tipus *low*, les formes tendien a suavitzar-se i els colors de fons a agrupar-se en menys tonalitats, per tal de reduir la quantitat d'informació i per conseqüència l'espai ocupat per la imatge.

¹¹CUDA

CUDA o *Compute Unified Device Architecture* fa referència a una API de programació i arquitectura, basada en C, que permet crear aplicacions que aprofitin la potència de càlcul de les targetes gràfiques sense haver d'aprendre un nou llenguatge de programació.

Si ens fixem en la taula de resultats 3, veurem com els resultats obtinguts pel model *SSIM*, són més semblants a la nostra percepció que el *PSNR*. Si comparem algunes dades, veiem que a la imatge *kodim10* 12, en compressió *hifc-hi*, el valor del **PSNR**, es troba al voltant dels 35 dB, el qual es troba més a prop de la zona "dolenta" o de mala reconstrucció que no pas de la bona reconstrucció. Però, si observem la imatge, i la comparem amb l'original, les diferències són difícilment apreciables i per la majoria de persones, la reconstrucció seria prou apte i detallada, i és justament aquest valor, el que ens representa el **SSIM**, el qual, en la mateixa comparació ens dona una puntuació de 0.917, molt a prop de l'1, que seria la igualtat perfecta.

Finalment, els darrers gràfics de l'apèndix 19, podem trobar representats en forma de diagrama de línies, la relació entre el *Compression Ratio* i el tipus de compressió aplicada, en els dos models, *HiFiC* i *HFC*, on podem veure com el model de compressió que utilitza *Tensorflow*, ofereix una representació més fidedigna segons el tipus de compressió aplicada, en canvi, el model amb *Pytorch*, obté uns resultats més dispersos, on, amb unes imatges s'obté una compressió molt alta, i en unes altres, més baixa, és a dir, no és tan regular com el model *HiFiC*, però tot i així, trobem uns resultats bastants semblants entre els dos models. Els altres dos gràfics que ens trobem, són referents a la comparació entre el *Compression Ratio* i el *PSNR* i *SSIM*, amb el model de compressió *HiFiC*. Si ens fixem, tant en el *PSNR* com el *SSIM*, trobem que no existeix gaire diferència entre un tipus de compressió *low* i un de tipus *high*, cosa que ens diu que els tres tipus de compressió fan realment bé el seu treball i que encara que el pes de la imatge disminueixi, no es perd gaire informació aparentment. De forma semblant a la relació entre el *CR* i el *SSIM*, podem fer la relació entre els *BPP* i el *SSIM*, on si ens fixem en la taula 1, veurem com tindrem uns resultats gairebé idèntics.

El problema, a l'hora de comprar mètriques torna a ser el que ja hem comentat. Encara que numèricament, sembla que les reconstruccions siguin molt semblants, si comparem les imatges visualment si que descobrirem algunes diferències més notables, com en els exemples que s'han esmentat anteriorment 14. Per això, i com també s'ha esmentat, encara que existeixen mecanismes de mesura, que ens poden donar una idea, cap d'ells, de moment, són tan precisos com ho són els éssers humans a l'hora de decidir si una reconstrucció és bona o dolenta i ni inclús els mateixos éssers humans ho són, ja que no deixa de ser una percepció el que una imatge estigui bé o mal reconstruïda.

7 CONCLUSIONS

El principal objectiu del treball, era aprofundir en les diferents tècniques de compressió que han estat creades, siguin basades en xarxes neuronals o no, però centrant-se més en les primeres. Oferint un estudi complet sobre els tipus, les seves característiques, diferències i com cadascuna d'elles s'ha enfocat cap a una direcció en concret, segons la seva aplicació, ja que són tantes, que és gairebé impossible que un mateix mètode pugui solucionar tots els problemes. A més, també destacar la implementació d'un mètode, en aquest cas el *HiFiC*, per tal de demostrar tot el que teòricament s'havia exposat al llarg del treball.

Personalment, aquest projecte ha estat un gran repte, ja que la meua experiència en l'àmbit era escassa i gràcies a tota la investigació i problemes que han anat sorgint, el coneixement ha augmentat fins al punt de veure un àmbit en el qual encara queda molt per desenvolupar i que en un futur no tan llunyà serà imprescindible a l'hora de treballar amb tota la quantitat de dades que es mouen arreu del món.

AGRAÏMENTS

Principalment, agrair al meu tutor, Joan Serra Sagrista, per la seva ajuda sobretot a la part inicial del treball, per canalitzar-lo cap al que finalment ha sortit i per tot el material que em va cedir per poder realitzar tot l'estudi de l'art i d'aquesta forma poder augmentar els meus coneixements en l'àmbit i sobretot, el meu interès.

REFERÈNCIES

- [1] Barracuda. What is data compression? . Visitat el dia 17-02-2022, Disponible en: [URL](#).
- [2] Debra A. Lelwer and Daniel S. Hirschberg. Data Compression, 1987. Visitat el dia 18-02-2022, Disponible en: [URL](#).
- [3] GeeksforGeeks. Difference between Lossy Compression and lossless Compression, 2020. Visitat el dia 28-02-2022, Disponible en: [URL](#).
- [4] Wikipedia. Warren Strugis McCulloch, 2022. Visitat el dia 23-03-2022, Disponible en: [URL](#).
- [5] Wikipedia. Wlater Pitts, 2022. Visitat el dia 23-03-2022, Disponible en: [URL](#).
- [6] Wikipedia. Neurona de McCulloch y Pitts, 2022. Visitat el dia 23-03-2022, Disponible en: [URL](#).
- [7] Wikipedia. Donald O. Hebb, 2022. Visitat el dia 23-03-2022, Disponible en: [URL](#).
- [8] Wikipedia. Hebbian Theory, 2022. Visitat el dia 23-03-2022, Disponible en: [URL](#).
- [9] Cornell University. Professor's perceptron paved the way for AI – 60 years too soon, 2022. Visitat el dia 23-03-2022, Disponible en: [URL](#).
- [10] Wikipedia. Paul Werbos, 2020. Visitat el dia 25-03-2022, Disponible en: [URL](#).
- [11] Le Cun, Y; Boser, B; Denker, J; Henderson, D; Howard, R; Hubbard, W.; Jackel, L. Backpropagation Applied to Handwritten Zip Code Recognition, 1989, *AT and T Bell Laboratories Holmdel, NJ, USA*. Visitat el dia 25-03-2022, Disponible en: [URL](#).
- [12] David Rumelhart, Geoffrey Hinton i Ronald Williams. Learning representations by back-propagating bugs, 1986, *Institute of Cognitive Science, San Diego*. Visitat el dia 25-03-2022, Disponible en: [URL](#).
- [13] Dan Lee. Reinforcement Learning, 2019. Visitat el dia 26-03-2022, Disponible en: [URL](#).

- [14] Jiang Guo. BackPropagation Through Time, 2013. Visitat el dia 27-03-2022, Disponible en: [URL](#).
- [15] Adri nas Gruslys, R mi Munos, Ivo Danihelka, Marc Lanctot and Alex Graves. Memory-Efficient Backpropagation Through Time, 2016. Visitat el dia 27-03-2022, Disponible en: [URL](#).
- [16] Yoshua Bengio. A Connectionist Approach to Speech Recognition, 1993, *University of Montreal*, Visitat el dia 27-03-2022, Disponible en: [URL](#).
- [17] Hochreiter, Sepp and Schmidhuber. Long Short-Term Memory, 1997, *Neuronal Comput.*. Visitat el dia 27-03-2022, Disponible en: [URL](#).
- [18] Geoffrey E. Hinton, Simon Osindero and Yee-Whye Teh. A fast learning algorithm for deep belief nets, 2006. Visitat el dia 27-03-2022, Disponible en: [URL](#).
- [19] Geoffrey E. Hinton. Training Products of Experts by Minimizing Contrastive Divergence, 2002, *Gatsby Computational Neuroscience Unit*. Visitat el dia 27-03-2022, Disponible en: [URL](#).
- [20] Yoshua Bengio. Greedy Layer-Wise Training of Deep Networks, 2007. *University of Montreal*. Visitat el dia 27-03-2022, Disponible en: [URL](#).
- [21] Arne Wolfewicz. Deep learning vs. machine learning – What’s the difference?, 2021. Visitat el dia 28-03-2022, Disponible en: [URL](#).
- [22] Kunal Rajan Deshmukh. Image Compression Using Neural Networks, 2019. Visitat el dia 29-03-2022, Disponible en: [URL](#).
- [23] Feng Jiang Wen Tao Shaohui Liu Jie Ren Xun Guo and Debin Zhao. An End-to-End Compression Framework Based on Convolutional Neural Networks, 2017. Visitat el dia 29-03-2022, Disponible en: [URL](#).
- [24] David J.Anderson and Andy Carmichel. Kanban Esencial Condensado, 2016. Visitat el dia 20-02-2022, Disponible en: [URL](#).
- [25] Trello, 2012. Visitat el dia 29-02-2022, Disponible en: [URL](#).
- [26] Wikipedia. Chroma Subsampling, 2022. Visitat el dia 15-03-2022, Disponible en: [URL](#).
- [27] Wikipedia. Fractal Compression, 2022. Visitat el dia 15-03-2022, Disponible en: [URL](#).
- [28] Wikipedia. Transform Coding, 2022. Visitat el dia 16-03-2022, Disponible en: [URL](#).
- [29] Wikipedia. Huffman Encoding, 2022. Visitat el dia 09-03-2022, Disponible en: [URL](#).
- [30] Wikipedia. Arithmetic Coding, 2022. Visitat el dia 10-03-2022, Disponible en: [URL](#).
- [31] Wikipedia. Run-length Encoding, 2022. Visitat el dia 09-03-2022, Disponible en: [URL](#).
- [32] Ziv, J. and Lempel, A. A universal algorithm for sequential data compression, 1997, *IEEE Transactions on Information Theory*. Visitat el dia 18-03-2022, Disponible en: [URL](#).
- [33] Ziv, J. and Lempel, A. Compression of individual sequences via variable-rate coding, 1998, *IEEE Transactions on Information Theory*. Visitat el dia 18-03-2022, Disponible en: [URL](#).
- [34] Terry A. Welch. A Technique for High-Performance Data Compression, 1984. Visitat el dia 19-03-2022, Disponible en: [URL](#).
- [35] Wikipedia. DEFLATE, 2022. Visitat el dia 19-03-2022, Disponible en: [URL](#).
- [36] an J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Nets, 2014. Visitat el dia 02-04-2022, Disponible en: [URL](#).
- [37] Patrick Kulp. Samsung’s AI Lab Can Create Fake Video Footage From a Single Headshot, 2019. Visitat el dia 02-04-2022, Disponible en: [URL](#).
- [38] James Vincent. Nvidia’s AI recreates Pac-Man from scratch just by watching it being played, 2020. Visitat el dia 02-04-2022, Disponible en: [URL](#).
- [39] Fabi n Mentzer, Jorge Toderici, Michael Tschanen, Eirikur Agustson. High-Fidelity Generative Image Compression, 2020. Visitat el dia 28-03-2022, Disponible en: [URL](#).
- [40] Mart n Abadi, Ashish Agarwal, Paul Barham. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems, 2015. Visitat el dia 28-04-2022, Disponible en: [URL](#).
- [41] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, Adam Lerer. Automatic differentiation in PyTorch, 2017. Visitat el dia 28-04-2022, Disponible en: [URL](#).
- [42] Quan Huynh-Thu and Mohammed Ghanbari. The accuracy of PSNR in predicting video quality for different video scenes and frame rates, 2012, *Telecommunication Systems*. Visitat el dia 10-05-2022, Disponible en: [URL](#).
- [43] Zhou Wang and Bovik, A.C. and Sheikh, H.R. and Simoncelli, E.P. Image quality assessment: from error visibility to structural similarity, 2014, *IEEE Transactions on Image Processing*. Visitat el dia 10-05-2022, Disponible en: [URL](#).

APÈNDIX

Tipus compressió	Fitxer	Espai comprimit	BPP	CR	PSNR	MSE	SSIM
Hific-lo	Girl	74 kb	0.1986	40.28	31.758 dB	43.381	0.841
Hific-lo	Kodim23	5kb	0.1194	67.00	34.454 dB	23.316	0.853
Hific-lo	Kodim10	7 kb	0.1545	51.77	33.427 dB	29.546	0.884
Hific-mi	Girl	136 kb	0.3644	21.95	32.831 dB	33.883	0.900
Hific-mi	Kodim23	10 kb	0.2210	36.20	35.863 dB	16.856	0.896
Hific-mi	Kodim10	14 kb	0.2979	26.85	34.664 dB	22.216	0.919
Hific-hi	Girl	202 kb	0.5419	14.76	31.758 dB	43.380	0.842
Hific-hi	Kodim23	15 kb	0.3068	26.08	36.679 dB	13.971	0.917
Hific-hi	Kodim10	21 kb	0.4285	18.67	35.617 dB	17.838	0.932
Hfc-py-lo	campament	25 kb	0.0719	111.27	33.07 dB	32.016	0.891
Hfc-py-lo	Kodim23	9 kb	0.1853	43.17	34.257 dB	24.402	0.883
Hfc-py-lo	Kodim10	10 kb	0.2159	37.05	31.200 dB	49.337	0.842
Hfc-py-mi	campament	54 kb	0.1570	50.96	34.01 dB	25.844	0.918
Hfc-py-mi	Kodim10	16 kb	0.3449	23.20	34.222 dB	24.595	0.888
Hfc-py-mi	Kodim23	14 kb	0.2868	27.89	35.315 dB	19.122	0.912
Hfc-py-hi	campament	80 kb	0.2298	34.81	33.64 dB	28.140	0.941
Hfc-py-hi	Kodim10	23 kb	0.4828	16.57	31.415 dB	46.944	0.910
Hfc-py-hi	Kodim23	18 kb	0.3736	21.41	35.950 dB	16.518	0.928

TAULA 1: TAULA GENERAL DE RESULTATS

PIXELS	ESPAI	FITXER
2048x1459 px	8.96 MB	girl_original.raw
512x768 px	1.17 MB	kodim23.raw
512x768 px	1.17 MB	kodim10.raw
768x512 px	1.17 MB	motos_original.raw
1536x2048 px	9.44 MB	edifici_original.raw
1512x2016 px	9.14 MB	façana_original.raw
512x768 px	1.17 MB	kodim9.raw
1000x1000 px	3 MB	mountain.raw
2048x1365 px	8.39 MB	campament.raw

TAULA 2: TAULA DE CARACTERÍSTIQUES DE LES IMATGES

IMATGE ORIGINAL	IMATGE COMPRIMIDA	PSNR (dB)	SSIM
kodim10.png	kodim10-hific-lo.png	33.43 dB	0.853
	kodim10-hific-hi.png	35.62 dB	0.917
	kodim10-hfc-lo-py.png	31.20 dB	0.882
	kodim10-hfc-hi-py.png	31.42 dB	0.909
edifici_original.png	edifici-hific-lo.png	32.88 dB	0.858
	edifici-hific-mi.png	33.70 dB	0.895
motos_original.png	motos-hific-lo.png	29.97 dB	0.770
	motos-hific-hi.png	31.90 dB	0.929
	motos-hfc-lo-py.png	29.93 dB	0.739
kodim23.png	kodim23-hific-lo.png	34.45 dB	0.853
	kodim23-hific-hi.png	36.68 dB	0.917
	kodim23-hfc-lo-py.png	34.26 dB	0.928
	kodim23-hfc-hi-py.png	35.95 dB	0.883

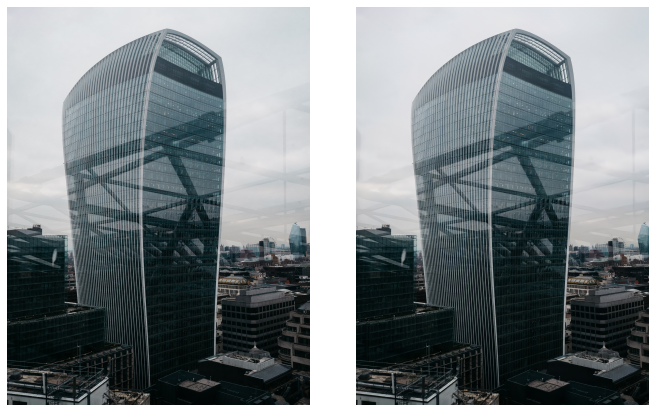
TAULA 3: TAULA DE RESULTATS DE LA MÈTRICA PSNR I SSIM



(a) imatge amb compressió LOW.

(b) imatge amb compressió HIGH.

Fig. 12: Comparació d'imatges comprimides en HiFiC



(a) imatge original.

(b) imatge amb compressió LOW.

Fig. 13: Comparació d'imatge original i comprimida



(a) imatge original.



(b) imatge amb compressió HIFIC low.



(c) Imatge amb compressió HiFiC Mid.



(d) Imatge amb compressió HiFiC High.

Fig. 14: Comparació d'imatges segons la compressió



(a) imatge comprimida amb HiFiC HI.



(b) imatge comprimida amb HiFiC Pytorch HI.

Fig. 15: Comparació d'imatges comprimides amb HiFiC HI i HiFiC-Pytorch HI



(a) imatge comprimida amb HiFiC LO.



(b) imatge comprimida amb HiFiC Pytorch LO.

Fig. 16: Comparació d'imatges comprimides amb HiFiC LO i HiFiC-Pytorch LO



(a) imatge comprimida amb HiFiC HI.



(b) imatge comprimida amb HiFiC Pytorch HI.

Fig. 17: Comparació d'imatges comprimides amb HiFiC HI i HiFiC-Pytorch HI



(a) imatge original.



(b) imatge comprimida amb HiFiC Pytorch LO.



(c) imatge comprimida amb HiFiC Pytorch MI.



(d) imatge comprimida amb HiFiC Pytorch HI.

Fig. 18: Comparació d'imatges comprimides amb HiFiC-Pytorch

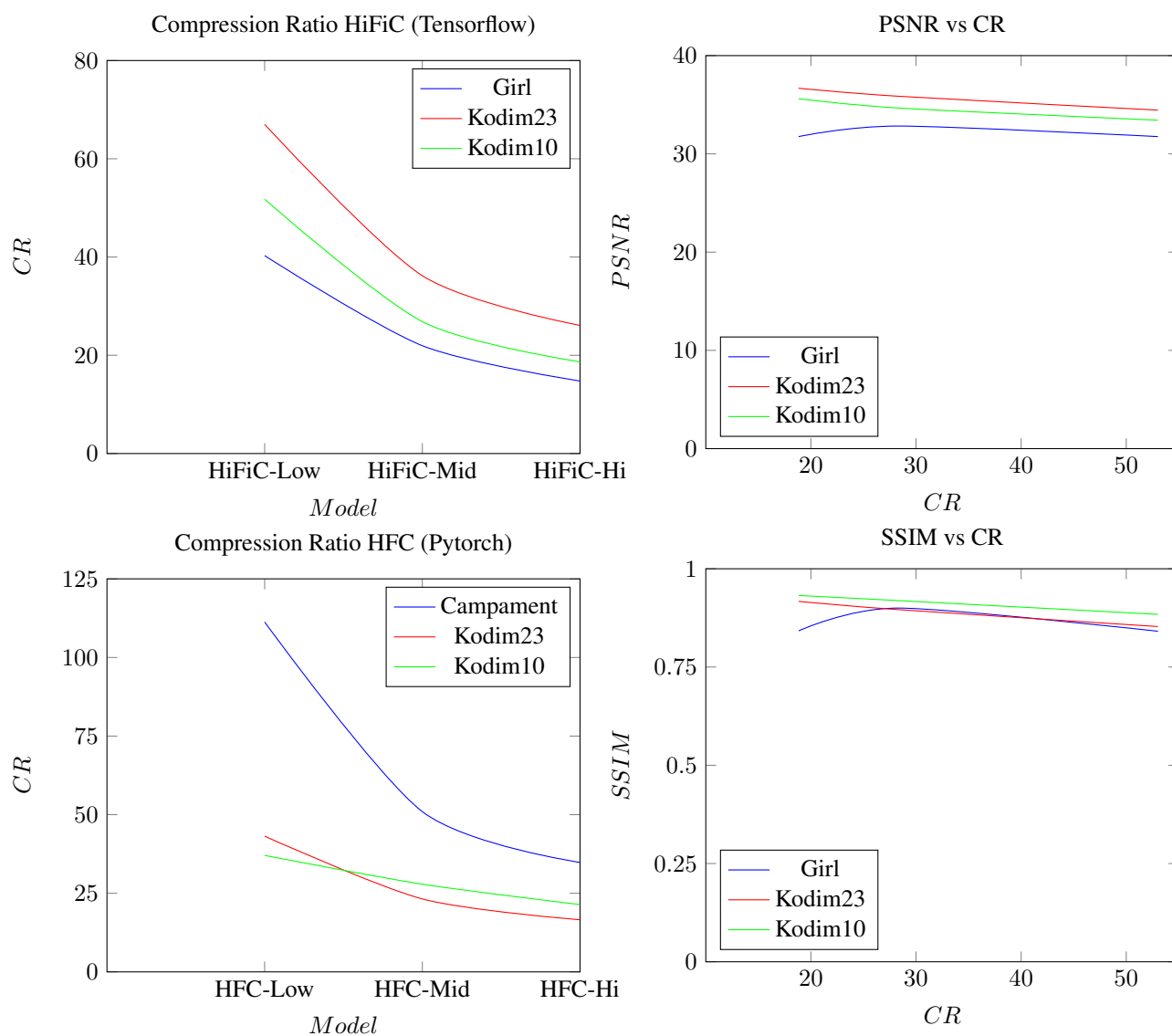


Fig. 19: Gràfics representatius