Compressió de punts de control d'entrenament d'IA

Marc Estapé Ayén

25 de juny de 2025

Resum– Aquest projecte presenta el disseny i la implementació d'un compressor adaptatiu sense pèrdua, orientat a la compressió eficient de fitxers de punts de control generats durant el procés d'entrenament de models d'intel·ligència artificial. Aquests fitxers binaris estan formats principalment per tensors numèrics de diferents tipus com poden ser `float32` o `int64` i poden assolir una mida de diversos gigabytes, fet que representa una limitació tant per a l'emmagatzematge com la distribució. El sistema proposat analitza el contingut intern de cada fitxer i selecciona dinàmicament el compressor òptim per a cada tipus de dada. A més, s'han aplicat millores a l'estructura dels headers per reduir la seva sobrecàrrega i millorar l'adaptabilitat amb el sistema. Els resultats obtinguts mitjançant un banc de proves mostren una taxa mitjana de compressió de 1.0961, superior a la dels mètodes actuals com `7-Zip`.

Paraules clau– Compressió sense pèrdua, punts de control, intel·ligència artificial, tensors numèrics, optimització de fitxers, compressió adaptativa.

Abstract– This project presents the design and implementation of a lossless adaptive compressor aimed at the efficient compression of checkpoint files generated during the training process of artificial intelligence models. These binary files are primarily composed of numerical tensors of various types, such as `float32` or `int64`, and can reach sizes of several gigabytes, representing a clear limitation for both storage and distribution. The proposed system analyzes the internal content of each file and dynamically selects the most suitable compressor for each data type. In addition, structural optimizations have been applied to the headers to reduce their overhead and improve the system's modularity and adaptability. The results obtained from a benchmark evaluation show an average compression ratio of 1.0961, which outperform existing general-purpose methods such as `7-Zip`.

Keywords– Loseless compresssion, checkpoints, artificial intelligence, numerical tensors, file optimization, adaptative compression.

1 Introducció - Context del treball

En l'àmbit de la intel·ligència artificial, els punts de control són arxius binaris que emmagatzemen l'estat intern d'un model entrenat, incloent-hi pesos i paràmetres. Aquests fitxers són essencials per a la recuperació i continuació del procés d'entrenament, la distribució de models i la seva inferència en altres sistemes. Amb l'augment de la mida dels models d'IA, especialment en àrees com el processament del llenguatge natural i la visió per computador, aquests arxius han crescut significativament arribant a ocupar diversos gigabytes. En la majoria de casos, aquests arxius contenen grans quantitats de tensors numérics de diferents tipus com poden ser float32 o int64.

La gestió eficient d'aquests fitxers és un repte important. La seva mida suposa un problema tant per a l'emmagatzematge com per a la seva transmissió en xarxa, afectant el desplegament de models i el seu ús en dispositius amb recursos limitats. Una compressió eficient d'aquests fitxers podria suposar un gran estalvi en temps i espai permetent una distribució més ràpida i una millor optimització dels recursos computacionals. Els compressors actuals com

• E-mail de contacte: marc.estape03gmail.com
• Mención realitzada: Tecnologies de la Informació
• Treball tutoritzat per: Joan Bartrina Rapesta — Ciències de la Computació i Intel·ligència Artificial
• Curs 2024/25

ZIP, GZIP o LZMA no estan optimitzats per a la naturalessa específica d'aquests arxius, ja que apliquen algorismes genèrics sense considerar l'estructura interna de les dades.

La motivació d'aquest projecte és desenvolupar un compressor i descompressor específic per a aquests arxius, buscant superar els mètodes tradicionals en termes de taxa de compressió sense pèrdua. La proposta es basa en una estratègia adaptativa: analitzar la composició dels fitxers de punts de control, identificar els diferents tipus de dades que contenen (com floats i integers) i aplicar el millor compressor disponible per a cada tipus de dada. Aquesta segmentació permetrà optimitzar la compressió en funció de les característiques específiques de cada conjunt de dades, obtenint un rendiment superior al d'un enfocament estàndard.

Per poder treballar aquest projecte, s'haurà de dur a terme una recerca exhaustiva en diverses fonts d'informació. En primer lloc, s'estudiarà en profunditat l'estructura interna d'aquests arxius per entendre què hem de comprimir. Seguidament es consultaran articles científics especialitzats en loseless compression amb l'objectiu d'identificar quins mètodes són els més eficients per a diferents tipus de dades. A més, s'analitzarà la documentació de diferents llibreries de compressió com podrien ser zlib, Zstandard o LZ4. Finalment s'estudiarà quina és la millor opció tècnica per desenvolupar el compressor.

2 Estat de l'art

En el context de la Intel·ligència artificial moderna, els *checkpoints* són fitxers binaris que permeten desar l'estat intern d'un model d'aprenentatge automàtic, incloenthi els pesos, els biaixos i altres paràmetres d'entrenament. Aquests arxius són essencials per a la continuació del procés d'entrenament, la reutilització del model en altres sistemes o la seva compartició en entorns de producció o investigació. En models d'ultima generació com els grans models LLM o xarxes neurals convulacionals per a visió per computador aquests fitxers poden assolir desenes de gigabytes de mida.

Un aspecte clau d'aquests *checkpoints* és la seva estructura interna. La major part del contingut són tensors, majoritariament representants com a arrays multidimensionals de valors numérics. Aquesta regularitat que no acostuma a trobar-se en fitxers de text o imatges permet aplicar estratègies de compressió especialitzades.

Els compressors genèrics actuals, com podrien ser GZIP, Brotli o Zstandard, utilitzen algorismes basats en tècniques com LZ77, codificació per Huffman o estadística. Encara que són molt eficients en formats generals, no estan optimitzats per a una estructura repetitiva i uniforme com aquest cas.

Diversos treballs han intentat abordar aquest problema amb enfocaments específics. Un dels més destacats és **LC-Checkpoint** [1], que explora tècniques específiques per comprimir checkpoints durant el procés d'entrenament. Aquest treball s'enmarca dins del mateix àmbit problemàtic que aquest treball: millorar la compressió dels fitxers per facilitar la seva manipulació i la reutilització.

Un altre enfocament similar al treball anterior el trobem a **Inshrinkerator** [2] que presenta una eina pensada per comprimir checkpoints de manera incremental durant l'entrenament, permetent recuperar qualsevol estat anterior. Encara que el seu enfocament es centra més en l'eficiència temporal i operativa a temps real, reforça la idea de la necessitat d'eines específiques per a gestionar la mida d'aquests fitxers.

Aquests treballs evidencien la manca de solucions generalistes i eficients per a la compressió de fitxers de punts de control. Aquest fet justifica el plantejament d'aquest treball: desenvolupar un compressor adaptatiu que a partir de l'anàlisi del contingut intern dels tensors seleccioni la millor estratègia de compressió per a cada segment de dades que millori la taxa de compressió a comparació dels actuals.

3 Objectius

L'objectiu principal del projecte és desenvolupar un compressor i descompressor funcional, específic per a fitxers de punts de control de models d'IA, que sigui capaç d'assolir una taxa de compressió sense pèrdua superior als compressors actuals.

Per assolir aquest objectiu es plantegen els següents subobjectius:

- Analitzar els diferents tipus de dades continguts en els fitxers de punts de control.
- Desenvolupar un banc de proves per avaluar quins compressors existents són més eficients per a cada tipus de dada.
- Implementar una primera versió del compressor basada en els resultats del banc de proves.
- Refinar el compressor mitjançant tècniques d'optimització, com la reducció de la mida de cada header.
- Explorar tècniques avançades com l'ús de lookup tables per aconseguir una taxa de compressió encara més elevada.
- Desenvolupar un descompressor per tal de reconstruir els arxius sense cap pèrdua d'informació.

3.1 Objectiu opcional

Durant el transcurs del projecte es va incorporar un objectiu addicional de caràcter opcional: la possible participació en un concurs internacional de compressió de dades.

Aquest concurs inclou una categoria dedicada a la compressió de punts de control d'IA, àmbit que s'alinea de manera directa amb l'enfocament i objectius del projecte. Tot i no formar part de la planificació acadèmica inicial, aquesta fita es va plantejar com un repte personal i una oportunitat per validar el rendiment del compressor en un context real i competitiu.

Aquesta iniciativa ha aportat un valor afegit reforçant la motivació, exigència tècnica i projecció pràctica del treball desenvolupat.

4 Metodologia

El desenvolupament d'aquest projecte s'ha basat en una metodologia iterativa ja que permet una millora contínua i progressiva del compressor a mesura que s'obtenen resultats. Aquesta aproximació facilita la identificació d'ineficiències

i l'adopció d'optimitzacions de manera escalonada, garantint que cada fase compleixi els objectius plantejats abans de passar a la següent.

4.1 Fases de desenvolupament

El procés es dividirà en diferents fases, cadascuna amb objectius específics per garantir un pla estructurat i eficient.

4.1.1 Anàlisi i estudi de l'estat de l'art

En primer lloc es realitzarà un anàlisi exhaustiu de les tècniques de compressió sense pèrdua i dels compressors més utilitzats actualment com podrien ser ZIP o GZIP. Aquest estudi permetrà fer una valoració d'aquests mètodes respecte la compressió dels diferents tipus de dades.

A continuació, s'identificaran els diferents tipus de dades presents en els arxius de punts de control com floats, integers o altres formats binaris. Aquesta classificació serà fonamental per aplicar estratègies de compressió específiques per a cada tipus de dada.

4.1.2 Desenvolupament d'un banc de proves

Un cop establerts els conceptes teòrics i sabent els tipus de dades a comprimir, es construirà un banc de proves per analitzar el rendiment dels diferents compressors disponibles.

A partir d'uns fitxers reals de punts de control, es realitzaran una sèrie de proves per comparar el rendiment de diferents compressors en funció del tipus de dada. Aquest procés permetrà identificar quins compressors són més eficients per a cada cas i servirà com a base per a la implementació del compressor final.

4.1.3 Implementació de la primera versió del compressor

Amb els resultats del banc de proves, es desenvoluparà una primera versió funcional del compressor i descompressor. Aquest esquema de compressió combinarà els millors compressors selectivament segons el tipus de dada a comprimir.

La implementació es farà mitjançant Python, utilitzant llibreries de compressió com zlib segons els resultats obtinguts en les proves. Aquesta versió inicial tindrà com a objectiu aconseguir una compressió efectiva encara que inicialment no estigui totalment optimitzada.

4.1.4 Fase de refinament

Un cop desenvolupada la primera versió es treballarà en la seva optimització per reduir encara més la mida dels fitxers comprimits. Aquesta fase inclourà millores en l'estructura interna del fitxer comprimit, com la reducció de la mida dels headers mitjançant tècniques com la compactació de metadades o l'agrupació de variables.

Es desenvoluparà una segona versió del compressor amb millores en l'eficiència, corregint possibles ineficiències l i ajustant els paràmetres dels compressors seleccionats per maximitzar el rendiment.

4.1.5 Exploració d'estratègies avançades

Si després de la fase de refinament encara es considera que la taxa de compressió pot millorar-se es plantejarà la incorporació de tècniques més avançades. Una d'aquestes estratègies podria ser l'ús de lookup tables per reduir la redundància de certs valors dins dels punts de control.

Aquesta fase serà opcional i es desenvoluparà en funció dels resultats obtinguts en les versions anteriors del compressor.

4.1.6 Avaluació i comparació de resultats

Finalment es realitzarà un anàlisi comparatiu del compressor desenvolupat respecte als compressors tradicionals. Es mesurarà la taxa de compressió obtinguda, el temps de compressió i descompressió i l'impacte de les optimitzacions aplicades. Aquests resultats serviran per validar l'eficàcia del mètode proposat i per determinar si compleix l'objectiu de millorar la compressió dels punts de control d'IA en comparació amb els mètodes actuals.

4.2 Planificació del desenvolupament

Per garantir un desenvolupament estructurat, eficient i alineat amb els objectius plantejats, el projecte s'ha dividit en diferents fases amb una durada estimada. Aquesta planificació ha reforçat la metodologia iterativa adoptada ja que ha proporcionat un marc temporal clar que ha permés aplicar millores continuades, revisar els resultats parcials i reestructurar les propietats en funció de l'estat i el progrés obtingut.

Tot i que la planificació inicial va ser elaborada abans de començar el desenvolupament, aquesta s'ha adaptat minimament durant el desenvolupament del projecte per tal d'optimitzar el valor del resultat final.

La taula següent mostra un resum de les fases principals del projecte, la seva durada estimada i les tasques a desenvolupar.

5 IMPLEMENTACIÓ

5.1 Format dels fitxers binaris d'entrada

Els fitxers d'entrada gestionats pel compressor són fitxers binaris estructurats que contenen múltiples blocs consecutius. Cada bloc representa una matriu multidimensional i està format per dos components: una capçalera descriptiva i les dades associades a la matriu.

La capçalera té com objectiu definir les caracterísitques de la matriu i conté els següents camps, codificats en format little-endian:

- **dim_num (4 bytes)**: un valor enter que especifica el nombre de dimensions de la matriu.
- **dim_info (4 bytes per dimensió)**: per a cada dimensió, un enter que indica la mida corresponent.
- **data_type (4 bytes)**: un enter que identifica el tipus de dada emmagatzemada a la matriu (0: uint, 1: int, 2:float).
- **data_type_len (4 bytes)**: un enter que determina la grandària en bits de cada element de la matriu (8, 16, 32 o 64 bits).

TAULA 1: PLANIFICACIÓ DE LES FASES DEL PROJECTE

Fase	**Durada** *(setmanes)*	**Tasques a realitzar**
Anàlisi i estudi de l'estat de l'art	1	Revisió de compressors i estructura dels arxius
Desenvolupament del banc de proves	2	Creació i execució de proves amb compressors
Implementació de la primera versió del compressor	2	Programació del compressor inicial en Python
Fase de refinament i optimització	4 – 5	Millora del format intern i compressió
Exploració d'estratègies avançades	6 – 7	Aplicació de tècniques com lookup tables
Avaluació i documentació final	3	Comparació de resultats i redacció de la memòria

Seguit de la capçalera, es troben les dades de la matriu codificades en binari, amb una mida total calculada com el producte de totes les dimensions multiplicat pel nombre de bits per elements (ajustat a bytes).

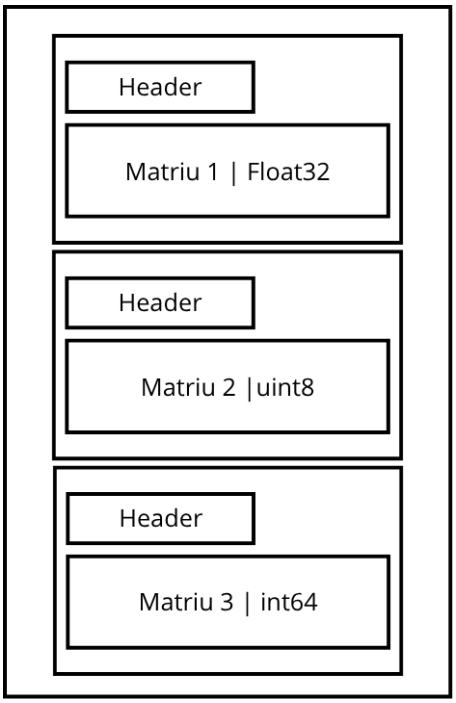

Fig. 1: Exemple d'estructura interna del fitxer binari

5.2 Estratègia de compressió

Durant el procés de compressió, cada bloc del fitxer input segueix els passos següents:

5.2.1 Lectura i optimització del header

Per cada matriu, es llegeix el header original i s'apliquen tècniques i estratègies d'optimització del header.

5.2.2 Compressió selectiva de la matriu

Un cop llegides les dades, el sistema determina automàticament el compressor òptim (en aquest cas entre `zlib` o `brotli`) en funció del tipus de la matriu.

5.2.3 Comprovació de mida final

En aquest punt, es compara la mida de la matriu original amb la mida de la matriu comprimida. Si la compressió no aporta beneficis, es guarda la matriu original. Gràcies a aquesta implementació, podem aplicar un nivell de compressió més elevat sense penalitzar els casos on les matrius son petites i la sobrecàrrega del header del compressor pot augmentar la mida final.

5.2.4 Escriptura ordenada al fitxer comprimit

S'escriu de manera sequencial la mida del header optimitzat, el header, la mida de la matriu i les seves dades, que poden estar comprimides o no.

L'output d'aquest procés és un fitxer anomenat `checkpoint_compressed.mxc` (*Mixed Compressors*), que pot contenir blocs comprimits amb tècniques diferents segons el tipus de dades.

Pel que fa al procés de descompressió, aquest segueix exactament el procés invers. El decompressor llegeix el fitxer `.mxc`, reconstrueix el header original, aplica el mètode de descompressió corresponent i escriu les dades a un nou fitxer binari `checkpoint_reconstructed.bin`.

Com que aquest compressor es sense pèrdua, la validesa de tot el procés es pot verificar fàcilment comprovant que el hash del fitxer original i el fitxer recuperat sigui el mateix.

5.3 Optimització del header

Tal com s'ha esmentat a l'apartat anterior, una de les primeres fases del procés de compressió és l'anàlisi i reestructuració del header de cada matriu.

El format del header és força explícit pero alhora bastant redundant, això és degut a que no aprofita l'espai utilitzat afegint bits innecessaris que acaben augmentat la mida final del fitxer.

Aquest header té una mida variable de 16 a 32 bytes, segons el nombre de dimensions.

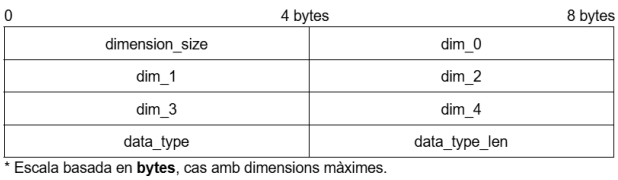

Fig. 2: Estructura del header original

Aquesta estructura, tot i ser funcional, representa un cost notable quan es treballa amb centenars de matrius petites. Per tal de millorar l'eficiència, s'ha implementat una optimització que redueix al màxim l'espai ocupat, mantenint tota la informació necessària.

Les optimitzacions aplicades són les següents:

- `dimension_size` s'ha codificat en un sol byte, en lloc dels 4 bytes originals ja que el valor màxim de dimensions possibles és 5.
- El nombre de bytes reservats per a cada dimensió es manté igual per garantir compatibilitat amb dimensions grans.
- Els camps `data_type` i `data_type_len` s'han empaquetat junts en un únic byte, utilitzant 4 bits per cada valor. Això és degut a que, tant els diferents tipus de dades (3) com els diferents tipus de mida de dada (4) es poden representar cadascun amb 4 bits. En el cas de la mida de la dada seria la següent: 8: 0b00, 16: 0b01, 32: 0b10, 64: 0b11.
- S'ha afegit un nou identificador que indica el tipus de compressió de la matriu. Aquest identificador fa servir 3 bits del primer byte (compartit amb `dimension_size`) i pot representar fins a 6 compressors diferents, incloent l'opció de cap compressor.

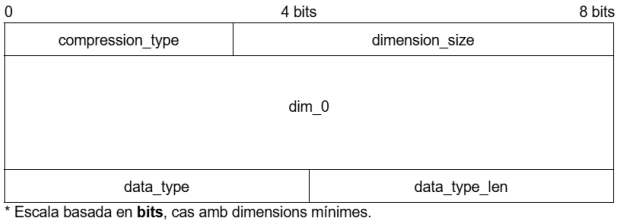

Fig. 3: Estructura del header optimitzat

Aquest conjunt de millores suposa una reducció significativa de l'espai ocupat pel header, especialment rellevant en fitxers amb moltes matrius de poques dimensions, passant de 16/32 bytes a 6/22. A més, aquestes optimitzacions s'han dissenyat expressament per adaptar-se a l'estructura concreta dels fitxers analitzats, tot i que cal tenir present que podrien no ser aplicables universalment si es treballa amb formats lleugerament diferents.

5.4 Optimitzacions addicionals

Amb l'objectiu de maximitzar la taxa de compressió final, s'han avaluat diverses tècniques addicionals que habitualment ofereixen bons resultats en entorns amb dades redundants o altament correlacionades.

5.4.1 Lookup Tables

La codificació de taules de consulta o Lookup Tables és útil en conjunts de dades amb un nombre reduït de valors únics ja que permet substituir valors repetits per índex compactes. En els fitxers tractats, la presència d'elements amb alta variabilitat no ha fet possible la seva implementació.

5.4.2 Codificació diferencial DPCM

Aquesta tècnica consisteix en emmagatzemar la diferència entre valors consecutius en lloc dels valors originals i resulta especialment efectiva en dades amb una correlació seqüencial forta, pròpia de les imatges. En el nostre cas, els tensors no presentaven aquesta estructura i per tant els valors resultants no milloraven la compressió.

5.4.3 Codificació amb Huffman

Es va plantejar aplicar Huffman com a preprocessament abans dels compressors convencionals, però aquest pas afegia complexitat computacional addicional i en la pràctica no generava guanys rellevants en la mida final. A més, les tècniques modernes de compressió ja integren variants d'aquesta codificació.

Les proves realitzades van demostrar que aquestes optimitzacions, malgrat ser efectives en altres contextos, no oferien una millora significativa en els fitxers utilitzats en aquest projecte. En molts casos, la mida final del fitxer es mantenia inalterada o fins i tot augmentava lleugerament degut al sobrecost de la codificació.

Per aquest motiu, s'ha decidit no incloure aquestes tècniques en la versió final del compressor, prioritzant una aarquitectura més simple, eficient i fàcilment mantenible.

5.5 Entorn de desenvolupament i control de versions

El desenvolupament d'aquest projecte s'ha realitzat utilitzant el llenguatge de programació Python, escollit per la seva versatilitat i la disponibilitat de biblioteques eficients per a la manipulació de dades binaries i la compressió com és Numpy [3].

Per garantir la traçabilitat i el control de canvis durant tot el procés, s'ha utilitzat Git com a sistema de control de versions. Aquesta eina ha permès mantenir un registre detallat de les modificacions i ha facilitat la portabilitat del projecte en diversos dispositius.

L'estructura del projecte s'ha organitzat modularment, separant amb funcions les diferents funcionalitats. A més, el repositori també disposa dels fitxers amb els quals s'han fet totes les proves d'aquest projecte. Gràcies a aquesta organització, s'ha afavorit la manteniblitat i escalabilitat del codi.

5.6 Arquitectura final del compressor

Per complementar la descripció de l'entorn de desenvolupament i organització modular del projecte, la Figura 4 mostra l'arquitectura final del compressor implementat, amb els principals components i flux de dades entre ells.

6 RESULTATS

6.1 Resultats del banc de proves

Les proves s'han realitzat sobre quatre arxius binaris gairebé idèntics, cadascun amb 792 matrius (740 `float32` i 52 `int64`). Les mètriques recollides corresponen exclusivament a les matrius que coincideixen amb el tipus de dada analitzat, assegurant així una avaluació precisa i rellevant per a cada cas.

6.1.1 Float32

A continuació es presenta una taula resum amb els valors mitjans obtinguts pels diferents compressors aplicats sobre

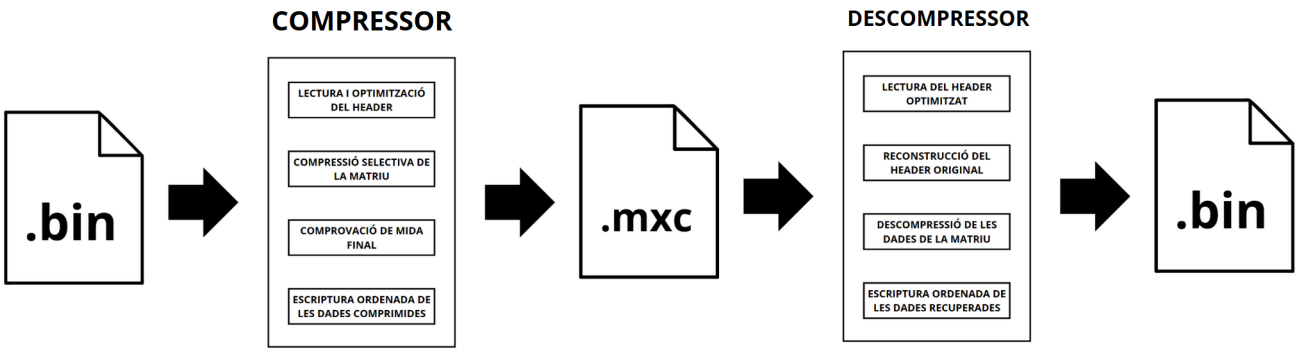

Fig. 4: Arquitectura final del compressor

les dades de tipus `float32`:

TAULA 2: MITJANES DEL BANC DE PROVES PER DADES `FLOAT32`

Compressor	**Taxa** *Compressió*	**Temps** *mitjà (s)*	**Temps total** *(s)*
Brotli	1.0705	0.0998	73.12
Bzip2	0.8844	0.1562	115.96
Gzip	1.0250	0.0722	53.28
LZ4	0.9597	0.0374	27.51
LZMA	0.9734	0.3772	279.47
Zopfli	1.0328	3.0967	2291.20
Zstd	1.0408	0.0055	4.04

Per complementar la informació de la taula anterior, s'ha generat una gràfica de dispersió de punts on es representa la relació entre la taxa mitjana de compressió i el temps total de compressió per a cada compressor. Aquesta visualització permet analitzar de forma més intuïtiva el compromís existent entre eficiència i velocitat, destacant quins compressors ofereixen un millor balanç global.

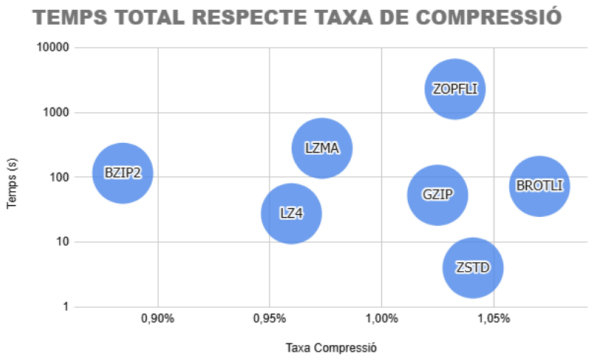

Fig. 5: Gràfic de Dispersió entre temps i taxa de compressió

6.1.2 Int64

En aquest cas, el rendiment obtingut pels compressors ha estat molt inferior al de les dades del tipus `float32`. Això és degut principalment al fet que la gran majoria de matrius presents als arxius analitzats no són d'aquest tipus, motiu pel qual la selecció i optimització dels compressors s'ha enfocat prioritàriament en l'altre tipus de dada.

A més, analitzant en detall les matrius de tipus `int64` , s'ha observat que pràcticament totes tenen una estructura mínima, corresponent a matrius amb un únic element. Aquestes matrius no generen gaire oportunitat per a compressions sofisticades, ja que acabarien augmentant la mida total del fitxer, per tant s'ha optat per una solució senzilla i eficient: comprimir-les amb el compressor `zlib`.

Aquest compressor, gràcies al seu funcionament basat en diccionaris i la seva baixa sobrecàrrega, resulta especialment adequat per a dades petites i repetitives, com és el cas d'aquestes matrius. Aquesta elecció permet mantenir una mida comprimida ajustada, sense afegir complexitat ni dades innecessàries i garanteix una compressió efectiva dins les limitacions del conjunt de dades.

6.2 Resultats finals del compressor propi

Un cop completada la implementació del compressor, s'ha procedit a provar el seu rendiment aplicant-lo sobre els quatre arxius del banc de proves.

Els resultats obtinguts sobre la taxa de compressió han estat els següents:

- **BIN1:** 1.0943%
- **BIN2:** 1.0950%
- **BIN3:** 1.0946%
- **BIN4:** 1.1008%

La taxa mitjana final obtinguda és de **1.0961**, una xifra lleugerament superior a la d'altres compressors tradicionals com 7Zip.

6.3 Discussió dels resultats

Els resultats mostren que el disseny del compressor propi ha estat adequat per al cas d'estudi plantejat. La seva arquitectura modular i l'estratègia de compressió adaptativa han demostrat ser eficients per a conjunts dominats per matrius de tipus `float32`, on s'ha obtingut una compressió superior al 1,09%.

Els resultats també han permés determinar quina combinació de compressors resulta més adequada per implementar la versió final del sistema. A partir dels resultats obtinguts en el banc de proves, s'ha pogut constatar que, si bé el compressor LZMA és el que genera una mida final

d'arxiu menor, la seva taxa mitjana de compressió és inferior als altres. Aquest comportament és degut a que, aquest compressor pot estar comprimint de manera molt eficient les matrius més grans, pero en el cas de les petites, acaba penalitzant la seva taxa mitjana.

Com que l'objectiu és dissenyar un compressor reutilitzable i aplicable a diferents conjunts de dades, cal prioritzar la taxa mitjana de compressió ja que representa millor el rendiment general del compressor.

Inicialment es va optar per utilitzar el compressor Zstd com a opció principal per a les dades tipus `float32` degut a la seva alta velocitat i bons resultats en la compressió.

Durant el desenvolupament, quan es va ampliar el conjunt de proves i es va descartar la integració de tècniques avançades com DPCM o Huffman, es van provar nous compressors. Un d'ells, Brotli, va aconseguir una millor taxa mitjana i final de compressió amb el mateix tipus de dades. Per aquest motiu, Zstd va ser substituït a la versió final del compressor.

A més, Brotli es pot configurar de manera que, a costa d'un increment significatiu del temps de compressió, millori la seva eficiència i redueix la mida de l'arxiu comprimit. Si apliquem un alt nivell de compressió, podem arribar a obtenir resultats millors que LZMA.

També es va explorar la possibilitat d'implementar un sistema de thresholds basat en la mida de les matrius, de manera que només es poguèssin comprimir aquelles matrius que superessin un cert valor. Aquesta estratègia buscava evitar la compressió ineficient de matrius molt petites. Malauradament es va observar que escollir un llindar òptim era altament arbitrari i no generalitzable, ja que el comportament de les matrius variava significativament entre diferents conjunts de dades i tipus. Aquesta manca de robustessa va portar a descartar aquesta aproximació.

Per a les dades de tipus int64 es farà servir el compresor zlib, ja que s'ha observat que aquestes matrius tenen dimensións mínimes i poca capaitat de compressió. Aquest compressor, amb una sobrecàrrega molt baixa, ofereix un bon equilibri entre simplicitat i eficiència per a aquest tipus concret de dades.

Es pot considerar que els objectius principals del projecte han estat assolits: s'ha dissenyat i implementat un sistema funcional, s'han obtingut millores respecte a compressors generals com gzip i s'ha validat el correcte funcionament tant del procés de compressió com de descompressió. També s'ha maximitzat l'eficiència mitjançant l'optimització de l'estructura del fitxer i l'ús selectiu de compressors especialitzats.

Tot i això, no s'ha pogut assolir l'objectiu opcional del repte competitiu, que exigia una taxa mínima de compressió de 1.15. Aquesta diferència s'explica principalment per les limitacions dels compressors tradicionals quan es treballa amb dades que ja tenen una entropia relativament baixa i una estructura senzilla.

6.4 Anàlisi crítica

Tot i l'assoliment global dels objectius, és important identificar els punts febles i les limitacions del projecte. En primer lloc, la diversitat del conjunt de dades utilitzat ha estat limitada ja que gairebé totes les matrius són de tipus `float32` i tenen dimensions reduïdes. Aquesta falta d'heterogeneïtat ha restringit l'avaluació del comportament del compressor en escenaris més diversos, com ara conjunts amb dades de tipus `int32`, `uint8` o `float64`, o fins i tot amb matrius de dimensions més elevades.

També s'ha optat per una estratègia basada exclusivament en compressors preexistents. Tot i que aquesta elecció ha permès un desenvolupament àgil i pràctic, ha limitat el marge de millora, especialment amb l'adaptabilitat del sistema a patrons específics del contingut. No s'han explorat tècniques més innovadores com l'anàlisi estadística avançada o l'aprenentatge automàtic, que podrien aportar millores en entorns amb una entropia més baixa o estructures repetitives.

A més, durant el desenvolupament, es van explorar diverses estratègies que finalment van ser descartades per falta de robustessa. Per exemple, es va estudiar la possibilitat d'aplicar thresholds depenent de la mida per decidir si una matriu s'havia de comprimir o no. Tot i que aquesta aproximació podia millorar lleugerament la mitjana global en alguns casos, la seva aplicació era massa arbitrària i depenia del tipus i mida de les matrius. Aquests resultats van fer que la solució no fos generalitzable i es descartés per prioritzar una estratègia més sistemàtica.

Finalment, l'optimització del *header*, tot i ser efectiva en la reducció d'espai, introdueix una complexitat addicional al procés de lectura i reconstrucció. Aquesta aproximació pot dificultar la seva extensió a formats més estàndard o la seva integració en entorns externs. Aquest fet denota la necessitat d'una futura revisió del disseny amb criteris d'estandardització i interoperabilitat.

De cara a una possible generalització del compressor a escenaris més amplis, seria fonamental ampliar el conjunt de proves amb datasets més diversos i realistes. Aquesta ampliació permetria validar el comportament del sistema amb més varietat de tipus de dades i escales, i afinar millor quins compressors o estratègies són òptimes per a cada cas. A més, obriria la porta a redissenyar l'arquitectura amb més flexibilitat o modularitat, per tal d'adaptar-la a casos d'ús específics dins d'un entorn productiu.

7 CONCLUSIONS I LÍNIES FUTURES

Un cop finalitzada la fase principal de desenvolupament del projecte, es pot afirmar que s'han assolit tots els objectius principals plantejats en la proposta inicial. El compressor funcional s'ha desenvolupat amb èxit, implementant una estratègia adaptativa capaç d'aplicar compressors diferents en funció del tipus de dada i obtenint resultats competitius, especialment en el cas de les matrius de tipus float32, que representaven la gran majoria dels conjunts analitzats.

Tot i que la fase d'exploració de tècniques avançades no ha donat els resultats esperats, ja que cap de les aproximacions plantejades han aportat un guany significatiu, aquesta fase ha estat clau per confirmar els límits pràctics d'algunes d'aquestes estratègies quan es treballa amb dades de naturalesa específica.

D'altra banda, l'objectiu opcional introduït posteriorment, que consistia en participar en una iniciativa competitiva que exigia una taxa de compressió mínima de 1.15 no s'ha pogut assolir. Malgrat els esforços per optimitzar el rendiment, els resultats obtinguts no assolien aquest llindar.

Els resultats globals del projecte han permès constatar un aspecte important: els compressors tradicionals i actuals com pot ser 7-zip ja estan molt optimitzats. Les tècniques actuals o heurístiques senzilles poden oferir certes millores puntuals, però no suposen un avenç significatiu ni tenen un gran impacte en el rendiment global.

A més, tot i que l'estratègia proposada és teòricament prometedora, no s'ha pogut explotar tot el seu potencial. El principal motiu ha estat la limitació del conjunt de dades disponibles i el seu contingut homogeni, format pràcticament per matrius de tipus float32. Aquesta limitació i falta de diversitat a les dades no ha permés explotar al màxim el propòsit de la proposta.

Si es disposés de més temps o si el projecte es volgués continuar en un futur, el plantejament es reorientaria cap a un enfocament diferent. En lloc de centrar-se en tècniques i compressors tradicionals, s'exploraria el potencial de les xarxes neurals i els models d'intel·ligència artificial aplicats a la compressió. A l'actualitat, aquestes aproximacions han començat a demostrar capacitats superiors als compressors convencionals, i fins i tot podrien acabar establint un nou estàndard per a la compressió de fitxers binaris complexos.

AGRAÏMENTS

Per concloure aquest treball, voldria expressar el meu sincer agraïment al meu tutor Joan Bartrina Rapesta per la seva dedicació, constància i orientació durant tot el procés. Els seus comentaris tècnics i suggeriments han estat claus per donar forma al projecte i encaminar-lo cap a una proposta sòlida.

També vull agrair a la meva família i, en especial, a la meva parella pel seu suport constant durant aquests mesos. Tot i no estar vinculats directament al camp de la informàtica, la seva visió externa ha estat molt valuosa per analitzar el projecte des d'altres perspectives i a prendre decisions amb més claredat en moments clau. El seu suport ha estat essencial per mantenir la constància i afrontar els contratemps que he tingut durant el desenvolupament del projecte.

REFERÈNCIES

[1] Y. Chen, Z. Liu, B. Ren, and X. Jin, "On efficient constructions of checkpoints," *arXiv:2009.13003*, 2020.

[2] S. R. Amey Agrawal *et al.*, "Inshrinkerator: Compressing deep learning training checkpoints via dynamic quantization," *arXiv:2306.11800*, 2023.

[3] "Numpy," https://numpy.org/, accessed: 2025-06-15.

[4] S. Y. Guoyu Li *et al.*, "Lut-dla: Lookup table as efficient extreme low-bit deep learning accelerator," *arXiv:2501.10658*, 2025.

[5] Global Data Compression Competition, "Gdcc," https://gdcc.tech/, accessed: 2025-06-15.

[6] Google, Open source, "Brotli compression algorithm," https://github.com/google/brotli, 2025, accessed: 2025-06-15.

[7] H. Schmidt, "Ieee754 floating point converter," https://www.h-schmidt.net/FloatConverter/IEEE754.html, accessed: 2025-06-15.

[8] H. Oroskar, "Differential pcm (dpcm) compression technique," https://medium.com/@hitesh.oroskar22/differential-pcm-dpcm-compression-technique-39ae6288474c, accessed: 2025-06-15.

[9] lz4, "lz4: Extremely fast compression algorithm," https://github.com/lz4/lz4, 2025, accessed: 2025-06-15.