Compressió de Dades Astronòmiques

Iván Adame Carmona

24 de juny de 2024

Resum– En aquest treball final de grau (TFG) es presenta l'estudi i desenvolupament d'un sistema de compressió de dades astronòmiques que busca optimitzar el procés de compressió i descompressió utilitzant una nova aproximació al sistema Asymmetric Numeral Systems Binary Contextual (ANS-BC). A través d'una col·laboració amb l'Institut d'Estudis Espacials de Catalunya (IEEC), es van plantejar dues possibles optimitzacions: la utilització d'una Look-Up Table (LUT) i el càlcul de probabilitats a nivell de blocs. La investigació s'ha centrat en la segona opció. Els resultats obtinguts indiquen una millora substancial en els temps de processament amb un lleuger increment en els bits per mostra (bps), confirmant així l'eficàcia de l'aproximació per blocs proposada. Proposant també altra millora a considerar que encara podria optimitzar encara més el procés.

Paraules clau– Compressió Astronòmica, LUT, Compressió, Bloc, Context, AND-BC, IEEC, Optimització de compressió, Descompressió de dades, Eficiència

Abstract– This final degree thesis (TFG) presents the study and development of an astronomical data compression system that seeks to optimize the compression and decompression process using a new approach to the Asymmetric Numeral Systems Binary Contextual (ANS-BC) system . Through a collaboration with the Institut d'Estudis Espacials de Catalunya (IEEC), two possible optimizations were considered: the use of a Look-Up Table (LUT) and the calculation of probabilities at block level . Research has focused on the second option. The results obtained indicate a substantial improvement in processing times with a slight increase in bits per sample (bps), thus confirming the effectiveness of the proposed block approach. Also proposing other improvements to consider that could further optimize the process.

Keywords– Astronomical Compression, LUT, Compression, Block, Context, ANS-BC, IEEC, Compression Optimization, Data Decompression, Efficiency

1 Introducció

La tecnologia de la compressió és adaptada a diversos aspectes del nostre dia a dia, des de les imatges dels nostres teléfons fins a la dels nostres televisors. Altra camp on és important és a les imatges astronòmiques, imatges realitzades a l'espai, les quals donen gran suport als investigadors per entendre l'espai exterior.[1]

La necessitat de compressió eficient es torna encara més evident davant el creixement exponencial dels volums de dades astronòmiques, projectades per assolir l'escala d'Exabytes[2] durant la propera dècada, gràcies als avenços en els telescopis de nova generació. En aquest context, l'eficiència de la compressió es converteix en un factor clau per contenir els costos d'emmagatzematge i millorar els temps de transmissió.

És important destacar que les dades astronòmiques presenten una singularitat que requereix enfocaments de compressió específicament adaptats a la seva naturalesa. La presència de diversos elements en una mateixa imatge, com àrees de gran foscor juntament amb fonts intenses de llum procedents d'estrelles o nebuloses, afegeix complexitat al procés de compressió.

Per tant, comptar amb algorismes de compressió eficients esdevé fonamental per preservar la màxima quantitat d'informació en el menor espai possible, sense comprometre la qualitat de les dades. Això no només maximitza el valor científic de les dades disponibles, sinó que també minimitza els temps de transmissió i garanteix uns costos demmagatzematge sostenibles a llarg termini.

• E-mail de contacte: ivan.adame@autonoma.cat
• Mención realitzada: Tecnologies de la Informació
• Treball tutoritzat per: Joan Bartrina Rapesta (Departament d'Enginyeria de la Informació i les Comunicacions)
• Curs 2023/24

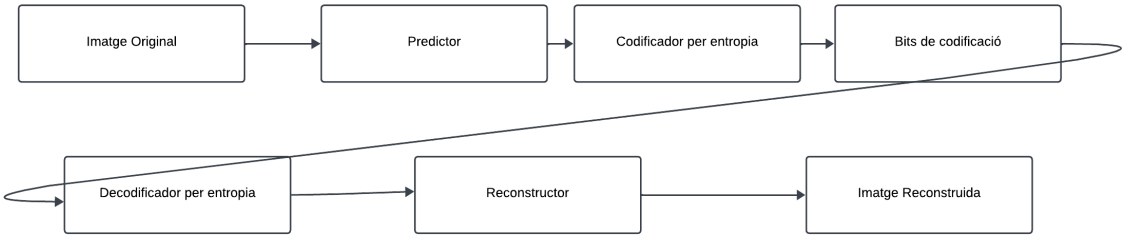

Fig. 1: Procés de codificació d'imatges per entropia

2 OBJECTIUS

Per començar el treball el meu tutor del TFG, en Joan Bartrina Rapesta, em va convocar per tal de tractar el problema amb el qual ens trobàvem i explicar les dues possibles vies que s'havien plantejat per solucionar-ho.

El problema a tractar venia donat en un projecte en col·laboració amb l'Institut d'Estudis Espacials de Catalunya (IEEC)[3] on s'utilitza l'algorisme conegut com a Asymmetric Numeral System Binary Contextual(ANS-BC)[4] aquest algorisme primer realitza el càlcul de probabilitats i després la codificació, la qual cosa fa que el seu volum de càlcul sigui massa gran. No obstant això, és important destacar que l'algorisme codifica d' esquerra a dreta i descodifica de dreta a esquerra. Això dificulta mantenir les mateixes probabilitats en el moment de codificar o descodificar un símbol específic.

Per tant, es van plantejar dues possibles solucions per abordar aquesta problemàtica i reduir la càrrega de treball associada amb l'ús de l'algorisme ANS-BC. Aquestes solucions podrien incloure optimitzacions de l'algorisme existent per reduir la complexitat computacional o la implementació d'algorismes de compressió alternatius que mantinguin un bon rendiment sense requerir un càlcul intensiu de probabilitats.

El procés de selecció i implementació de la solució més adequada requerirà una anàlisi dels requisits del sistema, els objectius de rendiment i les limitacions de recursos, així com una avaluació exhaustiva de les diferents opcions disponibles. Aquesta anàñisi de resultat haurà de garantir que la solució adoptada sigui òptima per al context específic de la compressió de dades astronòmiques i que compleixi amb les expectatives en termes de rendiment, eficiència i qualitat de les imatges comprimides.

2.1 Opció 1 - Utilització d'una Look-Up Table de probabilitats

La primera solució plantejada consisteix en l'ús d'una Look-Up Table (LUT), una tècnica amplament utilitzada en diversos àmbits de la informàtica i el processament d'imatges. En aquest context, es tractaria de generar una taula prèviament calculada que contingui totes les probabilitats per a cada possible context que es pugui trobar en el procés de compressió. Aquesta taula s'utilitzaria posteriorment per assignar directament les probabilitats corresponents durant el procés de compressió, en lloc de calcular-les dinàmicament com ho fa l'algorisme ANS-BC.

En el moment de la presentació d'aquesta opció, es van expressar dubtes sobre la seva adaptabilitat i flexibilitat. Es va senyalar que una LUT és una estructura estàtica que s'aplica a totes les imatges que es volen comprimir, sense adaptar-se específicament a les característiques individuals de cada imatge. Tot i això, també es va reconèixer que les imatges astronòmiques tendeixen a seguir patrons i estructures similars en molts casos, el que podria implicar que la variabilitat entre imatges fos relativament baixa.

Encara que en casos concrets potser no és l'opció més eficient cal tenir en compte per provar com funciona al nostre cas i si s'adequaria correctament, per tant, no és descartable.

2.2 Opció 2 - Càlcul de les probabilitats a nivell de blocs

Després de considerar aquesta possibilitat, va sorgir una altra alternativa que, tot i potser ser més complexa d'implementar, va ser percebuda com més adaptable.

Per entendre millor aquesta opció, és essencial comprendre com funciona un codificador per entropia. Aquests codificadors utilitzen la probabilitat d'aparició d'un símbol per assignar-li un codi; els símbols que s'utilitzen amb menys freqüència s'associen amb codis més llargs, que ocupen més espai, donat que es troben en menys ocasions. En canvi, els símbols que tenen més probabilitat de sorgir es representen amb codis més curts i que ocupen menys memòria.

Així, quan el nostre codificador rep una imatge, transforma els seus píxels en símbols. Per a cadascun d'aquests símbols, es calcula l'entropia, és a dir, la probabilitat d'aparició d'aquest símbol en particular. Per millorar aquest procediment, es fa servir un predictor. Aquest predictor, basant-se en valors previs, pot predir els símbols futurs, permetent al codificador ajustar els codis de manera més eficient segons les tendències detectades. Aquesta predicció ajuda a reduir l'entropia general de la imatge, optimitzant així l'espai de memòria necessari per emmagatzemar-la, podem observar aquest procés a la **figura 1**.

Aquesta solució parteix de l'objectiu de superar el repte presentat per l'ANS-BC. En aquest cas, es tracta de generar una estratègia on es calculen les probabilitats dels diferents valors a partir de la matriu que representa la imatge, per després codificar-la de manera completa. És evident que aquest procés, tal com s'ha plantejat fins ara, resulta extremadament lent, ja que implica calcular les probabilitats i, posteriorment, dur a terme la codificació completa. Així doncs, en aquesta opció es pretén explorar la viabilitat d'optimitzar aquest procés mitjançant una aproximació per blocs de la imatge i la utilització de possibles contextos per assignar valors a les combinacions binàries rebudes fins al moment.

A més, és important destacar que les probabilitats per a cada bloc es guarden com a informació auxiliar. D'aquesta manera, el descodificador podrà descodificar les dades correctament, ja que disposarà de les probabilitats necessàries per interpretar correctament cada bloc de dades. Aquesta incorporació d'informació auxiliar és clau per garantir l'eficàcia del procés de codificació i descodificació en el sistema proposat.

Per exemplificar més el problema i el desitjat he dissenyat els següents esquemes per ajudar al seu enteniment:

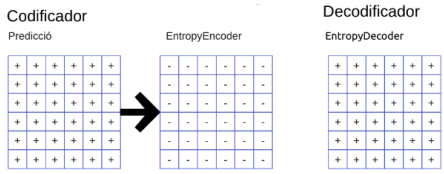

Fig. 2: Funcionament inicial del codi

Com podem observar a la imatge la matriu realitza el càlcul de probabilitats dues vegades realitzant així molt treball de còmput, per tant es desitja realitzar el càlcul només un cop i enviar com a sideInformation les probabilitats segons el seu bloc i context fent que no realitzi cap càlcul a costa d'utilitzar més memoria.

Per representar aquesta idea l'exemplifico amb el següent esquema:

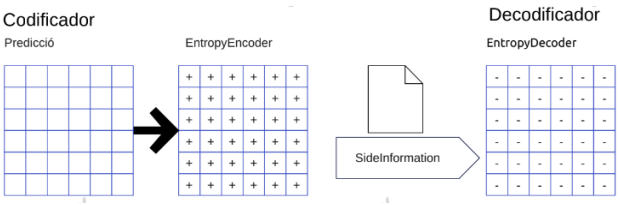

Fig. 3: Funcionament objectiu del codi

3 Metodologia i Planificació

Un cop considerades les dues opcions es va decidir realitzar la segona, encara que la primera també es voldria estudiar i considerar-la perquè potser no és tan rígida com es pensava en un prinicipi, per tant, es començarà implementat la segona opció, aquesta part portarà un temps gran del projecte, ja que s'haurà de familiaritz amb el projecte "Montsec"al complet que és el qual té l'arquitectura amb la qual es treballarà.

Per a la realització d'aquest projecte, es seguirà una metodologia Kanban[6] utilitzant la plataforma Trello. Mitjançant aquest sistema, les tasques seràn dividides en diferents etapes que es representaran en estats com a pendent, en curs i completat. Això permetrà portar un seguiment exhaustiu de tot el que s'haurà realitzat durant el treball, assegurant que cap detall quedi sense implementar.

Un cop dutes a terme les proves pertinents del correcte funcionament del codi desenvolupat es procedirà a fer una anàlisi entre el nou algorisme i l'original considerant característiques com bit-rate, temps, qualitat, etc.

Depenent del punt del semestre en què es trobi quan es finalitzi es podria considerar implementar la LUT i comparar-la amb els resultats de l'altre desenvolupat en comparació a l'original i extreure la conclusió final de quin dels ambdós mètodes seria el més adequat d'una manera empírica.

3.1 Divisió de Tasques

Per plantejar com realitzaria les tasques i les vaig plantejar en 3 grans etapes la primera era l'enteniment del codi, ja que no seria capaç de realitzar cap canvi si no entenc el codi des del qual començo, seguidament realitzaria la codificació(aquesta és l'etapa on ens situem) i per últim el procés contrari, la descodificació, on consegueixo llegir la side Information i decomprimir la imatge per finalment analitzar els resultats de manera adequada comparar els resultats obtinguts amb el codi que havia abans tant de qualitat de la imatge com temps de còmput.

Fig. 4: Panel inicial del Projecte [10]

4 Inconvenients

4.1 Configuració Repositori Git

Encara trobant-nos en una part molt prematura del projecte, van sorgir dificultats per accedir a la branca corresponent de git, ja que la VPN del departament no permitia l'accés, després d'haver anat amb l'informàtic responsable i facilitant accés amb altre IP es va aconseguir accedir, encara no ser un invonvenient massa greu va suposar un endarreriment del projecte.

4.2 Problemes Environment

Un inconvenient que va sorgir durant la segona etapa del projecte va ser la incompatibilitat del sistema operatiu, ja que mentre es desenvolupava en Windows, el projecte requería ser executat des d'un terminal shell de Linux. Inicialment, es va considerar l'ús d'un simulador que proporciona un terminal Linux dins de Windows, però aquest enfrontava la limitació de no poder accedir als arxius de Windows, requerint en canvi que els arxius es guardessin en una partició exclusiva per a Linux. Aquest mètode va resultar ser poc pràctic i potencialment retardador per al progrés del projecte, donat que requeria traslladar els arxius a la memòria reservada al terminal Linux després de realitzar les modificacions inicials al codi.

Per afrontar aquest repte, es va optar per utilitzar un ordinador antic, al qual es va instal·lar Linux, permetent així traslladar tot el projecte a aquest sistema. Aquesta solució va facilitar que els canvis en el codi estiguessin automàticament preparats per a la seva execució, optimitzant així el procés de desenvolupament i evitant retards addicionals.

4.3 Malentés de la Iteració

Un dels inconvenients més grans trobats i que més temps va portar solucionar és no comprendre com funcionava la iteració sobre la matriu en la codificació, més endavant s'explicarà amb més detall el problema, però això va fer que sorgís un endarreriment el qual no estava considerat i, per tant, un dels motius pel que no vaig realitzar la LUT com havia pensat fer en un principi.

5 Estat de l'Art

Després d'haver dut a terme la primera reunió es va a començar a dividir les tasques com s'ha esmentat al punt 3.1 la primera tasca va ser entendre el flux del codi aquest comença pel document App.java en aquest document es realitza la crida a diverses funcions d'altres classes per tal de portar a terme tot el procés de compressió i descompressió de les imatges.

Després de revisar el codi, es va obtenir una visió general del seu funcionament, tot i que van sorgir dubtes respecte a certes variables desconegudes. Per aclarir aquests interrogants, es va realitzar una reunió amb el tutor, on es va entendre que aquests paràmetres són passats a l'script "Montsec.sh"per permetre la realització de funcions específiques. A més, es va determinar la necessitat de crear un nou paràmetre que assegurés que el procés específicament desitjat fos el que es duia a terme.

La funció per la qual comença la part que tractarem és a la funció *compress()* aquesta funció recorre la matriu de la imatge per tal de realitzar l'encriptació segons convingui pel paràmetre passat, actualment hi ha tres cassos encriptació FLW, UABs i l'afegida per mí la qual parteix de la base de la UABs.

Però tindrà la diferència que ara mateix es fa el càlcul de probabilitats segons el context i el bloc pel que crida a la funció *setContextProbsForward()* en realitzar la predicció, però en realitzar la codificació el càlcul de probabilitats també es realitza pel que és poc òptim i consum molt temps i potència de càlcul. Per aquest motiu la meva versió ha de realitzar el càlcul de probabilitats depenent del context i el bloc a la vegada que es realitza la codificació i un cop s'hagi realitzat el càlcul guardar-ho a un document, en aquest cas he escollit que sigui CSV per a després al descompressor sigui més senzill la recerca depenent del bloc i el context, a més del total per realitzar el càlcul de probabilitats, ja que els valors guardats són els casos positius. Un cop tinguem això es comprimirà el fitxer per tal de passar-ho al descodificador com *"side information"*.

Actualment, està implementada la creació del fitxer, però no es pot encara assegurar que funcioni correctament(a abril de 2024) perquè caldrà implementar el procés de descompressió per veure els resultats.

Un cop s'ha realitzat el procés de descompressió i haver solucionat els problemes sorgits durant la codificació d'aquest considerant el valor del PSNR com el camp per determinar si és admissible o no, ja que ha de donar infinit s'han de crear una série de gràfiques per tal d'analitzar els resultats finals obtinguts.

5.1 Desenvolupament del Codificador

Durant la programació de la part del codificador es guarda la *"side information"* va sorgir el dubte per guardar les dades correctament, ja que en una primera versió guardava totes les iteracions per cada píxel, però després d'una reunió amb el tutor es va canviar, ja que es va arribar a la conclusió que havia de guardar l'estat final un cop calculades les probabilitats per cada context i bloc.

5.2 Desenvolupament del Descodificador

Després de la reunió, es va procedir a realitzar la descodificació, la qual va resultar ser més complicada del previst a causa d'un malentès en el recorregut de l'algorisme. Inicialment, es va assumir que la descodificació començava per la part superior esquerra de la matriu; no obstant això, en realitat, l'algorisme iniciava des del final de la matriu i procedia cap al començament. Aquest error en la comprensió del recorregut va resultar en resultats incorrectes en el decodificador fins que es va detectar i corregir l'error.

5.3 Recollida de Dades

Com es va esmentar en el plantejament inicial d'aquest projecte, un cop finalitzada la codificació de tot el procés de compressió i descompressió, serà necessari dur a terme la recollida i anàlisi de resultats.

Diverses metodologies van ser considerades per facilitar la recollida de dades de manera que permetessin observar clarament el comportament del programa. La recollida de dades es realitza utilitzant el script *"Montsec.sh"*, que és responsable d'executar tot el procés. Aquest script recopila dades com els bits per mostra (bps) i el temps en segons de cada part del procés, i posteriorment aquesta informació és enregistrada en un fitxer CSV per organitzar-la de manera eficient.

5.4 Anàlisi de Dades

Un cop recollides les dades, es va considerar necessari establir un mètode per visualitzar-les de manera senzilla i eficaç. Inicialment, es va plantejar l'ús de la funció de gràfics integrada als fulls de càlcul de Google. No obstant això, es va determinar que seria més apropiat utilitzar un script en Python que llegiria les dades des d'un fitxer CSV i generaria gràfics específics per a cada cas.

A més, es va valorar que les gràfiques de dispersió[9] serien especialment adequades per aquesta tasca, atès que aquest tipus de representació ja havien sigut utilitzades en l'assignatura *"Aprenentatge Computacional"*. Aquestes gràfiques permeten una visualització clara de cada punt que representa dades d'una imatge, facilitant la identificació de les zones on es concentra el major nombre de punts, i per tant, on es manifesten els comportaments més freqüents dins del conjunt de dades analitzades.

5.5 Extracció de conlusions

Un cop es tinguin les gràfiques pertinents es podrà valorar quin ha sigut el resultat del treball si ha sigut satisfactori, desitjat o per altra banda no ha sigut assolit.

Fig. 5: Imatge utilitzada per realitzar les proves de compressió

I a partir d'aquí extreure les causes del perqué han hagut els resultats obtinguts.

6 Canvis Realitzats al codi

En aquesta secció, es descriu la fase d'*"experimentació"* d'un projecte en l'àmbit de la informàtica, és a dir, en la programació. El codi inicial, que va ser proporcionat pel centre educatiu, requeria una optimització significativa del seu rendiment. Un enfocament particular d'aquesta optimització va ser l'eliminació de la duplicació, fins aquell moment necessaria, del càlcul de probabilitats però s'havia de canviar el procés per tal de que ja no fos necessari realitzar-ho dues vegades i, per tant, augmentava la càrrega computacional del sistema.

Per aprofundir en aquest càlcul, és essencial introduir dos conceptes fonamentals: el bloc i el context. El concepte de bloc es refereix a la subdivisió d'una imatge en unitats més petites, facilitant així el procés de compressió. Aquestes unitats es poden dividir en dimensions estàndard com 8x8, 16x16 o 64x64 píxels, depenent de les necessitats específiques de l'algorisme de compressió utilitzat.

Respecte al context, la seva definició es pot il·lustrar fent referència a un article coescrit pel tutor del projecte, Joan Bartrina Rapesta, titulat "A Lightweight Contextual Arithmetic Coder for On-Board Remote Sensing Data Compression"[11]. Segons aquesta font, el context d'una imatge, en el marc de la codificació, es defineix com la configuració específica de mostres adjacents i components espectrals anteriors, que s'utilitzen per modelar i estimar la probabilitat dels bits a codificar. Aquesta configuració permet un modelatge més precís de les probabilitats en els algoritmes de compressió de dades.

L'ús d'un algoritme basat en probabilitats(o entropía), com l'ANS-BC en aquest projecte, implica que l'eficàcia de la compressió depèn substancialment del model probabilístic aplicat. En aquest cas, el context facilita una millora significativa en les estimacions de probabilitat. A més, la subdivisió de la imatge en blocs contribueix a optimitzar el rendiment global del procés de compressió.

6.1 Codificador

A continuació es descriu la fase inicial del desenvolupament especificada a l'apartat 5, que va implicar una modificació del script Montsec.sh. Aquesta modificació va consistir en l'addició d'una nova opció de codificació basada en l'estructura preexistent denominada uABS. Inicialment, es va ajustar el codi per iniciar el procés de codificació des de les posicions inicials de la matriu, és a dir, x=0 i y=0. No obstant això, com es detalla en el punt 5.2, es va descobrir que l'aproximació apropiada era començar des del final de la matriu, una correcció fonamental per al correcte funcionament del codificador.

La principal funció d'aquesta secció del codi és la codificació dels símbols presents en la imatge, així com el càlcul de les probabilitats basades en el context i la mida del bloc. Aquesta part del codi incorpora un canvi significatiu, que consisteix en l'emmagatzematge de les probabilitats obtingudes durant el càlcul per ser enviades al decodificador com a *Side Information*.

Inicialment es va considerar l'ús d'un fitxer de text pla (.txt) per emmagatzemar dues variables essencials per al càlcul de les probabilitats: contexts0Block i contextTotalBlock. La primera variable es guarda el valor del nombre de zeros codificats en un bloc i context especifics, mentre que la segona porta el compte del nombre total de símbols(no només zeros). Posteriorment, es va optar per la utilització d'un fitxer en format .csv. Aquests tipus de fitxers separen la informació mitjançant el caràcter ";", van ser escollits a utilitzar-los per la seva facilitat a l'hora de recuperar dades durant el procés de descompressió.

6.1.1 Funció Modificada

La funció modificada ja partia d'una funció base com s'ha esmentat abans i té com objectiu codificar dades utilitzant el model de codificació entròpica adaptativa anomenat uABS adaptat per procesar diferencies entre píxels originals i predits en imatges. A la funció se li passa com a paràmetre la variable *"dataToCode"* que respresenta les diferències entre els pixels predits i originals.

La codificació es realitza bit a bit per cada pixel en cada plà de bits(bitplane) començant des del més signi-

ficatiu fins el menys significatiu definit per la variable **CONS.MAXBITDEPTH** i comença la iteració des del cantó inferior dret(width-1 i height-1) fins arribar al cantó superior esquerre(0 i 0).

Per cada pixel i cada bit es determina el valor actual a la variable *"symbolToCode"* utilitzant una operació d'enmascarament per aïllar el bit a processar en la posició actual del bitplane i desplaçament(o shifting) i si el bit actual és major o igual que un umbral de byPass establert s'obté un context a partir dels bits codificats fins al moment.

Aquesta obtenció del context es realitza a la funció *"getContextFromPreviousSamples"* de la classe **ContextModelling** que utilitzant les coordenades actuals del píxel i el context obtingut calcula l'índex del bloc al que pertany el píxel.

Basant-se en aquest context i aquest índex de bloc, es calcula la probabilitat. Aquesta probabilitat s'emmagatzema en un fitxer CSV; la estructura plantejada per aquest fitxer consisteix en guardar, en la primera columna, l'identificador del context, en la segona, l'índex del bloc, seguit pel nombre de zeros codificats i el total de símbols codificats en una altra columna. Aquesta estructura permetrà, posteriorment, calcular la probabilitat del símbol a descodificar en el moment de recuperar aquesta informació, ja que sabent la probabilitat que sigui 0 en un context i un bloc especifics també podrem saber la de 1, ja que serà la inversa(1 - probabilitat de ser 0).

6.2 Descodificador

El decodificador, un element crític en el sistema de compressió, serveix com a contrapartida indispensable del codificador, donant sentit i funcionalitat a tot el procés de compressió de dades. El principal objectiu del decodificador és la reconstrucció exacta de la imatge comprimida, amb l'objectiu de minimitzar qualsevol pèrdua de qualitat, tot mantenint un equilibri òptim entre l'eficiència del processament i l'eficàcia del resultat final.

Aquesta tasca requereix que el decodificador segueixi un procediment molt semblant al del codificador, per la qual cosa la correspondència precisa de les probabilitats entre ambdues fases és crucial. En la fase de codificació, a mesura que es van acumulant casos positius (o zeros), el decodificador ha de realitzar l'operació inversa, és a dir, restar aquests casos en l'ordre invers al que van ser afegits. Així, el procés de decodificació s'inicia a la part superior esquerre de la matriu (x = 0 i y = 0) i avança seqüencialment fins a completar la imatge.

Cal destacar que, abans de l'implementació actual, el procés de decodificació incloïa un re-càlcul de probabilitats, essencialment duplicant una operació ja realitzada durant la codificació.

L'implementació realitzada elimina aquest pas redundant, permetent que el decodificador utilitzi directament les probabilitats registrades durant la codificació. Aquesta optimització pot comportar un lleuger sacrifici en termes de *"bits per mostra"*, però resulta en un procés més ràpid i eficient, promovent un millor equilibri entre l'ús dels recursos i la qualitat de la imatge reconstruïda. Així, l'actualització representada per aquesta nova implementació no només millora l'eficiència operativa del sistema sinó que també optimitza la fidelitat amb què es reconstrueixen les imatges originals, culminant en un rendiment generalment superior.

6.2.1 Funció Modificada

En relació amb la funció del decodificador, aquest rep la variable *"dataToCode"* com a paràmetre, de manera similar al codificador. La diferència principal radica en que rep *"outputData"*, un array tridimensional, juntament amb un paràmetre z, que s'utilitza per determinar la banda on es desa el resultat decodificat. A diferència del codificador, el decodificador inicia el procés des del cantó superior esquerre de cada imatge (x = 0 i y = 0) i avança fins al final (x = height - 1 i y = width - 1).

Aquest procés també inclou un bypass establert, que determina si es requereix calcular el context específic o simplement procedir amb la decodificació utilitzant una probabilitat predefinida. Abans de procedir amb la decodificació, és necessari carregar les dades des de la *"Side Information"* rebuda, la qual es processa per omplir l'objecte de la classe **ContextModelling** amb els valors exactes registrats al final de la codificació. Això implica la implementació de mètodes setters dins la classe per assignar els valors a les variables corresponents.

Per confirmar la correcció de les dades carregades, es va generar un fitxer CSV auxiliar per comparar si les dades coincidien exactament amb les proporcionades pel codificador. Un cop verificada la igualtat, el decodificador procedeix seguint el funcionament establert en el codi.

El càlcul dels contextos es realitza basant-se en les mostres prèvies, similarment al que fa el codificador. Aquesta coherència és crucial, ja que qualsevol discrepància en la metodologia de càlcul dels contextos podria resultar en una descompressió incorrecta de la imatge.

Finalment, un cop calculats els contextos, els índexs dels blocs i les probabilitats (la probabilitat que el bit sigui 0), el decodificador utilitza el context i la probabilitat determinada per decodificar el símbol més semblant al d'origen, essencialment reproduint l'original.

El procés conclou amb la combinació del bit decodificat a la posició correcta del valor del píxel, utilitzant una operació OR amb una màscara de bits per assegurar que la informació sigui inserida correctament dins de la imatge reconstruïda.

6.3 Procés d'Experimentació

En el desenvolupament d'aquest codi, no va ser possible realitzar una avaluació dels resultats fins a la finalització completa tant del codificador com del decodificador. Especialment en el cas del decodificador, es van realitzar diverses execusions fins que els resultats obtinguts fossin els esperats. Aquest objectiu es va definir amb el criteri que el ràtio senyal-soroll de pic (PSNR) arribés a infinit, una situació que indica que no existeix cap diferència perceptible entre la imatge original i la imatge descomprimida, degut a que l'error quadràtic mitjà (MSE) era zero, representant el resultat ideal.

Aquesta metodologia de *"prova i error"* va ser adoptada com a estratègia principal per verificar l'assoliment dels objectius establerts, optimitzant així el procés fins a obtenir una reproducció exacta de la imatge original a través del sistema de codificació i descompressió desenvolupat.

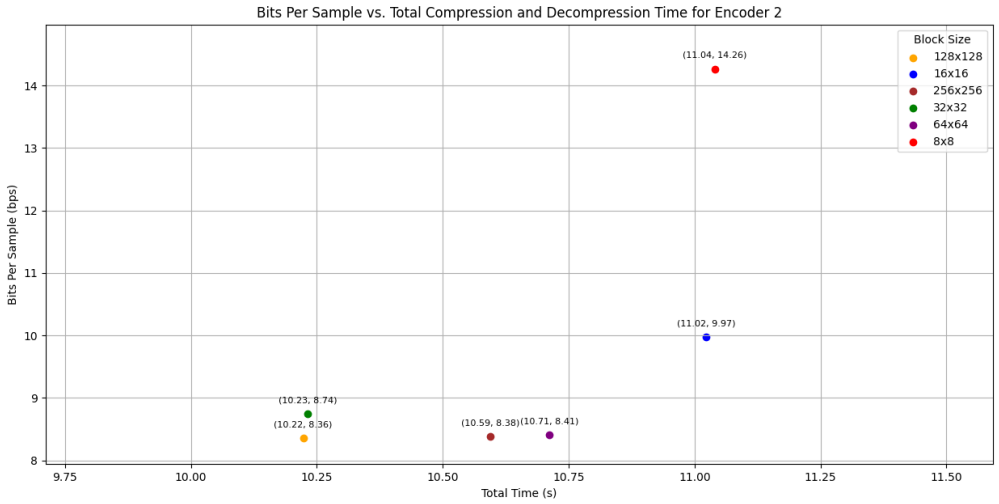

Fig. 6: Gràfica de distribució per Tamany de blocs

7 Anàlisi de Dades

Com es va esmentar en la fase inicial del projecte, la part final del mateix està dedicada a l'anàlisi dels resultats obtinguts. Finalitzat el desenvolupament del codi, es va procedir a una reunió amb el tutor del treball amb l'objectiu de determinar els aspectes més rellevants per a l'anàlisi i per a la posterior extracció de conclusions pertinents.

Prèviament a aquesta trobada, es va preparar un gràfic de dispersió. Aquest tipus de gràfic va ser seleccionat com l'opció més adequada per representar els resultats, ja que la localització del major nombre de punts proporcionava una visualització directa de l'eficàcia de l'optimització implementada. Els eixos del gràfic van ser definits de la següent manera: l'eix d'abscisses representava el temps total (sumant la codificació i la decodificació), mentre que l'eix d'ordenades mostrava els bits per mostra (bps) de la imatge junt amb els de la informació auxiliar comprimida. La iniciativa de comprimir la informació auxiliar va ser suggerida pel supervisor, qui va recomanar l'ús de l'Algoritme Lempel-Ziv-Markov (LZMA) [12], que va resultar ser molt més eficient comparat amb el mètode més comú ZIP. Després de diverses execucions amb ambdós algoritmes i de calcular la mitjana dels resultats, es va determinar que LZMA oferia una eficiència superior en un 35% respecte a ZIP, motiu pel qual es va optar per aquest algoritme per a la compressió de la informació auxiliar.

A partir de la reunió amb el supervisor, es va identificar la necessitat de produir tres tipus diferents de gràfics. El primer, un gràfic utilitzant un conjunt d'imatges per determinar el tamany de blocs amb el qual el codificador oferia el millor rendiment. Seguidament, es va generar un nou gràfic amb el tamany de blocs més òptim, identificat a partir del primer anàlisi, comparant temps i bps de ambdós codificadors amb un gran conjunt d'imatges. Finalment, es va elaborar un tercer gràfic que comparava els temps de codificació i decodificació per cada tamany de bloc, utilitzant tamanys múltiples de 8, que són els més utilitzats habitualment en aquest context.

7.1 Recerca del millor tamany de Blocs

En l'estudi representat en la gràfica de dispersió, es mostra cada mida de bloc com un punt individual s'ha utilitzat la nomenclatura "Encoder 2"per identificar el nostre codi proposat, ja que es l'identificador que té l'script per reconeixer-de quin codificador es tracta, de la mateixa manera "Encoder 1"es refereix al codi original. Les dades per a cada punt es deriven d'una mitjana calculada a partir de múltiples mesures, que s'han registrat en un arxiu CSV. Aquest arxiu ha estat processat utilitzant un script en Python, empleant la biblioteca *Matplotlib* per facilitar la visualització de les dades, com es pot observar a la **figura 6**.

La metodologia emprada per a recollir les dades ha sigut la execució repetida del procés de compressió i descompressió per cada mida de bloc, assegurant així que els resultats siguin robustos i representatius. Aquest enfocament està dissenyat per minimitzar l'error de mesura i proporcionar una estimació fiable de la relació entre el temps total de processament i els bits per mostra (bps) generats.

Es fa evident que els blocs de mida 8x8 són considerablement menys eficients. Aquesta menor eficiència es deu principalment al temps addicional requerit per escriure i processar les 'dades auxiliars'. Aquesta demora resulta en un augment significatiu de la mida de les dades, duplicant gairebé els bps en comparació amb les altres mides de blocs estudiades.

Per altra banda, les mides de bloc de 32x32 i 128x128, mostren una eficiència notablement superior. Aquests blocs poden processar una gran quantitat de dades simultàniament, el que no només redueix el temps de processament requerit sinó que també disminueix la quantitat de dades auxiliars necessàries. Basant-nos en aquest anàlisi, es recomana optar per una mida de bloc que optimitzi tant el temps de processament com els bps. Observant els resultats, la mida de bloc de 128x128 emergeix com la més òptima, ja que se situa pròxima a l'origen en ambdós eixos, i per tant, serà la mida seleccionada per a realitzar comparacions més detallades en futurs estudis comparatius entre diferents codificadors però degut a la seva proximitat, també es considerarà l'ús de blocs 32x32. Les altres mides de bloc ana-

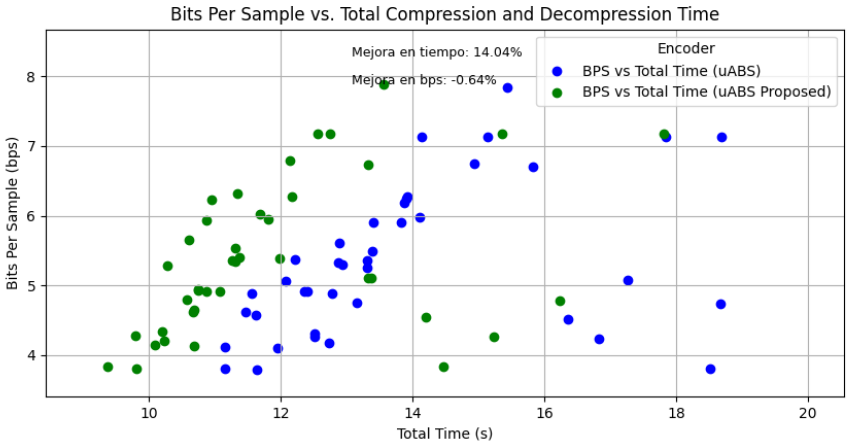

Fig. 7: Gràfica de distribució per comparar resultats entre els dos codificadors amb blocs de 128x128

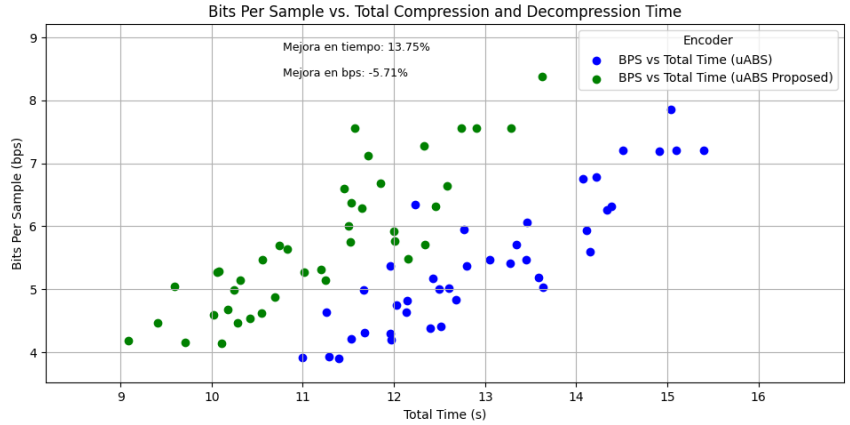

Fig. 8: Gràfica de distribució per comparar resultats entre els dos codificadors amb blocs de 32x32

litzades mostren un rendiment que varia des de mitjà a poc eficient. Els blocs de 16x16, encara que no assoleixen els nivells de bps i temps dels blocs de 8x8, presenten un rendiment significativament superior al de la mida més petita. Per altra banda, els blocs de 256x256 i 64x64 ofereixen una relació temps-dades equilibrada, posicionant-se com opcions viables quan no es pot utilitzar la mida de 128x128 ni 32x32, ja que són un terme mitjà.

7.2 Comparació de resultats entre els dos codificadors

Una vegada establert que la mida de bloc de 128x128 és l'ideal per al codificador en qüestió, s'ha procedit a implementar aquesta configuració en un conjunt ampliat d'imatges, utilitzant tant el codificador optimitzat com el codificador original. S'han registrat meticulosament els temps de processament i els bits per mostra (bps) generats per cada codificador. Els resultats obtinguts són exposats detalladament a la **figura 7**. També s'ha realitzat l'anàlisi amb els blocs de 32x32 que podem veure a la **figura 8**

L'anàlisi dels resultats indica una millora substancial en els temps de compressió i descompressió quan es fa ús del codificador proposat. Un càlcul detallat de les mitjanes de temps per a cada codificador revela una millora del 14,04% respecte a la versió original del codi, respecte als blocs de 128x128, mentre que per als de 32x32 hi ha una millora del 13,75%. Aquesta millora en el rendiment confirma l'èxit en el compliment dels objectius establerts inicialment per al projecte.

Addicionalment, és important destacar un increment lleuger en els bps, específicament d'un 0,64% amb els blocs de 128x128, però d'un 5,71% amb els de 32x32 aquí sí que hi ha una diferència rellevant entre ambdós tamanys de bloc per la qual cosa els blocs de 128x128 a més de ser lleugerament millors en rendiment de temps són també millors als dels bps significativament. Considerant l'augment dels blocs de 128x128 es considera menor i es valora com marginal en comparació amb la millora notable en el temps de processament. Aquesta observació permet inferir que, malgrat un augment lleuger en la mida de les dades, els beneficis derivats de la reducció en el temps de processament justifiquen aquest petit increment.

No obstant això, seria prudent realitzar un estudi més detallat per determinar fins a quin punt aquesta millora en el temps de processament és sostenible en relació amb l'increment dels bps. S'ha de tenir en compte que les imatges processades en aquest projecte són substancialment similars, com es va suggerir al començament d'aquest estudi, per la qual cosa podria ser més avantatjós prioritzar la millora en el temps de processament encara que això impliqui una eficiència de compressió lleugerament reduïda.

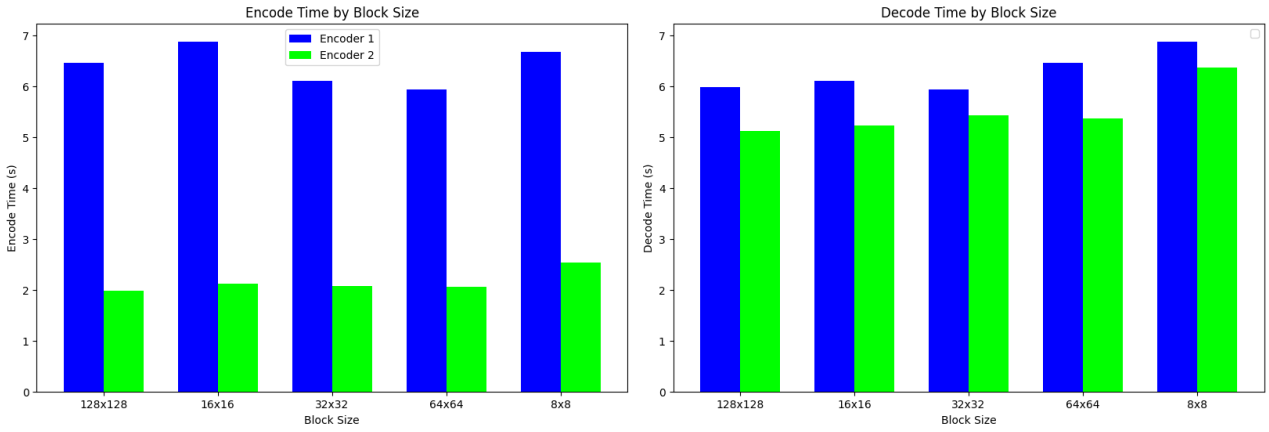

Fig. 9: Gràfica de comparació de temps de codificació i decodificació

7.3 Comparativa dels temps de codificació i decodificació

Analitzant en profunditat la **figura 9**, que mostra la divisió del temps dedicat a la compressió i la descompressió segons els diferents tamanys de blocs utilitzats amb dos codificadors diferents, s'observa una millora notable en l'eficiència del nou codificador en comparació amb l'original. La reducció en el temps de compressió del nou codificador es pot atribuir a una optimització específica: l'eliminació de la necessitat de calcular probabilitats dues vegades, una modificació que, si bé accelera la compressió, té un impacte menor en la descompressió. El temps de descompressió, tot i ser més curt en el nou codificador, no mostra una disminució tan pronunciada com la compressió. Aquest fenomen suggereix que les optimitzacions implementades afecten principalment la fase de compressió.

El fet que el nou codificador requereixi localitzar posicions específiques dins del fitxer CSV per processar les dades pot ser una font d'ineficiència que encara persisteix. Aquest aspecte es converteix en un punt crític d'oportunitat per a la millora en versions futures del codificador. La discussió detallada d'aquestes oportunitats d'optimització s'abordarà més endavant en l'apartat de **conclusions 8** del document.

Complementàriament, l'anàlisi de la gràfica que correlaciona els bits per mostra (bps) amb els temps de compressió per cada mida de bloc (vegeu **figura 6**), revela que els temps d'execució s'incrementen de manera inversament proporcional a la mida dels blocs. Aquesta tendència indica que els blocs més petits requereixen un nombre major d'operacions de processament per unitat de temps, el que pot resultar en un augment del temps total de processament. Aquesta relació és evident també en les gràfiques proporcionades, on els blocs més petits mostren els temps més elevats.

Moltes d'aquestes imatges no eren prou grans per justificar l'ús de blocs tan grans com 256x256 per aquest motiu no es troben representades.

8 Conclusions

A partir de l'anàlisi de les diverses gràfiques generades durant el projecte, arribem a la conclusió que, tot i haver-se assolit l'objectiu d'optimitzar el procés global evitant la recalculació de les probabilitats, encara existeixen àrees on el codi podria ser millorat significativament. Això podem veure-ho durant el procés de lectura de les dades des del fitxer CSV de *"Side Information"*, on és necessari construir l'array utilitzant els camps c i b, que representen el context i el bloc, respectivament. Aquest mètode presenta la limitació de requerir que aquests valors específics es trobin al fitxer per assegurar que la informació es recupera i emmagatzema exactament en la mateixa posició en què es va registrar originalment.

Aquesta problemàtica es va tractar amb el tutor del treball, qui va suggerir que potser seria innecessari emmagatzemar explícitament aquests valors. Alternativament, es podria simplificar el procés recorrent l'array de la mateixa manera com es fa durant l'escritura, assegurant així que la informació es recuperi des de la mateixa posició on va ser guardada inicialment. Malgrat que aquesta millora es va proposar tard en el projecte, és important considerar-la per a futures implementacions perquè potencialment podria augmentar significativament l'eficiència del sistema.

Addicionalment, es va plantejar la possibilitat de desenvolupar una Taula de Cerca (Look-Up Table, LUT) per gestionar aquests valors de manera més eficient. Aquesta metodologia permetria accedir de manera immediata a les dades ja calculades, disminuint la complexitat operacional i millorant el rendiment del procés de compressió. Desafortunadament, aquesta alternativa tampoc va poder ser implementada a causa del temps que disposava quan vaig acabar la opció escollida.

Però en definitiva, el model d'optimització proposat ha sigut exitós amb una millora del 14%, sobretot quan es treballa amb blocs de 128x128 i 32x32, però amb aquests ultims l'empitjorament que hi ha als bps mentre que amb els de 128x128 és negligible amb els de 32x32 potser depenent del que considerem negligible no s'aplica en aquest cas, ja que empitjora un 5% al costat d'un 0,6%.

AGRAÏMENTS

Voldria dedicar un petit agraïment al meu tutor Joan Bartrina Rapesta per la seva guia durant el treball, ja que sempre ha tingut molt clar el que s'havia de fer i m'ha ajudat a tenir-ho a mí clar i saber el camí que havia de seguir a més de respondre de manera freqüent a tots el missatges que li he deixat durant aquest temps.

REFERÈNCIES

[1] ArXiv, "Sky Surveys", Djorgovski, 2012 - https://ar5iv.labs.arxiv.org/html/1712.00012

[2] Data Science Journal, "Astronomy in the Big Data Era", 2023 - https://datascience.codata.org/articles/10.5334/dsj-2015-011

[3] Institut d'Estudis Espacials de Catalunya - https://www.ieec.cat

[4] Asymmetric Numeral System Binary Contextual - https://en.wikipedia.org/wiki/Asymmetric_numeral_systems

[5] Look-Up Table - https://en.wikipedia.org/wiki/Lookup_table

[6] Kanban - https://es.wikipedia.org/wiki/Kanban

[7] T. Bonny and J. Henkel, "Efficient Code Density Through Look-up Table Compression,"2007 Design, Automation Test in Europe Conference Exhibition, Nice, France, 2007, pp. 1-6, doi: 10.1109/DATE.2007.364390. keywords: Table lookup;Dictionaries;Embedded system;Energy consumption;Huffman coding;Reduced instruction set computing;Application software;Costs;Read only memory;Statistical analysis,

[8] Maireles-González, O., Bartrina-Rapesta, J., Hernández-Cabronero, M., Serra-Sagristà, J. (2023). Efficient Lossless Compression of Integer Astronomical Data. AstronomicalIntegers20230731.pdf. Retrieved from https://deic.uab.cat//~jbartrina/docs/articles/AstronomicalIntegers_20230731.pdf

[9] Gràfiques de dispersió - https://es.wikipedia.org/wiki/Diagrama_de_dispersi%C3%B3n

[10] Trello del Projecte - https://trello.com/b/8mKUX8Vx/tfg

[11] J. Bartrina-Rapesta, I. Blanes, F. Aulí-Llinàs, J. Serra-Sagristà, V. Sanchez, and M. W. Marcellin, "A lightweight contextual arithmetic coder for on-board remote sensing data compression- https://https://deic.uab.cat/~jbartrina/docs/articles/ccsds-123nearlossless.pdf

[12] Lempel-Ziv-Markov Algorithm - https://es.wikipedia.org/wiki/LZMA