Seminarski rad: Kompresija slike

Irhad Fejzić, *TKN, Odsjek za matematiku, PMF Sarajevo*

***Abstract*—U digitalnoj obradi slika, kompresija nastoji da umanji veličinu podataka za efikasnije pohranjivanje i prenos preko uređaja. U radu su opisane poznate tehnike kompresija digitalnih slika poređenjem finalnih rezultata i njihovih veličina. Kroz rad analiziraju se postojeće tehnike za kompresiju sa gubitkom i bez gubitka i detaljnije se uvodi u poznati algoritam Huffmanovog kodiranja. Sa rastućim uticajem cloud tehnologije i "remote" poslova dolazi do problema efikasnijeg slanja podataka putem interneta. Jedno od mogućih rješenja, kompresija slika, nudi razne algoritme koji će se daljnje istražiti kroz rad i uporediti nad klasičnim primjercima u obradi slika. Smanjivanjem memorijske veličine slike sa minimalnim gubitkom kvalitete dobija se i efikasan algoritam za kompresiju.**

***Ključne riječi*—Huffman, kompresija slike, tehnike kompresije**

I. Uvod

KOMPRESIJA SLIKE je tehnika i proces umanjivanja veličine slike sa ciljem što optimalnije pohrane i efikasnijeg prenosa između uređaja. Algoritmi koji se primjenjuju za kompresiju pokušavaju da uklone neželjene informacije na slici bez značajnog gubitka kvalitete, a kao rezultat dolazi do manjeg troška memorije. Podrazumijeva se da podaci originalne slike mogu sadržavati dio suvišnih podataka koje ne predstavljaju bitan dio same slike ili se ti isti podaci ponavljaju više puta. Brisanjem takvih podataka smanjujemo i veličinu bez drastičnog umanjenja kvalitete. Dobijena slika koristi izmjenjene podatke karakterisane sa manjim brojem bitova od originalnog primjerka. Prilikom transmisije tih podataka uz specifične i drugačije tehnike kompresije moguće je rekonstruisati originalnu sliku i kvalitetu sa procesom dekompresije. U području medicine, nauke i umjetnosti ovo je jedina validna opcija jer gubitak informacija na slici mora se izbjeći. Takva vrsta kompresije ne donosi iste rezultate kao prvobitna kompresija koja uklanja informacije za veće umanjenje [1]. Kod poređenja algoritama kompresije slika bitan faktor pored same slike je i vrijeme potrebno da se neka slika umanji bez gubitka kvalitete, tj. brzina kompresije u mnogim slučajevima zavisi od samog procesora uređaja [2] i od formata slike. Tokom testiranja i analize, slike *png* formata su zahtjevale više vremena prilikom kompresije od ostalih tipova što će se biti daljnje objašnjeno. Algoritam Huffmanovog kodiranja se smatra jednostavnim i relativno brzim za razliku od drugih tehnika. Korist ovakvog algoritma, pored što se koristi kao modifikacija u kompleksnijim metodama kompresije [2], je evidentna i u kriptografiji i sigurnosti podataka. Tekst je raspoređen na tri sekcije. U prvoj sekciji je opisan generalni uvod kompresije podataka i slika. Druga sekcija opisuje detaljnije objašnjenje Huffmanovog algoritma i poredak sa ostalim popularnim algoritmima. Zadnji dio uvodi razne tehnike poput Deflate, Burrow-Wheeler i LZMA algoritma da detaljnije uporedi efikasnost i umanjenje koje se može očekivati od kompresije bez gubitka kvalitete.

II. Pregled literature

Tehnike kompresije slika se kategorišu u dvije fundamentalne grupe, kompresije sa gubitkom i kompresije bez gubitka. Korištenjem kompresije sa gubitkom dolazi do gubitka informacija ili prelaza u šum tokom procesa kompresije slike [2] što daljnje onemogućuje povratak tih podataka. U drugom slučaju, kompresija bez gubitka zadržava te informacije čak i nakon njene kompresije. Ovakva tehnika omogućava rekonstrukciju originalne slike od kompresovanih podataka, ali jačina kompresije je drastično slabija za razliku od prvog načina. Kompresija bez gubitka koristi se kod izvršnih datoteka, tekst fajlova, izvornog koda i sličnih situacija gdje je veoma bitno zadržati identične podatke koji se kasnije mogu vratiti [3]. U mnogim slučajevima (kao slanje podataka preko interneta) gubljenje preciznosti prilikom rekonstrukcije nije problem. Lošija rekonstrukcija kvalitete govora i zvuka prilikom prenosa ili nakon pohranjivanja može se tolerisati zavisno od primjene, poput razgovora na telefonu [4]. Takve kompresije su izuzetno bitne i kod kompresija slika, ali dolaze sa velikim nedostatkom. U medicini, prenos slika preko interneta i njihovo arhiviranje olakšava doktoru dijagnozu problema pacijenta. Za kompresovane slike sa gubitkom doktor neće biti u stanju da primijeti razliku pogoršanja kvalitete slike i medicinske slike pacijenta [5]. U takvim situacijama gdje je preciznost bitnija potrebno je koristiti kompresiju bez gubitka, ali daljnja istraživanja donose dobre rezultate sa manjim gubitkom kvalitete koji dovode i do bolje transmisije podataka u medicini i sličnim područjima [6]. U sekciji testiranja i analize metoda kompresija, testne slike sastoje se od formata *.png* i *.tiff*. Ovi formati su poznati kao slike bez gubitka i testiranje na njima je pogodno pošto očuvaju rezoluciju i informacije slike, ali sa dodatnim opcijama moguće je umanjiti kvalitetu što neće biti odrađeno u ovom radu [7]. Za razliku od standardnog *.jpg* formata zauzimaju veći prostor na uređaju i kompresija se vrši bez gubitka. Kod kompresije sa gubitkom koriste se posebne metode poput Diskretne cosine transformacije (DCT) ili Wavelet kompresije koje se isključivo koriste za formate pogodne za takvu kompresiju poput *.jpg*, *.webp* i *.bpg* [8]. DCT kompresija je jedna od najkorištenijih kompresija za *.jpg* slike. Metoda funkcioniše tako što razdvaja sliku u dijelove koristeći diskretnu kosinusnu transformaciju i prikazuje vrijednosti slike u frekvencijskoj domeni [8]. U takvom procesu dolazi do kvantizacije slike gdje se visoke frekvencije kosinusoida izbacuju pošto one manje utjecaju na podatke slike. Formiranjem DCT matrice, koja će se koristiti za originalne slike, pomoću formule na Slici 1 kreira

$$f(x) = \begin{cases} \frac{1}{\sqrt{N}}, & \text{za } i = 0 \\ \sqrt{\frac{2}{N}} \cos \frac{(2j+1)1\pi}{2N} & \text{za i > 0} \end{cases}$$

Slika 1: Generisanje DCT matrice, dokazano u [9]

se diskretna kosinusna transformacija množeći matricu M (vrijednosti slike sivih tonova od *0* do *255*) sa DCT matricom T gdje se dobija vrijednost

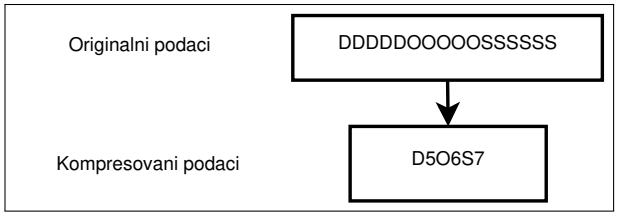
$$D = T * M * T'.$$

Proces kvantizacije vrši podijelu vrijednosti svakog elementa matrice D sa elementom iz kvantne matrice. Rezultat će proizvesti matricu koja sadrži niže vrijednosti nad koje je ljudsko oko više osjetljivo, dok nule na matrici predstavljaju više frekvencije koje se mogu izbaciti i samim tim uštediti na memoriji. [10].

$$\begin{bmatrix} 5 & 2 & -3 \\ 1 & 0 & 3 \\ 2 & 2 & -1 \end{bmatrix} \xrightarrow{\text{Kvantizacija}} \begin{bmatrix} 78 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix}$$

Slika 2: Primjer - kvantizacija matrice

Primjer na Slici 2 prikazuje rezultat kvanzitacije matrice koja sadrži vrijednosti visoke frekvencije (u matrici elementi 0). Takvi elementi i detalji slike se kroz proces kompresije sa gubitkom eliminišu za dodatnu uštedu memorije. Pošto ove tehnike imaju veću primjenu od kompresija bez gubitke, u sekciji testiranja i analize koristit će se popularni python paket *PIL* za poređenje sa prethodnim metodama. Jedna



Slika 3: RLE - Kompresija slika bez gubitka

od jednostavnijih metoda kompresije bez gubitka naziva se Run Length Encoding (RLE). Na Slici 3 dolazi do pronalaska sekvenci identičnih simbola (piksela) i grupisanja tih simbola (Flag karakter) sa brojem njihove frekvencije (Bajt ponavljanja) [11]. Sa ovim informacijama dekompresija će biti brza i finalna slika će biti identična originalnoj. Glavni nedostatak koji korisnik može očekivati je slaba kompresija kao i lošiji rezultat ukoliko se slika sastoji od raznih boja i vrijednosti piksela, tj. slika nije sivih tonova. Ovakva metoda je optimalna ukoliko podaci sadrže velike sekvence istih karaktera. Jedan od primjera je tekst koji se sastoji od velikog broja razmaka ili jednobojnih slika. Na Slici 3 korištenjem ove metode smanjujemo potrebnu memoriju za 12 bajta. Sa rastom rezolucija kao i kvaliteta slika, ovakva tehnika kompresije nije pogodna za svakodnevnu transmisiju podataka. U ovoj sekciji detaljnije se prolazi kroz drugu proceduru kompresije bez gubitka pod nazivom "Huffmanovo kodiranje". Procedura se zasniva na dva posmatranja za pronalazak optimalnih prefiksnih kodova:

1) Očekuje se da veća frekvencija simbola sadrži kraće kodove od simbola koji se rjeđe ponavljaju.
2) Simboli sa najmanjom frekvencijom sadržavat će istu veličinu koda [4, str. 41].

III. Opis algoritma

Tehnika Huffmanovog algoritma zasniva se na nadovezivanju kodova sa kraćim bitima onim dijelovima podataka (piksela) sa višom frekvencijom. Pikseli koji se rjeđe prikazuju na slici dobijaju kodove sa dužim bitima jer će i njihovo pojavljivanje u kompresovanoj slici biti znatno manje od piksela koji će se zamijeniti sa kraćim bitima. Izumljena je *1951.* godine od MIT naučnika David Huffman-a i od tog trenutka smatra se jednom od najefikasnijih i optimalnijih metoda kompresije. Koristi se radi jednostavnosti, velike brzine i generalno široke primjene u ostalim algoritmima [12]. Pseudokod 1 opisuje implementaciju generalnog algoritma korištenog u kompresiji procesom učitavanja slike u formi matrice i kodiranjem elemenata matrice zavisno od frekvencije. Za strukturu podataka standardni algoritam koristi binarno stablo koje omogućuje rekurzivno kreiranje kodova od najdužih prema najkraćim na sljedeći način:

(a) Prvobitno preuzme vrijednosti piksela iz slike i grupiše po frekvenciji njihovog pojavljivanja.
(b) Vrijednosti se ubacuju u stablo od simbola sa najvećom frekvencijom prema manjom i zatim se grupišu u parove.
(c) Svaka dva simbola (u stablu listovi koji nisu povezani) nadovezuju se za roditelja koji sadrži sumu njihove vjerovatnoće pojavljivanja. Algoritam nastavlja dodavati i povezivati listove dok ne prođe kroz simbole u slici.
(d) Nakon formiranja stabla njegove grane se označe sa 0 (lijeva grana) ili 1 (desna grana). Prolaz kroz korijen stabla formira kodove uzimajući u obzir vrijednosti grana i dodavanja tih vrijednosti na kod dok ne pronađe traženi simbol u stablu.

A. Primjer

Neka je data jednostavna slika sa simbolima iz alfabeta *A = {s1, s2, s3, s4, s5}* gdje *P(s1)=P(s2)=0.2, P(s3)=0.6, P(s4) = 0.1 i P(s5) = 0.1*. Kreiranje Huffmanovog koda zasniva se na sortiranju simbola u opadajućem poretku po vjerovatnoći pojavljivanja. Sa sortiranim podacima pridružujemo kodove *0* i *1* na simbole sa najmanjom frekvencijom pojavljivanja *s4* i *s5*. Prvi korak proizvodi

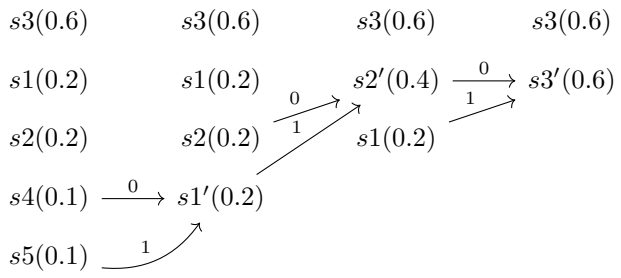
$$c(s4) = \alpha 1 * 0 \quad i \quad c(s5) = \alpha 1 * 1,$$

gdje $\alpha 1$ predstavlja binarni string nad kojem će se kroz korake nadovezivati biti. Sada je potrebno grupisati vjerovatnoću kodova iz alfabeta i vratiti natrag u listu simbola, ali sada kao grupisan simbol *s1'*. Ovaj postupak nastavljamo dok ne završimo sa 2 simbola unutar liste gdje u svakom koraku

Pseudokod 1 Algoritam Huffmanovog kodiranja

```
procedure HUFFMANCODING(a)
    slika ←učitavanje slike
    S ←VRATISIMBOLE(slika)
    F ←VRATIFREKVENCIJU(sim)
    sortiranje F i S u opadajućem redoslijedu
    Stablo ←{}
    while F.length > 2 do   ▷ Oznaci S[i] odgovara F[i]
        listA ←(S[0],F[0])
        listB ←(S[1],F[1])
        Z ←listA i listB, Z sadrži sumu listova
        F →F.pop() F →F.pop()
        S →S.pop() S →S.pop()
        F ←Z->suma
        Stablo ←UBACICVOR(Z)
        sortiranje F i S u opadajućem redoslijedu
    end while
    return Stablo              ▷ LijevaG ←0, DesnaG ←1
end procedure
```

dodajemo bite *0* i *1* na simbole sa najmanjom frekvencijom. Očekuje se da nakon konstruisanja Huffmanovog koda dobijamo prosječno kraću dužinu za kodove i manji broj iskorištenih bita u finalnoj kompresovanoj verziji [4, str. 45]. Kroz svaki korak dodajemo dodatne bite nad kodovima koji se



Slika 4: Postupak Huffmanovog kodiranja

manje pojavljuju na slici. Na Slici 4 primijetimo da simbol *s3* koji se najčešće pojavljuje na slici dobija i kod sa najmanjom veličinom jer se ne unosi u stablo do samog kraja.

Simbol	Vjerovatnoća	Kod
s3	0.6	0
s1	0.2	01
s2	0.2	000
s4	0.1	0010
s5	0.1	0011

Slika 5: Kodovi nakon kodiranja pomoću Huffmana

Dobijene kodove na Slici 5 sada je moguće iskoristiti na originalnoj slici zamijenom starih simbola i njihovih vrijednosti sa novim. Kod *s3* sa najvećom vjerovatnoćom pojavljivanja sada zauzima 1 bit što može biti ogromno poboljšanje ukoliko je prije zauzimao više [4, str. 46].

IV. Rezultati

U ovoj sekciji primjenjuje se kompresija slika pomoću Huffmanovog kodiranja i ostalih popularnih tehnika poput Deflate kompresije, Burrows-Wheeler transformacije i LZMA (Lemper-Ziv-Markov lanac algoritam) algoritma. Pošto su ovo algoritmi koji se koriste za standardnu kompresiju podataka na današnjim uređajima očekuje se da će i rezultati kompresije biti bolji od jednostavnijeg Huffmanovog kodiranja.

(a) Lena

(b) Paprike

(c) Bumbar

(d) Mandril

Slika 6: Primjeri slika za testiranje

Pored slika na Slici 6 koristit će se dodatne slike sa različitim rezolucijama i bojama prilikom testiranja. Slike u boji su *png* formata, dok slike sivih tonova sadrže *tiff* i *jpg* format. Huffmanovo kodiranje je jednostavna metoda od koje nekad nije moguće očekivati najbolju kompresiju. Za bolji prikaz mogućnosti kompresija za testiranje i analizu koristit će se dodatne tehnike:

1) DEFLATE
2) Burrow-Wheeler
3) LZMA

Ove metode su poznate kao jednostavne kompresije bez gubitka. Kod takve metode prilikom rekonstrukcije slike očekuje se identična slika kao i originalna. U drugom slučaju, rekonstruisane slike će imati vidljive razlike, ali njihova kompresija će biti efektivnija, tj. kompresija će dati bolje rezultate u vidu veličine. Sa metodama koje će se testirati u ovom radu ne očekuje se znatna promjena u veličini, ali nadogradnja nad ovim algoritmima može daljnje pokazati funkcionalnost modernijih tehnika. Na Slici 6 prikazane su četiri slike u boji korištene za testiranje. Prateći rezultate kompresija u Tabeli I ove slike nisu dale najbolje rezultate nakon njihove kompresije. Slike poput *Einstein* i *Jetplane* vide umanjenje za 10% od njihove originalne veličine, a takva promjena je moguća jer su slike u crno-bijeloj boji. Metoda Huffmanovog kodiranja ili RLE algoritam grupiše piksele iste boje i zavisno od frekvencije takvih piksela uspijeva

dodatno da umanji veličinu finalne slike. Za slike u boji takva situacija je znatno teža jer svaki piksel može da prima malo drugačiju vrijednosti iako su pikseli susjedni što daljnje onemogučuje dobru efikasnost primjenjenih algoritama [13]. U Tabeli I rezultati kompresije su veličine slika prikazane u bajtu (B). Prethodne metode su kompresije bez gubitka i

Slika	Source	Huff C	Deflate	B-W	LZMA
Bumbar*	415026	998126	415162	416275	415088
Einstein	65803	62216	52109	45528	44844
Jetplane	525123	-	219327	152435	170588
Kamen*	669134	1041254	669345	667314	669208
Kmart*	373345	1004911	373466	374702	373408
Lake	525123	-	262817	184629	206508
Dnevna	262767	256603	219870	187980	181824
Mandril*	103167	158924	103208	103822	103216
Paprike*	726523	3009416	726754	729605	726600
Patka*	645380	1066096	645586	645516	645456
Most	525123	-	278409	209450	222476
WBlonde	262767	259241	212685	180714	179656
WDark	262767	257368	195639	140367	153796

* Slike u boji, - Slike nisu u prikladnom formatu

Tabela I: Tabela poređenja (bajt) raznih metoda kompresija bez gubitka

po rezultatima iz Tabele I uočljivo je da efikasnost takvih kompresija ovisi od vrijednosti samih piksela. Slike sivih tonova su imale puno bolje rezultate jer kroz analizu memorije kompresovanih slika dobija se prosječno umanjenje 2.8% kod Huffmanovovog kodiranja, 41.1% za Deflate, 35.7% B-W i 41% za LZMA algoritam. Slike u boji za algoritme Deflate, B-W i LZMA su umanjenje za 0.01% što je veoma neefikasno poboljšanje kad se poredi sa kompresijom sivoskaliranih slika. Dodatno se vidi da slike u formatu *png* nemaju vidljivu promjenu u kompresiji i u mnogim situacijama veličina se i povečava. Pošto slike *png* formata su slike bez gubitka procesor mora sliku prebaciti u drugi format, pokušati umanjiti i zatim ponovo vratiti u originalni format što dovodi do dodatnih nepravilnosti. Implementirani Huffmanov algoritam imao je velikih problema prilikom kompresije slika *png* formata pošto se finalna veličina drastično uvečala. U poređenju sa stabilnijim i efikasnijim algoritmima davao je mala poboljšanja i slabije umanjenje kompresovane slike što u mnogim slučajevima ne bi bilo korisno. Jednostavniji algoritmi, poput Huffmanovog, koriste se kao modifikacije u mnogim ostalim algoritmima i kontribucija ovakvih algoritama je daleko veća od njihovog rezultata [14]. Dodatne dorade na algoritmu Huffmanovog kodiranja poboljšale bi finalne rezultate ali i tad se ne bi mogao porediti sa efikasnijim algoritmima korištenim danas. Ostali formati slika koji su pogodni za kompresiju vidjeli su umanjenje njihove originalne veličine. Kompresija *png* slika nije dala istaknute rezultate niti za jednu metodu kompresije. U većini slučajeva veličina slike je ostala ista i sva 3 algoritma pored Huffmana su davala slične rezultate. Algoritam LZMA dao je poboljšanje od 74 bajta na slici *Bumbar* u poretku sa drugim najboljim algoritmom, dok je Burrow-Wheeler imao uspješnije rezultate sa 1894 bajta razlike na slici *Kamen*. Deflate algoritam, koji koristi kombinaciju algoritma Huffmanovog kodiranja i dodatnog LZSS algoritma [15], nije se pokazao najboljim prilikom kompresija testnih slika. Sa malim testnim podacima, nije moguče garantovati preciznost i efikasnost ova 4 algoritma pošto postoji mnogo faktora pored formata ili tipa slike koji mogu uticati na finalnu kompresovanu veličinu.

Slika	Source (B)	Kompresija (B)	CR
5.1.13	65670	36960	1.77
7.2.01	1048710	812574	1.29
5.3.01	1048710	930306	1.27
5.3.02	1048710	974935	1.07
7.1.07	262278	243533	1.07
7.1.01	262278	245958	1.06
7.1.03	262278	252818	1.03

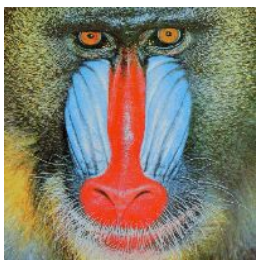
Tabela II: Kompresija *tiff* slika uz Huffmanovo kodiranje

Tabela II prikazuje rezultate kompresije korištenjem implementiranog Huffmanovog kodiranja nad slikama iz datoteke **"slike/new"**. Pošto su ovo slike nad kojim se može vršiti kompresija rezultati su vidljivo bolji. Za efikasnije kompresije žrtvujemo kvalitetu slike izbacivanjem podataka koje su manje uočljive ljudskom oku.

Slika	Source (B)	Kompresija (B)	CR
Bumbar	415009	18928	21.92
Kamen	669117	67081	9.970
Mandril	103150	9136	11.29
Paprike	726506	41226	17.62
Patka	645363	58390	11.05
Pexel	170335	135914	1.250
Kmart	373328	18236	20.47

Tabela III: Kompresija sa gubitkom (PIL), kvaliteta 50%

(a) 10% kvalitete 2.98kB



(b) 80% kvalitete 15.4kB

Slika 7: Primjer kompresije sa gubitkom

Kompresija u Tabeli III daje znatno bolje rezultate ali sa gubitkom kvalitete od 50%. Stepen kompresije (CR) objašnjava poboljšanje gledajući u obzir veličinu originalne i kompresovane slike. Pošto je vrijednost iznad 1.0 dobijamo informaciju da je kompresija bila djelotvorna. Posmatrajući sliku *bumbara* i *paprike* primijetimo ogromna poboljšanja jer su te slike same po sebi veoma jednostavne i ne postoji velika razlika između boja piksela. Slika *pexel* ima prilično lošu kompresiju što nije neobično posmatrajući detalje same

slike. Kompresije sa gubitkom izbacuju podatke prilikom procesa umanjenja slike. Mandril na Slici 7 pokazuje veliku razliku u veličini nakon drastične kompresije koja kroz proces umanjuje kvalitetu na 10%. Ovakva kompresija daje odlično očuvanje memorije uređaja prilikom arhiviranja ili transmisije preko interneta, ali pogoršanje kvalitete je očito. Kroz analizu kompresija, situacije u kojim se može tolerisati gubitak kvalitete slika dobijaju znatno bolja rješenja i umanjenja ukupne memorije za razliku od kompresija bez gubitka. Sa razvojem tehnologije i transmisije podataka, korisnici očekuju bolju kvalitetu i brže slanje podataka preko interneta što čini kompresiju bez gubitka izuzetno bitnom i u svakodnevnom životu.

V. Aplikacija

Huffmanovo kodiranje je jednostavna tehnika kompresije slika koja sama po sebi ne daje jake rezultate. Algoritam opisan u ovom radu preuzima vrijednosti piksela iz niza i za svaki simbol dodaje novu kodnu riječ. Ovo nije najoptimalniji način i sama implementacija Huffmanovog koda ima mjesta za poboljšanje. Slike *png* formata, iako nisu pogodne za kompresiju, uvećale su se nakon kompresije Huffmanovog algoritma što sa boljim načinom čitanja piksela slike ne bi bio slučaj. Za budući rad tehnika Huffmanovog kodiranja može se primijeniti na ostale tipove podataka poput zvuka, animacija, teksta i videa. Skup podataka nad kojem su testirane kompresije je limitovan i sa većim skupom novi detalji bi se jasnije uočili. Finalno, kombinacija algoritama kompresije sa tehnikama mašinskog učenja i neuronskih mreža bi dovelo do bržeg treniranja modela i eventualno efikasnije kompresije sa dobro podešenim parametrima.

VI. Zaključak

U ovom radu ukratko su opisane glavne metode kompresija slika kao i njihove razlike u efikasnosti. Kompresija bez gubitka pokazala se da ima bolju primjenu ukoliko je informacije potrebno zadržati uz manju promjenu u veličini. Područja medicine, fotografije i procesovanja teksta zahtjevaju potpunu kvalitetu tako da kompresija sa gubitkom ne bi bila pogodna. U drugom slučaju, kompresije sa gubitkom su važnije za transmisiju podataka na internetu i arhiviranja podataka na bazama koje su ograničene u memoriji, tako da je njihova primjena i puno opširnija. Kompresija slika je od kritične važnosti u procesiranju digitalnih slika sa dodatnim porastom korištenja tehnologije prouzrokovane od pandemije i razvoja tehnologije. U digitalnom procesovanju slika postoje razne tradicionalne i savremene tehnologije koje omogućavaju kompresiju. U ovom radu su opisane neke od najpopularnijih i optimalnijih metoda koje su omogučile nastanak i razvoj tehnika kompresije koje korisnici koriste danas.

Reference

[1] R. K. Kurmi and S. Gupta, *A Review of Lossless and Lossy Based Image Compression Techniques*, Aug 2017.

[2] R. Patel, V. Kumar, V. Tyagi, and V. Asthana, "A fast and improved image compression technique using huffman coding," in *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*. IEEE, 2016, pp. 2283–2286.

[3] H. Samra, "Image compression techniques," *INTERNATIONAL JOURNAL OF COMPUTERS TECHNOLOGY*, vol. 2, pp. 49–52, 04 2003.

[4] K. Sayood, *Introduction to data compression*. Mogan Kaufmann, 2018.

[5] G. Patidar, S. Kumar, and D. Kumar, "A review on medical image data compression techniques," in *2nd International Conference on Data, Engineering and Applications (IDEA)*, 2020, pp. 1–6.

[6] V. V. Lukin, M. S. Zriakhov, N. N. Ponomarenko, S. S. Krivenko, and M. Zhenjiang, "Lossy compression of images without visible distortions and its application," in *IEEE 10th INTERNATIONAL CONFERENCE ON SIGNAL PROCESSING PROCEEDINGS*, 2010, pp. 698–701.

[7] G. Scarmana, "Lossless data compression of grid-based digital elevation models: A png image format evaluation," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. II-5, 05 2014.

[8] F. Alfiah, A. Setiadi, Saepudin, A. Supriadi, and I. Maulana, "Discrete cosine transform dct methods on compression rgb and grayscale image," 09 2020.

[9] A. Raid, W. Khedr, M. El-dosuky, and W. Ahmed, "Jpeg image compression using discrete cosine transform - a survey," *International Journal of Computer Science Engineering Survey*, vol. 5, 05 2014.

[10] K. Cabeen and P. Gent, *Image compression and the Discrete Cosine Transform*, p. 1–11, 2014.

[11] R. A. Hasan, "Combination of lossy and lossless for image compression," Nov 2014.

[12] A. J. Qasim, R. Din, and F. Qasim Ahmed Alyousuf, "Review on techniques and file formats of image compression," *Bulletin of Electrical Engineering and Informatics*, vol. 9, no. 2, 2020.

[13] K. Lina J., "Chapter 16 - lossless image compression," in *The Essential Guide to Image Processing*, A. Bovik, Ed. Boston: Academic Press, 2009, pp. 385–419. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780123744579000160

[14] M. Rahman, "Burrows–wheeler transform based lossless text compression using keys and huffman coding," *Symmetry*, vol. 12, 10 2020.

[15] N. Dhawale, "Implementation of huffman algorithm and study for optimization," in *2014 International Conference on Advances in Communication and Computing Technologies (ICACACT 2014)*, 2014, pp. 1–6.