VPLIV SLIKOVNE KOMPRESIJE NA STROJNO PREPOZNAVANJE RAZPOK NA CESTAH

Tomi Božak

Povzetek

Prispevek se osredotoča na vpliv slikovne kompresije na strojno prepoznavanje razpok na cestah. Zaznavanje objektov je vseprisotno, zato je dobro razumeti, kako kompresija vpliva na učinkovitost tega procesa. Raziskava preučuje, do kakšne mere je mogoče komprimirati sliko, da model za prepoznavanje razpok še vedno deluje zadovoljivo. Osredotočamo se na vprašanje, kako močno lahko zmanjšamo velikost slik, hkrati pa ohranimo zadovoljivo delovanje modela za prepoznavanje razpok. Uporabljamo dva pristopa za kompresijo: singularni razcep in privzeto tehniko kompresije za format JPEG. Rezultate analiziramo s štirimi ključnimi metrikami: presek/unija, natančnost, priklic ter kvadratna napaka.

Rezultati raziskave razkrivajo zanimive ugotovitve. Opažamo, da se metrike spreminjajo sorazmerno z močjo kompresije. Posebej opazno je, da kvadratna napaka ohranja natančnost slikovnih pik, vendar ta natančnost ne prehaja na ohranjanje natančnosti pri prepoznavanju objektov. Primerjava med uporabljenima kompresorjema pokaže, da je privzeta kompresija za JPEG učinkovitejša pri ohranjanju kakovosti prepoznavanja razpok. Sklepi kažejo, da je kljub vizualni podobnosti med komprimirano in originalno sliko potrebno upoštevati razhajanja med subjektivnimi ocenami in metrikami. S to raziskavo prispevamo k razumevanju, kako kompresija vpliva na delovanje strojnega prepoznavanja razpok in odpiramo vrata nadaljnjim raziskavam za izboljšanje metod in meril učinkovitosti.

1 Uvod

Zaznavanje objektov (ang. *object detection*) je v zadnjem času skorajda vseprisotno. Razvoj modelov je šel celo tako daleč, da imajo novejše mobilne naprave zaznavanje objektov že vgrajeno v svojo primarno aplikacijo za pregled fotografij. Zaznavanje objektov se uporablja tudi pri zaznavanju obrazov, prepoznavi obrazov, spremljanju vozil in pešcev na cestah ter v križiščih ipd. Med drugim lahko modele strojnega učenja s tega področja uporabimo tudi za zaznavo razpok na voznih površinah, česar se lotimo v tem delu. Pri tem se bomo naslonili na že razvite modele za detekcijo razpok [3], ki temeljijo na globokem učenju, zanimalo pa nas bo, kako bi postopek zaznavanja objektov bilo možno optimizirati z vidika pomnilniškega prostora. Natančneje, ugotovili bomo, kako zelo smemo zmanjšati velikost slik – jih stisniti (komprimirati), da je zaznavanje objektov še vedno dovolj uspešno. Problem so v splošnem nekajkrat že obravnavali [1, 2], vendar je naše podatkovje (slike cestišča) zelo specifično in se morda obnaša drugače, kot npr. podatkovje COCO [6], ki je eno bolj znanih podatkovij za

preizkušanje modelov za detekcijo objektov in vsebuje razne slike (ime COCO je okrajšava polnega imena Common Objects in Context). Zato smo se odločili, da se osredotočimo na ta ožji problem, in sicer, kako bi postopek bilo mogoče izboljšati za specifične potrebe zaznavanja razpok. Analizirali smo, kako zelo lahko komprimiramo slike, tj. zmanjšamo velikost slike (kot datoteke), da model za prepoznavanje razpok na cestah še vedno deluje relativno dobro. S tem bi lahko delo na tem področju naredili učinkovitejše, saj bi lahko avtomatsko zaznavali, kje so najkritičnejši deli cest, prav tako pa bi potencialno ogromna količina slik zasedla precej manj prostora, kot ga zasedejo izvorne.

2 Metode dela in potek raziskave

Za raziskavo je bilo treba najti dober model za prepoznavo razpok na slikah ter napisati dva programa za kompresijo slik. Model smo našli na spletu [3] ter ga prilagodili [4], programa pa napisali sami v Pythonu (najdemo ju v našem repozitoriju [4] v datotekah *./executable/pcaColor_kompresor.py* ter *./executable/pil_kompresor.py*).

2.1 Zaznavanje razpok

Z razvojem računalniških procesorjev so (globoke) nevronske mreže v zadnjih letih doživele razmah in postavile nove standarde pri analizi slik [11]. Opis nevronskih mrež je izven obsega tega dela, zato omenimo le, da praktično vse temeljijo na konvolucijskih slojih [9]. Na začetnem, vhodnem sloju, model sprejme sliko, kot jo vidimo ljudje, nato pa sliko pošlje skozi več skritih slojev, pri čemer jo predela v človeku vedno bolj abstraktno, a za računalnik precej bolj uporabno obliko, s pomočjo katere na izhodem sloju naredi napoved. Naš model [3] temelji na VGG16 [7] in U-Net [8], ki sta med bolj znanimi arhitekturami. Najprej je bil naučen za splošno detekcijo objektov, nato pa specializiran za razpoke. Model je namenjen segmentaciji slik in pove, kje vse na sliki je razpoka (glej razdelka Delotok in Rezultati).

2.2 Pristopa za stiskanje slik

Prvotna zamisel za komprimiranje je bil singularni razcep (ang. *singular value decomposition oz. SVD*). To je matematična metoda, ki poljubno kompleksno matriko $A \in \mathbb{C}^{m\times n}$ razcepi na produkt matrk U, Σ in V , tako da velja $A = U\Sigma V^*$. Stolpcem matrik $U \in \mathbb{C}^{m\times m}$ in $V \in \mathbb{C}^{n\times n}$ (ki sta ortogonalni) pravimo (levi oz. desni) singularni vektorji in jih označimo z u_i ter v_i. Matrika $\Sigma \in \mathbb{R}^{m\times n}$ je diagonalna in ima na diagonali po velikosti padajoče urejene nenegativne singularne vrednosti σ_i matrike A. Če zgornjo matrično enakost zapišemo kot $A = \sum_1^r \sigma_i\, u_i\, v_i^*$, kjer je r rang matrike *A,* lahko hitro dobimo tudi najboljši približek za matriko A z matriko nižjega ranga $r' < r$ (pri čemer kvaliteto približka merimo npr. v drugi operatorski normi). Vse, kar moramo storiti, je, da gremo z vsoto le do *r'.* Z zmanjševanjem ranga zmanjšujemo kvaliteto približka, hkrati pa tudi njegovo velikost na disku. Barvno sliko lahko opišemo s tremi matrikami – po eno za vsako komponento RGB (ang. *red-green-blue*). Zato lahko na vsaki od teh matrik izvedemo singularni razcep, ohranimo le prvih *r'* singularnih vrednosti in s tem

zmanjšamo dimenzije matrik. Cilj je, da se z zmanjševanjem dimenzije matrik, zmanjšuje tudi bitna velikost slike. Singularni razcep je zelo tesno povezan z metodo imenovano metoda glavnih osi (ang. *principal component analysis oz. PCA*). Na slednji temelji algoritem v programu, ki smo ga uporabili za raziskavo, tj. (kompresor) PCA.

Drugi program [5] (imenujemo ga PIL) uporablja knjižnico Pillow, ki poskrbi za stiskanje slike. Obravnavane slike so bile kodirane v formatu JPG, zato funkcija *image.save(dest_path, quality)* iz knjižnice Pillow uporablja privzeti algoritem stiskanja za JPG/JPEG. Ta sliko stisne v več korakih, pri čemer uporabi diskretno kosinusno transformacijo in kvantizacijo (za znižanje prisotnosti visokih frekvenc, kar naše vidne zaznave ne zmoti). Tu se nekaj informacij z začetne slike izgubi, na koncu pa se trenutno predelano sliko dodatno stisne (brez izgube inforamcij) s Huffmannovim kodiranjem [5, 10]. Postopek je odvisen od vrednosti parametra kakovosti (*quality*), ki vpliva na stopnjo kvantizacije: višje vrednosti ohranjajo več podrobnosti in izhodna datoteka je večja.

2.3 Kaj in kako smo raziskovali?

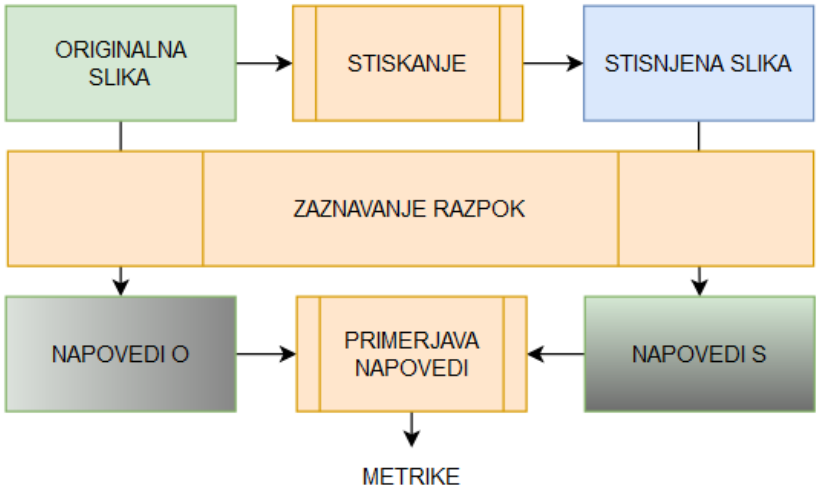
Preverili smo, kako stopnja kompresije vpliva na rezultate modela za prepoznavo razpok na (*velikih*) fotografijah cest. Zadali smo si odgovoriti na vprašanje, za koliko odstotkov smemo zmanjšati velikost fotografij, da je uspešnost modela še vedno relativno visoka. Uspešnost modela smo ugotavljali na podlagi štirih različnih metrik: presek/unija, natančnost, priklic ter kvadratna napaka. Vse metrike se računajo na črno-belih slikah, ki so izhod modela za prepoznavanje razpok. Če izhod modela na originalni sliki označimo z $O \in [0, 1]^{m \times n}$ (kjer vrednost O_{ij} interpretiramo kot verjetnost, da je na pikslu (i, j) razpoka) in s $S \in [0, 1]^{m \times n}$ označimo izhod modela na stisnjeni sliki, potem so omenjene metrike definirane kot spodaj:

- Presek/unija (ang. *Intersection over Union* oz. *IoU*) $|O \cap S|/|O \cup S|$ izračuna razmerje med velikostjo preseka dveh množic/slik (pravih in napovedanih regij) ter velikostjo njune unije. Pri tem presek oz. unijo računamo po komponentah (pikslih) kot minimum oz. maksimum pripadajočih vrednosti, velikost pa kot vsoto vrednosti v matriki (pri tem je pomembno, da so vrednosti na intervalu [0, 1]). Višja vrednost pomeni boljšo usklajenost med pravimi (originalna slika) in napovedanimi (stisnjena slika) regijami.
- Natančnost (ang. *Precision*) $|O \cap S|/|S|$ izračuna razmerje med pravilno napovedanimi pozitivnimi primeri (razpoke, ki so bile pravilno prepoznane) in vsemi napovedanimi pozitivnimi primeri. Višja natančnost pomeni manj lažnih pozitivnih napovedi.
- Priklic (ang. *Recall*) $|O \cap S|/|O|$ izračuna razmerje med pravilno napovedanimi pozitivnimi primeri in vsemi pravilnimi pozitivnimi primeri (razpoke v izvirnih slikah). Višji, kot je priklic, manj dejanskih razpok smo zgrešili.
- Kvadratna napaka (ang. *Mean Squared Error,* MSE) $||O - S||_2^2 / (mn)$ meri povprečno kvadratno razliko med dejanskimi in napovedanimi vrednostmi. Manjša vrednost pomeni boljše prileganje napovedi dejanskim vrednostim.

Vrednosti, ki jih poročamo, so povprečja zgornjih metrik čez 500 slik, ki smo jih analizirali. Algoritmi za računanje vseh štirih metrik so dostopni v našem direktoriju [4], v datoteki *metrics.py*.

2.4 Delotok

Naš delotok predstavlja diagram na desni (obdržali smo oznaki O in S iz prejšnjega razdelka):



Program deluje v štirih glavnih korakih. Najprej se požene model za prepoznavo razpok na izvornih slikah, pri čemer se rezultati shranijo kot referenčni podatki in se privzamejo za *resnične vrednosti*. S tem dobimo napovedi *O*. Vzorčne podatke namreč predstavlja 500 slik, ki bi jih bilo predolgotrajno označevati na roke (s človeškim posredovanjem računalniku posredovati, kje na sliki so razpoke). Originalno sliko nato stisnemo in shranimo na disk. Nato še njo pošljemo skozi model za razpoznavanje razpok, s čimer dobimo napovedi *S*. Na zadnjem koraku napovedi primerjamo napovedi *O* in *S* ter izračunamo vrednosti metrik za dani par slik. Vse izračunane metrike se shranijo v obliki CSV datoteke za nadaljnjo analizo. Posebej izračunamo še razmerje velikosti stisnjene in originalne slike, česar pa na diagramu zaradi preglednosti ni.

Programa za stikanje PCA in PIL smo implementirali sami, pri čemer smo se naslonili na Pythonovi knjižici scikit-learn in Pillow. Kodo za poganjanje modela za zaznavanje razpok [3] smo nato prilagodili svojim potrebam, saj (med drugim) nismo imeli na voljo grafične kartice (ki bi pohitrila napovedi). Vse skupaj povezuje glavna skripta *pozeni.py* (najdemo jo v našem direktoriju [4], pod *./executable/pozeni.py*). Pognali smo jo na 500 slikah cest in s tem pridobili podatke, ključne za našo raziskavo. Nazadnje smo podatke o slikah, kompresijah in metrikah analizirali in napisali strnjen povzetek v razdelku Rezultati. Za lažje poganjanje programa na različnih računalnikih smo ustvarili tudi pripadajočo Dockerjevo sliko. S tem tudi dosežemo daljšo življenjsko dobo napisane kode.

Za obdelavo 500 slik z ločljivostjo 4000×3000 slikovnih pik program *pozeni.py* porabi približno 40 ur na 64-bitnem računalniku z Intelovim procesorjem (12th Gen Intel(R) Core(TM) i7-12700, 2100 MHz, 12 jeder, 20 logičnih procesorjev) in 32 GB delovnega pomnilnika. Računalnik poganja Windows 11 Pro.

2.5 Obravnavane stopnje stiskanja

Na danem računalniku smo za analizo ene skupine 500 slik (izvornih ali stisnjenih) potrebovali nekaj ur, zato smo bili omejeni pri številu različnih vrednosti parametrov kakovosti in algoritmov, ki jih lahko preizkusimo.

Pri pristopu stiskanja PIL smo parameter kakovosti *q* nastavili na 10, 25, 50, 75, 90, 95 in 99. Pri PCA smo rang približka *r'* nastavili na 1, 2, 4, 8, 16, 32, 64, 128 in 256. Pričakovati je namreč, da je prvih nekaj singularnih vrednosti velikih, nato pa te postanejo čedalje manjše.

3 Rezultati

V tem razdelku predstavimo rezultate stiskanj. Začnemo s PIL-om, nadaljujemo s PCA-jem, na koncu pa sledi še njihova analiza.

3.1 PIL

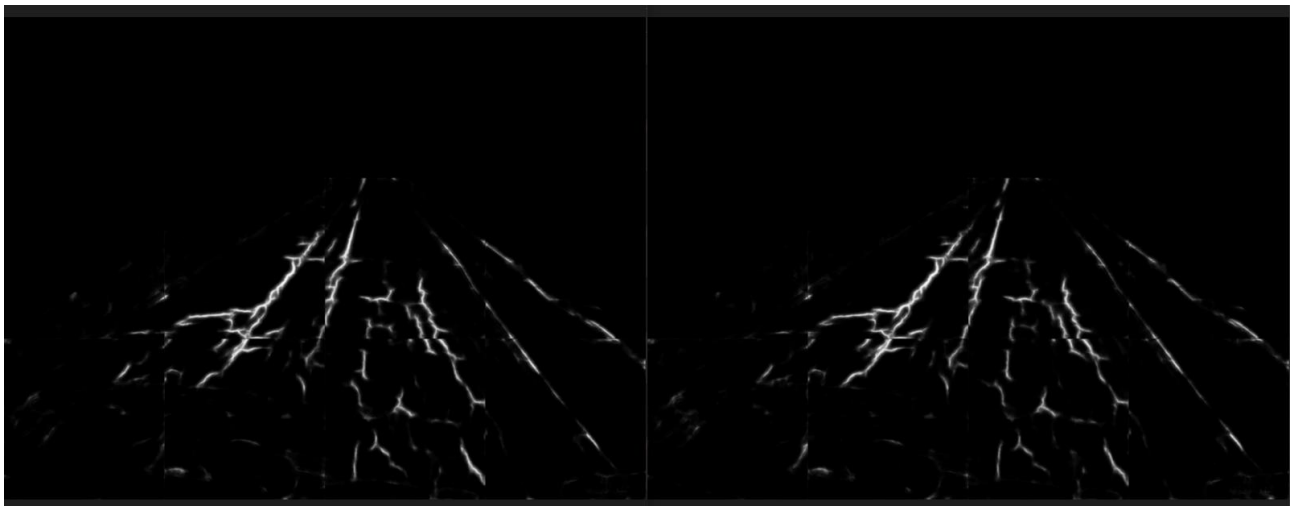
Z zmanjševanjem vrednosti parametra kakovosti se zmanjšuje velikost slik. Višja stopnja kompresije povzroča upad vrednosti metrik presek/unija, natančnost in priklic, kar nakazuje na izgubo natančnosti pri prepoznavi objektov. Vrednosti kvadratne napake vseskozi ostajajo nizke (v absolutnem smislu), kar kaže na ohranjanje natančnosti na ravni celotne slike. Rezultati modela komprimiranih slik so vizualno podobni rezultatom izvirnikov.

3.2 PCA

Manjše število singularnih vrednosti (*r'*) vodi do višje kompresije, vendar je že pri *r'*=64 povprečna velikost stisnjene slike večja od izvirnika. Zaradi tega se bomo pri opazovanju omejili na situacije, kjer je *r'*<64. Višja stopnja kompresije povzroča upad vrednosti metrik presek/unija, natančnost in priklic, kar nakazuje na izgubo natančnosti pri prepoznavi objektov. Vrednosti teh treh metrik so precej nizke, kar nakazuje na to, da je prepoznava objektov po kompresiji s PCA slaba. Kvadratna napaka tudi tukaj vseskozi ostaja nizka.
Napovedi (kot slike) so do vključno *r'*=64 večinoma črne. To pomeni, da model ni zaznal prav nobenih razpok (kljub temu, da so na slikah velike razpoke).

3.3 Vizualna primerjava rezultatov

Rezultate smo primerjali tudi »na oko«. Zanimivo je, da človek med rezultati izvirnikov ter rezultati srednje kompresiranih (ko je q med 25 in 50) slik s kompresorjem PIL na oko ne opazi bistvene razlike (Slika 1). To nam da dober uvid v to, katere vrednosti metrik, kot sta npr. priklic in natančnost še predstavljajo povsem zadovoljive rezultate.



Slika 1: Leva polovica prikazuje izhod modela pri vhodu izvorne slike, desna polovica pa rezultat pri vhodu iste slike, komprimirane s PIL z vrednostjo parametra quality=50.

3.4 Analiza rezultatov

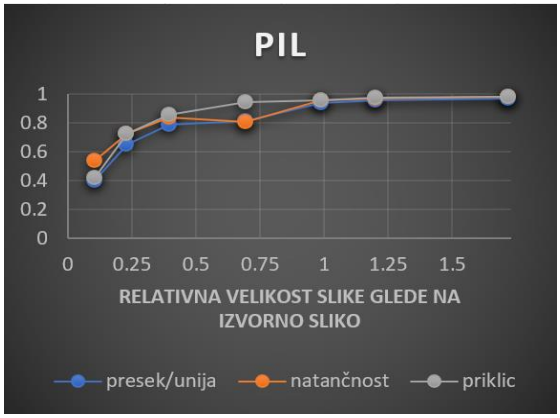
Diagrami na spodnjih grafikonih prikazujejo vrednosti izračunanih metrik v odvisnosti od razmerja med velikostjo slike, nad katero je model izvedel predikcijo, in izvorno sliko. Ker se točne vrednosti teh razmerij nekoliko razlikujejo od slike do slike (tj. pri isti vrednosti parametra q pri različnih slikah dobimo različna razmerja), je na diagramih prikazana vrednost metrike v odvisnosti od *povprečne* vrednost tega razmerja. Kot je razvidno z diagramov, so nekatere slike, obdelane s PIL oz. PCA, večje od izvornih. To je najverjetneje posledica shranjevanja, oz. kodiranja slik s formatom JPG – ta namreč pred shranjevanjem do neke mere že sam stisne sliko. To stori celo tako dobro, da se sliko velikosti 3000 x 4000 že pri zelo nizkih vrednostih ranga r' v PCA bolj splača hraniti v JPG formatu kot pa seznam r' trojic (σ_i, u_i, v_i), čeprav eno tako trojico predstavimo s 7001 števili.

Opazimo lahko tudi, da se vrednosti metrik presek/unija, natančnost, priklic in kvadratna napaka v obeh primerih (PIL in PCA) pričakovano slabšajo (vrednosti prvih treh padajo, vrednosti kvadratne napake pa rastejo). Dejstvo, da so vrednosti kvadratne napake vseskozi (absolutno gledano) precej nizke nam razkrije tudi, da je večji del slike nezanimiv oz. so razpoke na njej na precej malo pikslih. Po analitični primerjavi rezultatov kompresorjev, smo ugotovili, da je algoritem v PIL za dan problem precej učinkovit, medtem ko je PCA tako rekoč neuporaben. Iz grafikonov 1 in 3 je dobro razvidno, da je PIL v sliki ohranil več pomembnih

podatkov za prepoznavo razpok – presek/unija, natančnost in priklic so namreč za poljubno stopnjo kompresije pri PIL precej višjih vrednosti kot pri PCA.

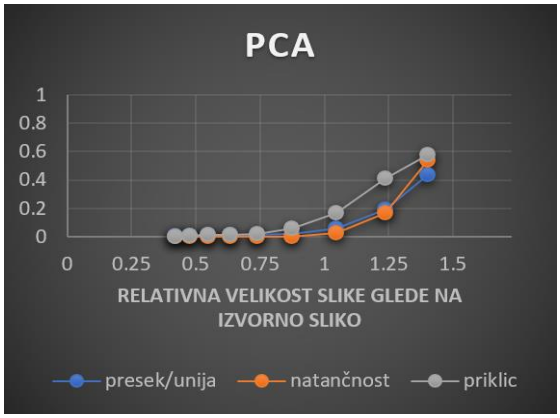
4 Zaključki

V raziskavi smo se osredotočili na učinkovitost kompresije slik, z izrazitim poudarkom na prepoznavanju razpok na komprimiranih slikah. Kljub očitni vizualni podobnosti med komprimirano in originalno sliko smo se soočili z neusklajenostjo med subjektivnimi ocenami ter rezultati meritev metrik. Ta razhajanja nas spodbujajo k razmisleku o primernosti uporabljenih meril.

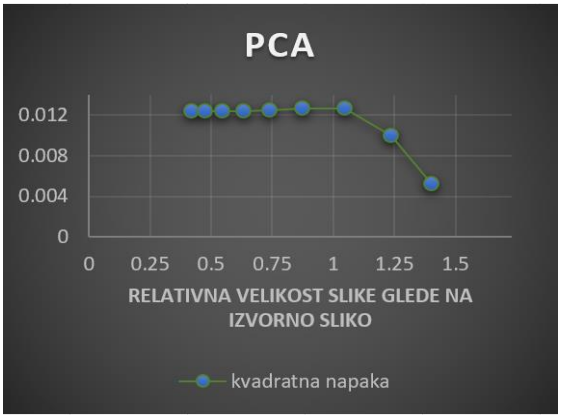


Grafikon 1: Vrednosti metrik v odvisnosti od razmerja med velikostjo izvorne slike ter velikostjo slike komprimirane s PIL.

Grafikon 2: Vrednosti metrike v odvisnosti od razmerja med velikostjo izvorne slike ter velikostjo slike komprimirane s PIL.



Grafikon 3: Vrednosti metrik v odvisnosti od razmerja med velikostjo izvorne slike ter velikostjo slike komprimirane s PCA.



Grafikon 4: Vrednosti metrike v odvisnosti od razmerja med velikostjo izvorne slike ter velikostjo slike komprimirane s PCA.

Iz rezultatov, pridobljenih z uporabljenimi metrikami, pa vendarle izluščimo nekaj bistvenih spoznanj: visok priklic nakazuje, da model na komprimiranih slikah še vedno učinkovito prepoznava večino razpok, vendar ta metrika ne razkriva lažno pozitivnih rezultatov. Nizka vrednost kvadratne napake nakazuje relativno natančnost ocenjevanja posameznih slikovnih pik, a ta natančnost se ne prenaša na raven celotnih objektov, kot so razpoke. Poleg tega naši

rezultati kažejo, da je z uporabo kompresorja PIL mogoče slike stisniti na približno 30%-50% prvotne velikosti, ob tem pa ohraniti sprejemljivo kakovost prepoznavanja razpok.

V prihodnjih raziskavah bi bilo smiselno preizkusiti različne formate slik (npr. PNG) ter jih komprimirati tudi s kakšno metodo za stiskanje brez izgub (ang. lossless compression) – v raziskavi sta namreč uporabljeni dve različni metodi za stiskanje z izgubami (ang. lossy compression). Poleg tega bi lahko komprimirali tudi z metodami strojnega učenja. Vzporedno s tem bi bilo koristno raziskati, kako kompresija vpliva na porabo energije in časa pri prepoznavanju, kar je ključno za praktično uporabnost. Nadalje predlagamo uporabo bolj občutljivih metrik za ocenjevanje prepoznavanja objektov, kot je npr. stopnja poškodovanosti vozišč, da bi zagotovili natančnejše rezultate. S pomočjo te rešitve bi se ocenjevanje uspešnosti na komprimiranih slikah predvidoma precej poenostavilo.

5 Zahvala

Zahvaljujem se predstavnikom Zavoda za gradbeništvo Slovenije za vzorčne podatke in strokovna mnenja pri nastajanju te naloge ter mentorju in Mateju Petkoviću za pomoč pri pisanju kode in prispevka.

6 Literatura

[1] J. M. Janeiro, S. Frolov, A. El-Nouby, J. Verbeek, Are Visual Recognition Models Robust to Image Compression?, 2023 Dostopno na: https://arxiv.org/abs/2304.04518

[2] Gandor, T.; Nalepa, J. First Gradually, Then Suddenly: Understanding the Impact of Image Compression on Object Detection Using Deep Learning. Sensors 2022, 22, 1104. Dostopno na: https://doi.org/10.3390/s22031104

[3] Model za prepoznavanje razpok: khanhha, crack_segmentation, uporabljeno avgusta 2023. Dostopno na: https://github.com/khanhha/crack_segmentation

[4] Koda za ta prispevek. Dostopno na: https://github.com/Space0code/kompresijaZaModel

[5] baeldung, JPEG Compression Explained. Dostopno na: https://www.baeldung.com/cs/jpeg-compression

[6] Tsung-Yi Lin, Maire, M., Belongie, S. J., et al. (2014). Microsoft COCO: Common Objects in Context. *CoRR*, *abs/1405.0312*. Retrieved from http://arxiv.org/abs/1405.0312

[7] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[8] Ronneberger, O., Fischer, P., Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In: Navab, N., Hornegger, J., Wells, W., Frangi, A. (eds) Medical Image Computing and Computer-Assisted Intervention. MICCAI 2015. Lecture Notes in Computer Science(), vol 9351. Springer, Cham. https://doi.org/10.1007/978-3-319-24574-4_28

[9] O'Shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*.

[10] D. A. Huffman, A Method for the Construction of Minimum-Redundancy Codes, in *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098-1101, Sept. 1952, doi: 10.1109/JRPROC.1952.273898.

[11] Zou, Z., Chen, K., Shi, Z., Guo, Y., & Ye, J. (2023). Object detection in 20 years: A survey. *Proceedings of the IEEE*.