SVEUČILIŠTE U ZAGREBU
**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

DIPLOMSKI RAD br. 2292

# Vrednovanje metoda rastresanja podataka za konvolucijske modele

Natko Kraševac

Zagreb, srpanj 2020.

*Umjesto ove stranice umetnite izvornik Vašeg rada.*
*Da bi ste uklonili ovu stranicu obrišite naredbu* `\izvornik`.

*Zahvaljujem se mentoru prof. dr. sc. Siniši Šegviću na pomoći tijekom studija te svojoj obitelji i prijateljima na pruženoj podršci tijekom cijelog školovanja.*

# SADRŽAJ

# 1. Uvod

Česti problem u području strojnog i dubokog učenja je dostupna količina označenih primjera za učenje modela. Kako bi model naučio bolje i time davao kvalitetnije predikcije, potreban je veliki skup podataka za učenje, a samo postupak označavanja uglavnom je dugotrajan zbog ručnog onačavanja, skup zbog radne snage potrebne da se posao obavi ili specijaliziran pa je stoga potreban stručnjak koji će označavati primjere. Kako bi se doskočilo ovome problemu, uz metode polunadziranog i nenadziranog učenja koje djelomice ili u potpunosti eliminiraju potrebu za označenim podacima, najčešće se koriste metode rastresanja (engl. *data augmentation*) dostupnih označenih podataka. Ovaj postupak podrazumijeva stvaranje novih promjera za učenje od već postojećih te pridodjeljivanje oznake novonastalim primjerima od izvornog označenog primjera. U području računalnog vida postoje razne metode rastresanja podataka. One uglavnom uključuju modificiranje originalne slike kako bi se generirala nova slika koja i dalje predstavlja stvarni mogući primjer iz realne distribucije skupa primjera. Korištenjem ovog postupka, bitno je paziti da novonastala slika bude dovoljo različita od originalne, ali i dovoljno slična kako se ne bi unio šum u skup podataka. Šum podrazumijeva slike koje je čak i čovjeku teško klasificirati ili je slika promijenjena u onoj mjeri da bi ju objektivni promatrač svrstao u potpuno drugi razred od onog kojim je slika označena.

U ovom radu cilj je implementirati nekoliko metoda rastresanja za problem klasifikacije i semantičke segmentacije slika te ih vrednovati i usporediti. Eksperimentima ćemo pokazati i odrediti najbolje metode rastresanja za pojedine probleme dubokog učenja te ćemo pokušati pronaći metode kojim bi se izjednačili ili barem približno postigli rezultati s većim brojem označenih primjera.

# 2. Konvolucijske neuronske mreže

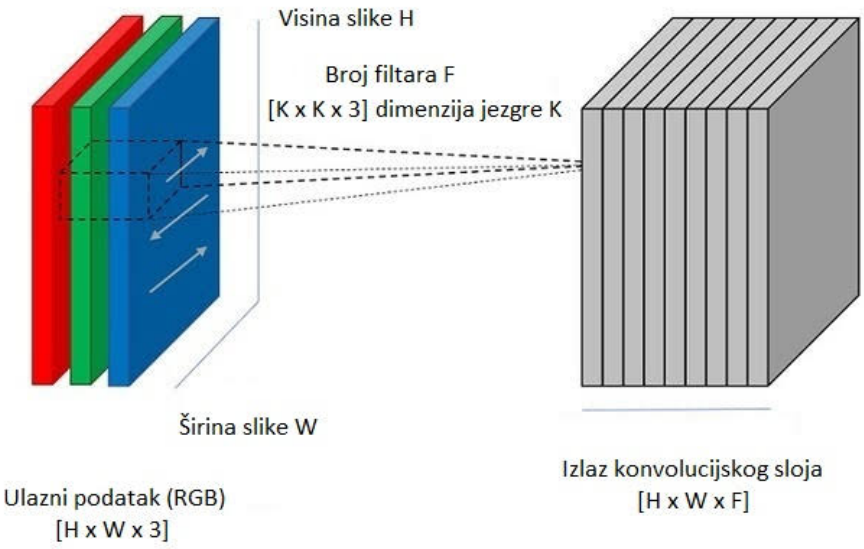
Za rješavanje problema računalnog vida, a time i klasifikacije slika, koriste se konvolucijske mreže zbog prirode konvolucijskog sloja. Trenutno najbolji rezultati na području klasifikacije slika postignuti su upravo učenjem konvolucijskih neuronskih mreža. Nad popularnim i često korištenim skupovima poput MNIST-a i CIFAR-10 takvi modeli postižu vrlo visoke točnosti (primjerice 99.61% za MNIST [15] i 96.53% za CIFAR-10 [9] Slično kao i kod neuronskih mreža, konvolucijske neuronske mreže imaju definirane slojeve s neuronima koji primaju ulazne podatke, izvršavaju računske operacije i predaju izlaz sljedećem sloju. Razlika je što konvolucijske mreže uz ostale slojeve sadrže barem jedan konvolucijski sloj. Bez obzira na malo drugačiju strukturu, konvolucijske mreže i dalje na izlazu daju ocjenu razreda ulaznog podatka, ali uz manju računsku kompleksnost i dijeljenje težina što ih čini memorijski i računski efikasnijima od običnih neuronskih mreža. Zbog toga, konvolucijske mreže koriste se za modeliranje rješenja problema s područja računalnog vida te ispoljavaju zadovoljavajuće rezultate.

## 2.1. Slojevi konvolucijske neuronske mreže

### 2.1.1. Konvolucijski sloj

Prvi sloj u svakoj konvolucijskoj neuronskoj mreži uvijek je konvolucijski. On sadrži filtre i njihove težine koje je potrebno naučiti kako bi model radio kvalitetne predikcije. Konvolucijski sloj sliku u boji ne promatra u dvije dimenzije poput čovjeka, već ju promatra kao trodimenzionalni objekt s dubinom 3 (po jedan sloj za svaku od 3 osnovne boje). Slika dimenzija 64x64 će stoga biti spremljena u matricu dimenzija 64x64x3. Na takav ulazni podatak primjenjuju se filtri te svaki od njih stvara po jednu dvodimenzionalnu aktivacijsku mapu. [12] Izlaz iz konvolucijskog sloja je, dakle, skup svih izlaznih matrica ukupne dubine jednake broju filtara. Slika 2.1 prikazuje primjenu konvolucije na sliku visine H, širine W s 3 kanala za boje. Primjenjeno je F filtara s

konvolucijskom jezgrom visine i širine K. Izlazni podatak je dimenzija $H * W * F$.




**Slika 2.1:** Primjena konvolucije na ulazni podatak.

### 2.1.2. Aktivacijski sloj i sloj sažimanja

Aktivacijski sloj ima ulogu aktivacije neurona. Pošto na ulaz aktivacijski sloj dobiva zbroj prethodnih ulaza množenih pripadajućim težinama (engl. *weights*) i pomaka (engl. *bias*), odnosno

$$Y = \sum(weight * input) + bias$$

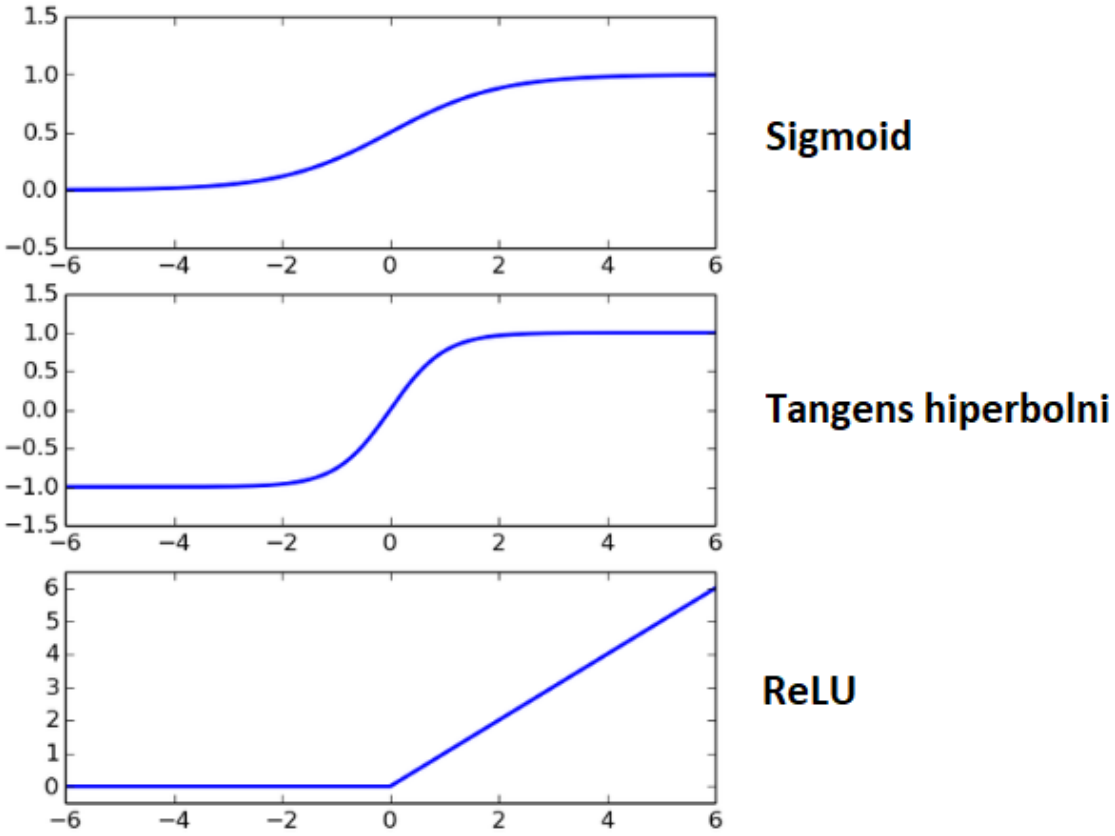
ulaz u aktivacijski sloj može poprimiti vrijednosti iz intervala $[-\infty, \infty]$. Kako mreža ne zna granice koje bi izlaz trebao poprimiti, potreban je način da se odredi koji će se neuroni "aktivirati" i koliki će imati utjecaj. Upravo tu zadaću ima aktivacijski sloj koji nad ulaznim podatkom izračuna vrijednost aktivacijske funkcije i tu vrijednost propagira dalje. Bez korištenja aktivacijskih slojeva, mreža bi bila isključivo linearna i ne bi bila sposobna učiti kompleksne značajke ulaza. Linearna mreža nikako nije dobra jer bez obzira koliko je duboka, ponašat će se jednako onoj s jednim linearnim slojem. Nelinearni aktivacijski sloj dodaje potrebnu nelinearnost u strukturu mreže. Najčešće korištene aktivacijske funkcije prikazane su u nastavku.

**ReLU (engl. *Rectified Linear Unit*):** $f(x) = \max(0, x)$

**Tangens hiperbolni:** $f(x) = tanh(x) = \frac{2}{1+e^{-2x}} - 1$

**Sigmoid:** $f(x) = \frac{1}{1+e^{-x}}$

Pripadajući grafovi funkcija vidljivi su na slici 2.2.



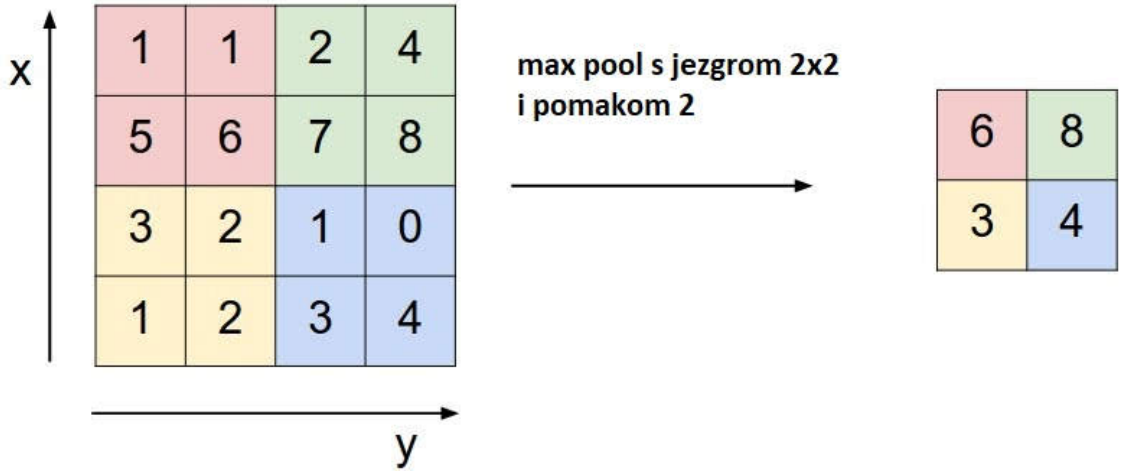

**Slika 2.2:** Grafovi aktivacijskih funkcija

Operacije nad podacima velikih dimenzija često su vremenski i računski zahtjevne. Sloj sažimanja (engl. *pooling layer*) smanjuje dimenzije mape i broj parametara za računanje. Sažimanje podataka postiže se korištenjem funkcija sažimanja koje rade tako da na manjim dijelovima ulaza izračunaju jedan podatak i tako stvore izlaz manjih dimenzija. To se postiže korištenjem jezgre (engl. *kernel*) određenih dimenzija (najčešće 2x2) te se takvom jezgrom nakon svakog pomaka (najčešće iznosa 2 u kombinaciji s prethodno navedenom jezgrom) izračuna vrijednost i upiše na pripadajuće mjesto u novom, sažetom izlaznom podatku. Dubina podatka prilikom ove operacije ostaje nepromijenjena. Ovaj sloj se u većini mreža postavlja nakon jednog ili između više sukcesivnih konvolucijskih slojeva [12] te se najčešće koristi računanjem maksimalne vrijednosti (engl. *max pool*) ili srednje vrijednosti (engl. *average pool*). Funkcija maksimalne vrijednosti odbacuje tri od četiri vrijednosti (podrazumijevajući pritom jezgru dimenzija 2x2) te tako odbacuje 75% aktivacijskih vrijednosti što je i razlog najčešćim

posezanjem za ovom funkcijom u sloju sažimanja. Općenito, sloj sažimanja može se prikazati sljedećim koracima:

- Ulaz dimenzija W x H x D, gdje su
  - W - širina ulaznog podatka
  - H - visina ulaznog podatka
  - D - dubina ulaznog podatka
- Iščitavanje parametara F i S, gdje su
  - F - dimenzija jezgre
  - S - pomak (engl. *stride*)
- Funkcijom sažimanja stvaranje izlaza dimenzija W' x H' x D', gdje su
  - $W' = 1 + \frac{W-F}{S}$
  - $H' = 1 + \frac{H-F}{S}$
  - $D' = D$

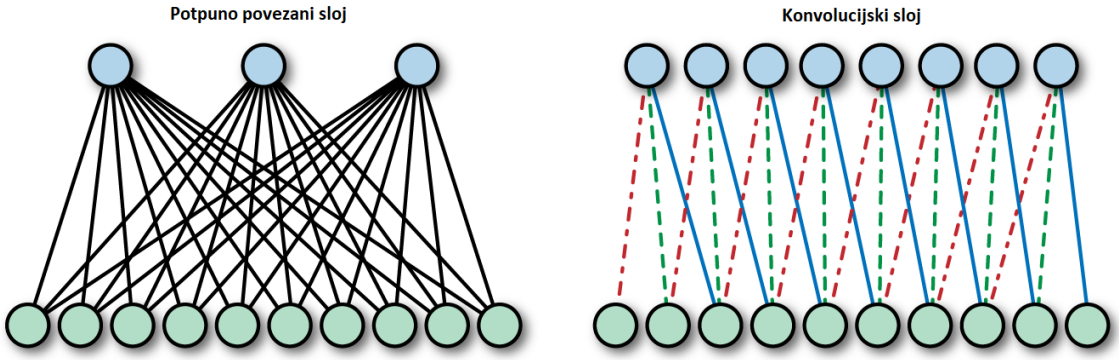
Slika 2.3 jasno prikazuje rezultat sažimanja funkcijom maksimalne vrijednosti.




**Slika 2.3:** Sažimanje maksimalnom vrijednošću

### 2.1.3. Potpuno povezani sloj

Potpuno povezani sloj sastoji se od neurona pri čemu svaki od njih na ulazu prima sve aktivacije prethodnog sloja u mreži. Iz tog razlog nakon njega u mreži ne mogu postojati dodatni konvolucijski slojevi. Njegova uloga je klasifikacija na temelju značajki izdvojenih u prethodnim slojevima. Potpuno povezani sloj stoga je gotovo uvijek




**Slika 2.4:** Potpuno povezani i konvolucijski sloj.

posljednji sloj mreže. Aktivacijske vrijednosti u neuronima računaju se matričnim množenjem ulaza s težinama uz zbrajanje pomaka. Na slici 2.4 je prikazana razlika između potpuno povezanog sloja i konvolucijskog sloja.

## 2.2. Optimizacijski algoritmi

Učenje modela svodi se na korištenje optimizacijskog algoritma koji pri svakom koraku učenja izračunava korekcije težina u modelu. Minimizacijom funkcije pogreške pokušavamo doseći minimum, odnosno pokušavamo svesti pokrešku modela na nulu. Funkcije pogrešaka dizajnirane su na način da što bolje predstavljaju stvarni problem koji pokušavam riješiti. Za, primjerice, klasifikaciju pokušavamo smanjiti razliku izlaza modela koji predstavlja vjerojatnosti pripadnosti primjera svakom od razreda i stvarne pripadnosti primjera određenom razredu. Pošto su funkcije pogreške najčešće derivabilne, minimum možemo probati pronaći izravnim rješenjem - odnosno rješavanjem prve derivacije dotične funkcije. No, većina ovih problema nema rješenje u zatvorenom obliku pa je najbolji pristup koristiti gradijentni spust kojim se u svakoj iteraciji približavamo minimumu funkcije.

### 2.2.1. Stohastički gradijentni spust

Stohastički gradijentni spust izvršava osvježavanje parametara u svakom koraku učenja. Za svaki primjer $x^{(i)}$ i njegovu pripadajuću oznaku $y^{(i)}$ izračunavaju se novi parametri po sljedećoj formuli.[18]

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i)}; y^{(i)}) \tag{2.1}$$

Prethodna formula radi korekciju parametara za svaki pojedinačni primjer. Kako

bi se smanjila računska složenost, najčešće se korekcija radi nad manjim skupom podataka, odnosno mini grupom (engl. *mini-batch*). Tada se korekcija izvodi nad više podataka od jednom:

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i:i+n)}; y^{(i:i+n)}) \tag{2.2}$$

Na ovaj način također se smanjuje varijanca osvježavanja parametara što dovodi do stabilnije konvergencije i mogu se iskoristiti optimizirani algoritmi za rješavanje matričnih operacija što uvelike ubrzava postupak i smanjuje računsku složenost algoritma.

Neki od izazova na koje nailazimo je određivanje pogodne stope učenja $\eta$. Tijekom konvergencije, moguće je da je stopa učenja premalena što dovodi do vrlo spore konvergencije ili čak divergencije. Ako je pak stopa učenja prevelika, pogreška modela pasti će do neke mjere, ali korak učenja zatim će biti prevelik da dođe do minimuma.

### 2.2.2. Adam

U ovome radu kao optimizacijski algoritam koristi se upravo Adam (Adaptive Moment Estimation) koji rješava neke od problema s kojima se suočava SGD. Adam uvodi adaptivne stope učenja za svaki od parametara. [11] Kako bi se smanjenje stope učenja odradilo u pravi trenutak, algoritam prati eksponencijalno prigušeni prosjek kvadrata starih vrijednosti gradijenata kao i kod algoritma RMSprop[19]. Na ovaj način, novije vrijednosti imaju veći utjecaj:

$$r_t = \rho_{2r_{t-1}} + (1 - \rho_2) g_t^2 \tag{2.3}$$

Razlika u odnosu na RMSprop algoritam je u tome što Adam prati i eksponencijalno prigušeni prosjek gradijenata bez kvadrata. Ovime se postiže da je moment implicitno ugrađen u praćenje kretanja gradijenata.

$$s_t = \rho_{1s_{t-1}} + (1 - \rho_1) g \tag{2.4}$$

Pseudokod algoritma Adam[20] prikazan je u nastavku:

**Potrebno:** Vrijednost koraka učenja $\epsilon$, preporučeno 0.001

**Potrebno:** Stope eksponencijalnih prigušenja za estimaciju momenta $\rho_1$ i $\rho_2$ iz intervala $[0, 1)$

**Potrebno:** Mala konstanta $\delta$ za numeričku stabilnost (Preporučeno $10^{-8}$)

**Potrebno:** Inicijalni skup parametara $\theta$

Inicijalizirati varijable momenata $s = 0$ i $r = 0$

Inicijalizirati vremenski korak $t = 0$

**while** *Uvjet zaustavljanja nije ispunjen* **do**

Uzeti mini grupu podataka veličine ***m*** iz skupa za učenje $\{x^{(1)}, ...., x^{(m)}\}$ i pripadajuće oznake $y^{(i)}$

Izračunati gradijent ***g***

$t \leftarrow t + 1$

Osvježiti pristranu procjenu prvog momenta: $s \leftarrow \rho_1 \mathbf{s} + (1 - \rho_1)\mathbf{g}$

Osvježiti pristranu procjenu drugog momenta: $r \leftarrow \rho_1 \mathbf{r} + (1 - \rho_1)\mathbf{g} \odot \mathbf{g}$

Korekcija pristranosti u prvom momentu: $\hat{s} \leftarrow \frac{\mathbf{s}}{1-\rho_1^t}$

Korekcija pristranosti u drugom momentu: $\hat{r} \leftarrow \frac{\mathbf{r}}{1-\rho_2^t}$

Izračunati korekciju parametara: $\Delta\theta = -\epsilon\frac{\hat{s}}{\sqrt{\hat{r}}+\delta}$

Primjeni korekciju: $\theta \leftarrow \theta + \Delta\theta$

**end**

**Algoritam 1:** Pseudokod optimizacijskog algoritma Adam

# 3. Metode rastresanja podataka

Tijekom razvoja metoda strojnog i dubokog učenja pojavila se potreba za povećanjem skupa podataka. Kako bi se izbjeglo dodatno prikupljanje i označavanje novih podataka, razvijene su razne metode kojima bi se poveća postojeći skup te time pridonijelo kvaliteti modela. Kako postoji više problema koji se rješavaju strojnim učenjem, potrebno je bilo osmisliti različite metode rastresanja za pojedine tipove podataka. Tako primjerice za probleme koji se bave prirodnim jezikom postoje metode poput povratnog prijevoda (engl. *back translation*) u kojoj se određeni dio teksta prevodi na neki drugi jezik te se zatim prevodi u originalni jezik korištenjem primjerice Googlovog prevoditelja. Time se dobiva različiti tekst s istim semantičkim značenjem te se time efektivno povećava skup označenih podataka. U ovom radu, naglasak će ipak biti na metodama rastresanja podataka za konvolucijske modele, odnosno na metodama rastresanja za slike. Najjednostavnije metode koje garantiraju povećanje točnosti modela za nekoliko postotaka podrazumijevaju jednostavno manipuliranje slikom - zrcaljenje po vertikalnoj osi, rotacija slike za neki mali kut (kako bi slika i dalje bila reprezentativna), promjena kontrasta, izrezivanje manjeg dijela slike, dodavanje Gaussovog šuma u sliku i slične metode. Razvojem metoda u ovome polju, došlo je do napredaka u smislu korištenja neoznačenog skupa uz manji označeni skup podatataka (polunadzirano učenje) i korištenja samo neoznačenog skupa podataka (nenadzirano učenje). U ovome radu naglasak je na usporedbi metoda rastresanja u nadziranoj okolini. U nastavku su opisane metode koje su korištene i vredovane u sklopu ovog rada.

## 3.1. AutoAugment

Kvalitetne metode rastresanja podataka iziskuju ekspertno znanje kako bi se osmislile transformacije kojima će se dobivati novi podaci koji će stvarno pridonijeti boljim performansama modela. Metoda AutoAugment pokušava zamijeniti ekspertno znanje novim modelom koji će naučiti najbolje transformacije slika za određeni skup podataka. Cilj ove metode je pronaći najbolje transformacije u prostoru pretraživanja koji

sadrži mnogo transformacija. Problem je postavljen kao potpomognuto učenje. Najbolje transformacije uče se na način da su one koje daju najveću preciznost nagrađivane. Motivacija za ovako postavljen problem leži u tome da neke transformacije dobro rade za određene skupove podataka, dok za druge ne rade. Primjer je horizontalno zrcaljenje koje dobro radi nad CIFAR-10 skupom podataka, dok za MNIST skup podataka ne radi zbog drugačijih simetrija prisutnih u ovim skupovima podataka. Ovaj postupak cilja na automatizaciju procesa pronalaska najboljih transformacija za određeni skup. Prostor pretraživanja uključuje nekoliko politika (engl. *policy*) od kojih svaka sadrži nekoliko odabira i redoslijeda operacija rastresanja. Svaka od tih operacija je funkcija obrade slike (translacija, rotacija, normalizacija boja...).[5] U originalnom članku, autori provode eksperimente na skupovima CIFAR-10, CIFAR-100, SVHN. ImageNet te reduciranim skupovima CIFAR-10 i SVHN. U sklopu ovog rada koncentrirat ćemo se na skup podataka CIFAR-10. Eksperimentima u članku pokazalo se kako model koji uči najbolje transformacije bolje radi na manjem skupus više epoha nego s mnogo podataka, a malo epoha. Također se pokazalo kako na ovom skupu podataka transformacije koje uključuju promjenu boja bolje rade u odnosu na translaciju, pomicanje i slične druge transformacije.

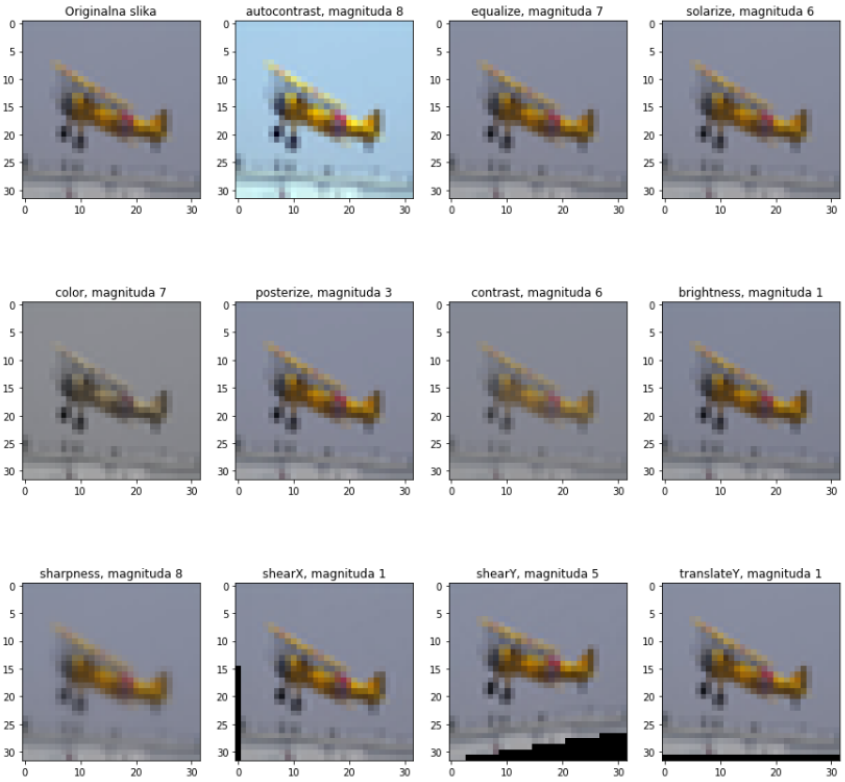
### 3.1.1. Učenje povoljnih transformacija

Moguće transformacije koje tvore prostor pretraživanja navedene su i opisane u tablici 3.1. Ukupno je to 16 operacija od kojih svaka ima pridodan opseg magnituda koje su raspodjeljene u diskretni niz od 10 iznosa kako bi se mogao koristiti diskretni algoritam pretraživanja. Svakoj transformaciji također je pridodana i vjerojatnost primjenjivanja transformacije diskretizirana na 11 vrijednosti. Cilj je pronaći 5 najboljih transformacija te je stoga prostor pretraživanja veličine $(16 \times 10 \times 11)^{10}$.

Model je konstruiran od dvije mreže. Prva mreža je tzv. *Controller RNN*, a drugi dio je model-dijete (engl. *child model*) koji je neuronska mreža koja uči klasifikaciju koristeći rastresene podatke. RNN dio modela je nagrađivan kada dotična transformacija poboljšava točnost u podmreži za klasifikaciju. Ovaj dio modela je jednoslojna LSTM mreža sa 100 skrivenih čvorova sa *softmax* predikacijskim slojem. Tijekom svakog koraka, RNN dio mreže uzorkuje nekoliko transformacija iz početnog prostora pretraživanja, a zatim je podmreža evaluirana nad validacijskim skupom kako bi se izmjerila točnost i skladno s time nagradila glavna mreža RNN. Na kraju svakog pretraživanja, 5 najboljih politika (engl. *policies*) konkatenirano je u jednu politiku. Konačni rezultat učenja je 5 najboljih politika s ukupno 25 pod-politika (engl. *sub-policy*).

| Ime operacije | Opis | Opseg magnituda |
|---|---|---|
| ShearX(Y) | Pomicanje jednog dijela slika u jednom smjer, a drugoj dijela u drugom smjeru pravcem X ili Y osi u veličini magnitude. | [-0.3, 0.3] |
| TranslateX(Y) | Translacija slike u smjeru vertikalne ili horizontalne osi određenom magnitudom. | [-150, 150] |
| Rotate | Rotiranje slike za neki kut veličine magnitude. | [-30, 30] |
| AutoContrast | Maksimizacija kontrasta slike, postavljanjem najtamnijeg piksela na crnu i najsvjetlijeg piksela na bijelu boju. | [0, 256] |
| Invert | Invertiranje piksela slike. | |
| Equalize | Ujednačavanje histograma slike. | |
| Solarize | Invertiranje svih piksela iznad praga iznosa magnitude. | [0, 256] |
| Posterize | Reduciranje broja bitova za svaki piksel na broj bitova iznosa magnitude. | [4, 8] |
| Contrast | Kontrola kontrasta slike. Magnituda 0 daje sivu sliku, dok magnituda 1 daje originalnu sliku. | [0.1, 1.9] |
| Color | Prilagodba ravnoteže boja na slici. Magnituda 0 daje crnu sliku, dok magnituda 1 daje originalnu sliku. | [0.1, 1.9] |
| Brightness | Prilagodba svjetline slike. Magnituda 0 daje crnu sliku, dok magnituda 1 daje originalnu sliku. | [0.1, 1.9] |
| Sharpness | Prilagodba oštrine slike. Magnituda 0 daje zamućenu sliku, dok magnituda 1 daje originalnu sliku. | [0.1, 1.9] |
| Cutout | Postavlja nasumični kvadrat duljine stranice vrijednosti magnitude na sivu boju. | [0, 60] |
| Sample Pairing | Linearno dodavanje druge slike (odabrane nasumično) s težinom magnitude i bez mijenjanja oznake. | [0, 0.4] |

**Tablica 3.1:** Opisi operacija za rastresanje i vrijednosti njihovih magnituda.




**Slika 3.1:** Prikaz operacija korištenih u RandAugment i AutoAugment metodama i njihovih učinaka.

## 3.2. RandAugment

Iako se učenje najboljih transformacija za rastresanje podataka čini itekako primamljivo, uz prednosti dolaze i svojevrsni nedostaci. Sama činjenica kako je za postupak potrebno optimizirati dvije zasebne mreže znači kako je postupak računski, a i vremenski skup. Neki projekti ne mogu si priuštiti dodatno vrijeme za pronalaženje politika rastresanja pa se javlja potreba za automatiziranim postupkom bez dodatnog pretraživanja prostora. U radu koji opisuje AutoAugment autori su pokazali kako čak i nasumično odabrane politike daju poboljšane rezultate, odnosno poboljšavaju generalizaciju konačnog modela. RandAugment metoda bazira se na nasumičnom odabiru politika iz početnog određenog prostora pretraživanja. Kako bi ovaj postupak bio moguć, potrebno je radikalno smanjiti prostor pretraživanja. Kao rješenje problema, predložen je pojednostavljeni prostor pretraživanja koji koristi samo dva interpretabilna hiperparametra. [6] Ovom formulacijom problema moguće je iskoristiti jednostavno potpuno pretraživanje po svim hiperparametrima (engl. *grid search*). U nastavku je prikazan programski kod funkcije *randaugment*.

```
transforms = [
'Identity', 'AutoContrast', 'Equalize',
'Rotate', 'Solarize', 'Color', 'Posterize',
'Contrast', 'Brightness', 'Sharpness',
'ShearX', 'ShearY', 'TranslateX', 'TranslateY']
def randaugment(N, M):
  sampled_ops = np.random.choice(transforms, N)
  return [(op, M) for op in sampled_ops]
```

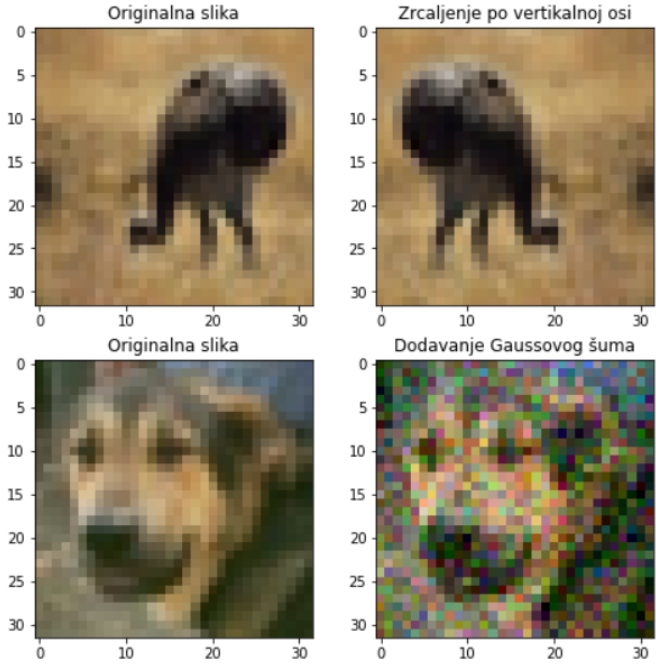
U prikazanom kodu, argumenti N i M predstavljaju broj transformacija za rastresanje i magnitudu danih transformacija.

U članku, autori su pokazali kako se mogu dobiti *state-of-the-art* rezultati u kombinaciji s određenim modelima poput Wide-ResNet i PyramidNet. Također se pokazalo kako konačni dprinos rastresanja podataka i dalje uvelike ovisi o složenosti modela i veličini početnog skupa podataka za učenje.

## 3.3. Ostale metode rastresanja

Metode rastresanja podataka za slike mogu se podijeliti u dvije glavne skupine - tradicionalne metode (metode koje pozantim funkcijama djeluju na sliku) i metode bazirane na dubokim modelima, tzv. *black box* metode.[16] Ranije spomenute tradicionalne metode u metodi AutoAugment korištene su u kombinaciji s dubokim učenjem, dok metoda RandAugment predlaže pretraživanje prostora s mogućim operacijama.

Uz prethodno navedene metode, u ovome radu vrednovana je i metoda rastresanja koja koristi samo zrcaljenje po vertikalnoj osi te dodavanje Gaussovog šuma u sliku. Na slici 3.2 vidljivi su primjeri efekta obje spomenute operacije.




**Slika 3.2:** Prikaz dvaju operacija korištenih u vlastitoj imlementaciji metode rastresanja podataka.

Zrcaljenje po vertikalnoj osi moguće je koristiti nad skupom podataka CIFAR-10 zbog simetrije podataka u samome skupu. Ovim postupkom ne mijenjamo oznaku slike, a efektivno dvostruko povećavamo skup podataka jer je slika pri učenju interpretirana kao skup brojeva pa model originalnu i zrcaljenu sliku tretira kao dvije sasvim različite.

Metoda dodavanja Gaussovog šuma ima za posljedicu bolju generalizaciju modela. Učenjem nad šumovitim slikama indirektno dolazi do regularizacije te je stoga model otporniji na kasnije neviđene, ali šumovite slike. U nastavku je prikazana funkcija kojoj dodajemo šum u sliku pomoću paketa *scikit-image*.

```
from skimage.util import random_noise
noisy_img = random_noise(img, mode='gaussian',
        mean=mean, var=std)
```

Od metoda baziranih isključivo na dubokim modelima valja spomenuti metode koje koriste generativne neprijateljske mreže (engl. *Generative Adversarial Networks - GAN*). GAN mreže koriste sposobnost da razlikuju prave primjere od onih umjetno generiranih. Dvije mreže natječu se - jedna generira primjere dok druga mreža pokušava razabrati prave primjere od generiranih primjera prve mreže. Međusobnim natjecanjem, mreže se poboljšavaju te dobivamo primjere koje je vrlo teško razlikovati od pravih. [8] Ovaj pristup moguće je stoga iskoristiti kako bi se na umjetan način povećao skup primjera za učenje. Antoniou et al. [1] su tako konstruirali tzv. DAGAN (engl. *Data Augmentation GAN*) mrežu te uspješno generirali nove podatke za skupove podataka Omniglot, EMNIST, and VGG-Faces. Na slici 3.3 prikazan je isječak generiranih slika koristeći DAGAN. [1]

**Slika 3.3:** Prikaz generiranih lica (Gore lijevo je jedina prava slika).

Iz navedene slike vidljivo je kako ova metoda može dati vrlo zadovoljavajuće rezultate. Nedostatak je, kao i kod AutoAugment metode, dodatna računska i vremenska složenost kako bi se DAGAN naučio generirati dobre primjere. U današnje vrijeme, kada se na području dubokog učenja koriste sve složenije i dublje mreže koje iziskuju veliku količinu podataka za učenje, ovakve metode ipak su dobra alternativa ručnom označavaju dodatnih primjera, pogotovo kada to uopće nije ni moguće.

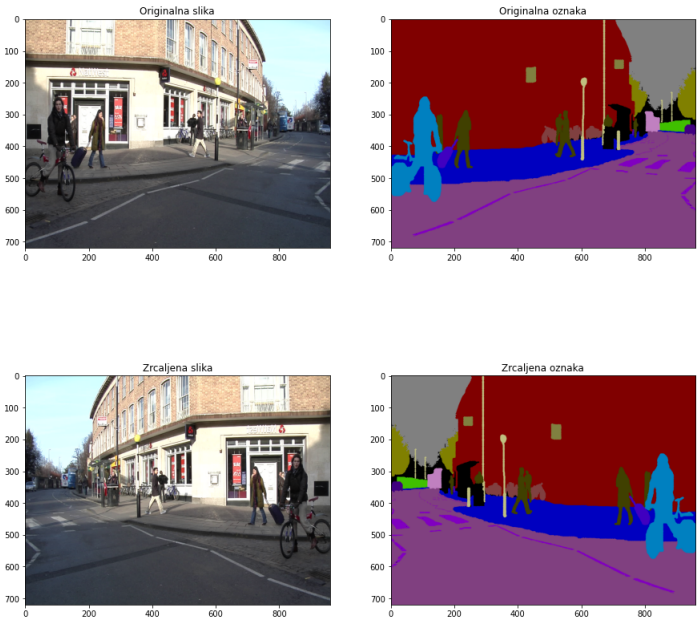
## 3.4. Metode rastresanja podataka za semantičku segmentaciju

Semantičke segmentacija pokazala se kao posebni problem u području računalnog vida. S obzirom da svaki piksel na slici ima svoju oznaku, označavanje pojedinih slika dugotrajan je i skup posao. Povećavanje skupova podataka na način da se mijenja originalna slika, kao što je opisano u prethodnim poglavljima, nije uvijek moguće iz razloga što neke transformacije implicitno mijenjaju *oznaku slike*. Pod time se misli da se transformacijama može promjeniti položaj pojedinih objekata na slici i time postojeća oznaka postaje nevažeća. Stoga dolazimo do zaključka kako je moguće koristiti samo transformacije koje ne mijenjaju položaj piksela u slici već samo vrijednosti piksela. Ovoj skupini pripadaju operacije poput *solarize, autocontrast, brightness* i sl. Druga mogućnost je da se koriste operacije koje je moguće jednako i u potpunosti primjeniti i na pripadajuću oznaku slike. Od operacija iz ove skupine možemo spomenuti rotaciju slike za neki mali kut (uz pripadajuće popunjavanje slike praznim pikselima) ili zrcaljenje slike s obzirom na neku os.

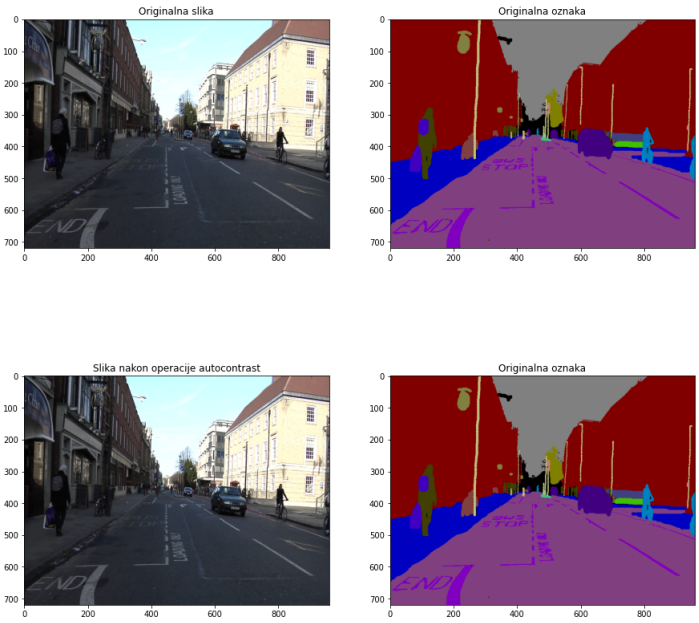
U ovome radu bit će implementirane metode iz prve spomenute skupine, konkretnije zrcaljene po vertikalnoj osi i metode *autocontrast* i *brightness*. Na slici 3.4 prikazana je operacija zrcaljenja, a na slici 3.5 prikazana je primjena operacije *autocontrast*. Obje operacije iz navedenih slika primjenjene su na slike iz skupa podataka CamVid.

Kako bi se komplementirale osnovne metode navedene prethodno, postoje dodatne metode bazirane na dubokom učenju koje potpomažu rastresanju podataka za semantičku sgmentaciju. Slično kako je navedeno u poglavlju 3.3, i u ovome području postoje metode koje koriste pristupe temeljene na GAN modelima. Liu et al. [14] koriste GAN kako bi generirali realistične slike i oznake piksela i tako poboljšali točnost modela za semantičku segmentaciju.

Wang et al. [21] u svome članku istražuju mogućnosti povećanja skupa podataka za semantičku segmentaciju sasvim drugačijim pristupom. Ovom metodom implicitnog rastresanja podataka (engl. *Implicit Semantic Data Augmentation*) pokušava se promjeniti slika na način da promjene imaju semantičkog smisla - primjerice da primjene odgovaraju dodavanju naočala na lice osobe ili promjenu pozadine.




**Slika 3.4:** Prikaz primjene operacije zrcaljenja na sliku i pripadajuću sliku koja sadrži oznake za svaki piksel.




**Slika 3.5:** Prikaz primjene operacije *autocontrast* na sliku.

# 4. Eksperimenti

## 4.1. Klasifikacija

Klasifikacija slika jedan je od osnovnih i često susretanih problema u dubokom učenju. Primjenu klasifikacije slika moguće je pronaći u raznim granama znanosti - medicine, geologije, meteorologije i sl. Glavna ideja je naučiti model nad postojećim označenim podacima kako bi se mogli rješavati problemi iz stvarnoga života. Kako bi to bilo moguće, označeni podaci se pripremaju kako bi se na njima moglo učiti. Svakome razredu pridodajemo brojčanu reprezentaciju - ako u problemu postoji 10 razreda onda im pridodajemo vrijednosti 0-9. Slike reprezentiramo, kao što je to u računalima normalno, trojkama koje predstavljaju udio crvene, zelene i plave boje u svakome od piksela. Postoje također i slike koje nisu u boji - tada kažemo da ti slika sivih nijansi (engl. *greyscale*) i tada se piksel reprezentira samo jednim cijelim brojem.

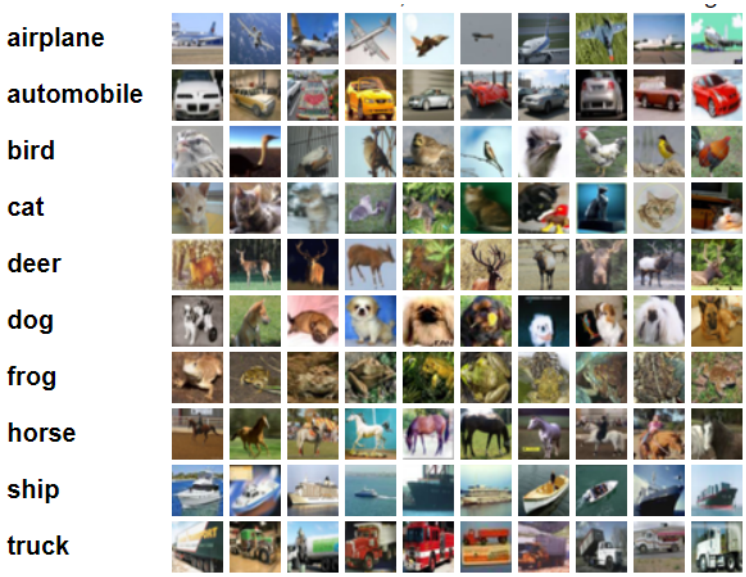
Za rješavanje problema klasifikacije uobičajeno je koristiti duboke konvolucijske modele. Ovi modeli zasnovani su na spajanju više konvolucijskih slojeva spomenutih u poglavlju 2 u jednu koherentu cjelinu. Konvolucijski filtri sposobni su naučiti prepoznavati neke uzorke na slikama. Od jednostavnijih stvari u ranijim slojevima - poput linija, krugova i sličnih oblika pa sve do kompozicija tih oblika u kasnijim slojevima, primjerice lica sastavljenog od očiju, nosa i usta.

U ovome radu korišteni su upravo konvolucijski modeli kako bi se proveli eksperimenti vezani za klasifikaciju slika te ispitao utjecaj korištenja metoda rastresanja u svrhu povećanja skupa za učenje i bolje generalizacije modela.

### 4.1.1. CIFAR-10

Ovaj skup podataka često se koristi pri učenju i provjeri modela za problem klasifikacije slika. CIFAR-10 zapravo je označeni podskup skupa *80 millon tiny images*. Kao što to oslikava brojka u imenu, skup je podijeljen u 10 razreda sa 6000 slika za svaki razred. Ukupno, skup se sastoji od 60 000 slika podijeljenih u u dva podskupa: pod-

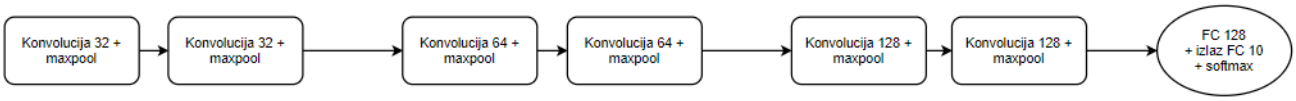
skup za učenje i podskup za ispitivanje. Skup za učenje sastoji se od 50 000, a ispitni skup od 10 000 slika. Primjeri slika iz svakog razreda vidljivi su na slici 4.1.




**Slika 4.1:** Razredi u skupu podataka CIFAR-10 s 10 nasumično odabranih slika iz svakog razreda.[13]

### 4.1.2. Implementacija modela i tijek učenja

Za ovaj problem odabran je konvolucijski model sa šest konvolucijskih slojeva. Prva dva sloja sačinjena su od po 32 konvolucijska filtera, zatim slijedi sloj sažimanja po maksimalnoj vrijednosti. Nakon toga slijede 2 jednaka bloka, s razlikom da su u prvome bloku konvolucijski slojevi s 64 filtera, a u posljedenjm sa 128 filtera. Posljednji dio modela čini potpuno povezani sloj sa 128 neurona i zatim izlazni sloj sa brojem neurona jednakim broju razreda, odnosno 10 i aktivacijskom funkcijom *softmax* kako bismo dobili vjerojatnosnu interpretaciju rezultata. Model je pojednostavljeno prikazan na slici 4.2.




**Slika 4.2:** Grafički prikaz slojeva u modelu.

S obzirom da se učenje odvija s više prolazaka svih podataka kroz modela, odnosno u više epoha, moguća je pojava prenaučenosti. Prenaučenost se javlja kada model radi

s velikom točnošću na skupu za učenje, ali na novim, neviđenim podacima, primjerice na ispitnom skupu, ne daje jednako dobre rezultate. Kako bi se ovo spriječilo, uvodimo neku od metoda regularizacije. U ovome slučaju, nakon svakog bloka konvolucijskih slojeva, dodan je tzv. *dropout* sloj koji odbacuje nasumično čvorove u mreži sa zadanom vjerojatnošću. Vjerojatnost odbacivanja postavljena je na 0.2, odnosno 20%. Kao optimizacijski algoritam za ovaj model odabran je Adam. Adam se u raznim zadaćama pokazao kao vrlo dobar algoritam te je stoga odabran i u ovome problemu. Prednosti i sam algoritam objašnjeni su u poglavlju 2.2.2. Zadaća optimizacijskog algoritma minimizirati je funkciju pogreške, a kao logičan odabir za problem klasifikacije odabrana je pogreška unakrsne entropije. Ova pogreška kažnjava razliku između vjerojatnosti pripadnosti određenome razredu i stvarnom razredu kojem primjer pripada. U slučaju binarne klasifikacije (broj razreda jednak je 2), funkcija pogreške unakrsne entropije izgleda ovako:

$$L = -ylog(p) + (1-y)log(1-p) \tag{4.1}$$

gdje $y$ predstavlja stvarnu oznaku primjera, a $p$ vjerojatnosnu predikciju modela. U slučaju više razreda, kao što je to slučaj sa skupom podataka CIFAR-10, funkcija pogreške koju minimiziramo dana je sljedećom formulom:

$$L = -\sum_{c=1}^{M} y_{o,c} log(p_{o,c}) \tag{4.2}$$

gdje M predstavlja broj razreda (u našem slučaju 10), $c$ predstavlja stvarnu oznaku razreda, a $o$ oznaku razreda s najvećom vjerojatnošću na izlazu modela.

### 4.1.3. Validacija hiperparametara

Kako bismo odredili povoljne hiperparamtre - stopu učenja, regularizacijsku konstantu, broj epoha i sl. potreno je bilo izdvojiti još jedan podskup. Od skupa za učenje, odvojeno je 5000 primjera i stvoren je validacijski skup nad kojim se podešavaju hiperparametri kako bismo osigurali da model ne *vidi* ispitne podatke sve do konačne procjene modela. Konačna podjela skupa izgleda ovako:

- Skup za učenje: 45 000 primjera
- Skup za validaciju: 5000 primjera
- Ispitni skup: 10 000 primjera

Ispitivanjem raznih stopa učenja, pokazalo se kako je stopa 0.001 najbolja te da model s njome ne zapne u lokalnom minimumu. Učenje sa cijelim skupom podataka se

odvijalo u 25 epoha jer se pokazao kako treniranjem na više od tog broja ne pridonosi točnosti modela, no broj epoha je varirao u odnosu s ukupnim brojem primjera za učenje (kada se uključi rastresanje podataka). Također, provedeni su i eksperimenti s l2-regularizacijom *dropout* u slučaju kada je ostavljen manji broj označenih primjera uz veliki broj operacija rastresanja kako bismo izbjegli prenaučenost i usporedili ove dvije metode regularizacije.

| Broj označenih primjera | Broj operacija rastresanja | Ukupni broj primjera za učenje | Epohe | Točnost |
|---|---|---|---|---|
| 45 000 | 0 (baseline) | 45 000 | 25 | 0.7002 |
| 45 000 | 1 | 90 000 | 25 | 0.7968 |
| 45 000 | 2 | 135 000 | 25 | **0.8062** |
| 45 000 | 4 | 225 000 | 25 | 0.7761 |

**Tablica 4.1:** AutoAugment, prikaz točnosti s cijelim skupom podataka.

U tablici 4.1 prikazani su rezultati za metodu AutoAugment. Vidljivo je kako je korištenje ove metode povećalo točnost modela za više od 10%. Također je vidljivo da najbolji rezultat dobivamo kada napravimo po dvije operacije rastresanja za svaku pojedinu sliku. Sličan rezultat dobivamo i samo s udvostručavanjem skupa, dok radikalno povećavanje skupa i dalje pridonosi točnosti modela, ali u manjoj mjeri.

| Broj označenih primjera | Broj operacija rastresanja | Ukupni broj primjera za učenje | Epohe | Točnost |
|---|---|---|---|---|
| 45 000 | 0 (baseline) | 45 000 | 25 | 0.7002 |
| 45 000 | 1 | 90 000 | 25 | 0.7775 |
| 45 000 | 2 | 135 000 | 25 | 0.7881 |
| 45 000 | 4 | 225 000 | 25 | **0.7968** |

**Tablica 4.2:** RandAugment, prikaz točnosti s cijelim skupom podataka.

U tablici 4.2 prikazani su rezultati sličnog eksperimenta kao prethodno navedenog, ali s metodom RandAugment. U ovome slučaju također dobivamo znatno povećanje točnosti modela, ali ovaj put nešto manje od 10%. Pokazalo se kako metoda AutoAugment očekivano daje bolje rezultate, ali s obzirom na jednostavnost metode RandAugment i ne toliko velike razlike, i ova metoda se pokazala kao odlična i osigurava povećanje točnosti modela.

Sljedeći provedeni eksperimenti uključivali su vrednovanje ovih metoda, ali s manjim početnim skupom podataka. Iz skupa za učenje, nasumično je odabrana manja količina primjera te je provedeno učenje i evaluacija modela. U tablici 4.3 prikazani su rezultati za metodu RandAugment sa skupom označenih podataka veličine 5000 primjera. Iz rezultata je vidljivo kako je bez ikakvih metoda rastresanja model posti-

| Broj označenih primjera | Broj operacija rastresanja | Ukupni broj primjera za učenje | Epohe | Točnost |
|---|---|---|---|---|
| 5000 | 0 (baseline) | 5000 | 25 | **0.6004** |
| 5000 | 4 | 25 000 | 25 | 0.5593 |
| 5000 | 9 | 50 000 | 25 | 0.5685 |

**Tablica 4.3:** RandAugment, prikaz točnosti s umanjenim skupom podataka (5000 primjera).

| Broj označenih primjera | Broj operacija rastresanja | Ukupni broj primjera za učenje | Epohe | Točnost |
|---|---|---|---|---|
| 20 000 | 0 (baseline) | 20 000 | 50 | 0.6120 |
| 20 000 | 1 | 40 000 | 50 | **0.6464** |
| 20 000 | 2 | 60 000 | 50 | 0.5500 |

**Tablica 4.4:** RandAugment, prikaz točnosti s umanjenim skupom podataka (20 000 primjera).

gao najveću točnost. Ovo možemo pripisati premalom skupu primjera. Naime, ispitni skup u ovome slučaju je veći te to povlači sa sobom činjenicu da skup za učenje nije reprezentativan, a rastresanjem takvih podataka može doći do pogoršanja rezultata, kao što je to i vidljivo ovdje. Druga stvar koja je mogla pridonijeti lošijim rezultatima je premali broj epoha te model nije stigao konvergirati.

U tablici 4.4 prikazani su rezultati s metodom RandAugment i početnim označenim skupom podataka veličine 20 000 primjera. U ovome eksperimentu, broj epoha pri učenju povećan je na 50. Iz tablice je vidljivo da rezultati variraju, vjerojatno zbog neosigurane razmjerne podjele broja primjera po svakom razredu. Ipak, vidiljivo je poboljšanje za više od 3% u odnosu na *baseline*.

Sličan eksperiment proveden je i za metodu AutoAugment, a rezultati su prikazani u tablici 4.5. Ovdje se vidi kako metoda AutoAugment donosi bolje rezultate od metode RandAugment. Poboljšanje od 8% se ističe kao odličan rezultat s obzirom na umanjeni skup podataka.

Posljednji eksperiment, korištenje zrcaljenja i dodavanje Gaussovog šuma kao metode rastresanja, vidljiv je u tablici 4.6. Ova metoda pokazala se kao dobar način za poboljšati točnost modela za nekoliko postotaka.

Rezultati eksperimenata vezanih za metode rastresanja za klasifikaciju pokazali su, u konačnici, kako metoda AutoAugment mnogo pomaže pri poboljšavanju rezultata

| Broj označenih primjera | Broj operacija rastresanja | Ukupni broj primjera za učenje | Epohe | Točnost |
|---|---|---|---|---|
| 20 000 | 0 (baseline) | 20 000 | 50 | 0.6120 |
| 20 000 | 1 | 40 000 | 50 | **0.6918** |
| 20 000 | 2 | 60 000 | 50 | 0.6524 |

**Tablica 4.5:** AutoAugment, prikaz točnosti s umanjenim skupom podataka (20 000 primjera).

| Broj označenih primjera | Operacije rastresanja | Ukupni broj primjera za učenje | Epohe | Točnost |
|---|---|---|---|---|
| 45 000 | - (baseline) | 45 000 | 25 | 0.7002 |
| 45 000 | Gaussov šum + zrcaljenje | 135 000 | 25 | 0.7325 |

**Tablica 4.6:** Eksperiment sa zrcaljenjem i dodavanjem Gaussovog šuma.

te skoro uvijek garantira zamjetno poboljšanje. Također, RandAugment metoda se pokazala kao dostojna zamjena s obzirom da je računski puno manje zahtjevna, a daje solidne rezultate.

## 4.2. Semantička segmentacija

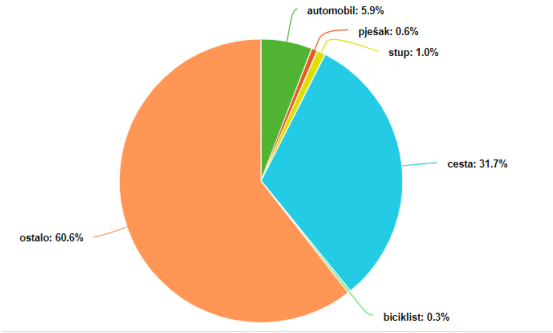
### 4.2.1. CamVid

CamVid[2] (The Cambridge-driving Labeled Video Database) je originalno prvi skup video podataka za semantičku segmentaciju.[3] Skup podataka dolazi s oznakama pripadnosti svakog piksela jednome od 32 razreda. CamVid podaci prikupljeni su snimanjem okoline direktno s automobila koji vozi kroz grad. S frekvencijom 1Hz dostupni su označeni okviri videa te je dobiveno preko 700 označenih slika koji se mogu koristiti u svrhe učenja modela za semantičku segmentaciju, poglavito za područje autonomne vožnje.

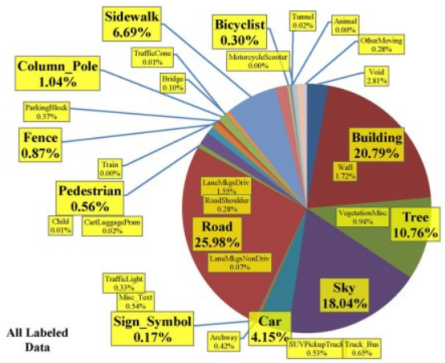
Iako su u problemima autonomne vožnje svi razredi podjednako bitni, u ovome radu naglasak je na evaluiranju metoda rastresanja pa je broj razreda za potrebe eksperimenata smanjen. U obzir su uzeti razredi *automobil, pješak, stup, cesta i biciklist* dok su svi ostali razredi spojeni u jedan - *ostalo*. Na slici 4.4 prikazan je odnos broja piksela u svakome od navedenih razreda u skupu za učenje koji se sastoji od 367 slika i pripadajućih mapi oznaka. Iz ove podjele vidljivo je da u izabranom podskupu razreda, ne gledajući *neoznačene* piksele, najveći udio odnosi cesta i zatim auto. Ako pogledamo stvarnu podjelu na slici 4.5, vidimo da ostali razredi uglavnom ne nose neki veliki udio osim neba i zgrade pa možemo reći da je naš skup i dalje reprezentativan i može poslužiti za evaluaciju metoda rastresanja.

Daljnja priprema podataka uključivala je prebacivanje oznaka piksela razreda koje ne koristimo u posebnu oznaku za razred *ostalo*. Na slici 4.3 prikazan je primjer kako izgledaju oznake za pojedine razrede. Vidljivo je kako su onačeni samo pikseli koji pripadaju određenom objektu, primjerice autu, pješaku i dr.


**Slika 4.3:** Primjer slike i pripadajućih označenih piksela za svaki razred.



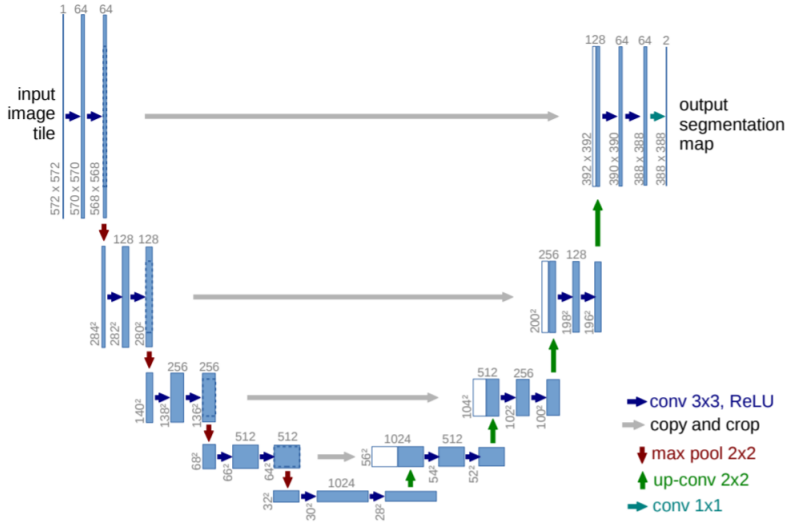

**Slika 4.4:** Prikaz odnosa broja piksela u odabranim razredima u skupu za učenje.




**Slika 4.5:** Prikaz odnosa broja piksela u svakom razredu cijelog skupa podataka.[7]

### 4.2.2. Implementacija modela i tijek učenja

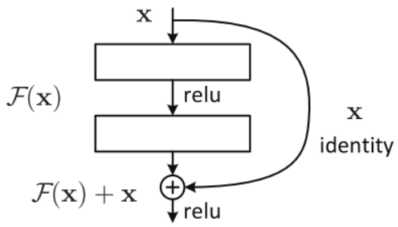
Za potrebe rješavanja ovog problema odabrana je arhitektura U-Net. Ova arhitektura pokazala se dobrim rješenjem za problem semantičke segmentacije biomedicinskih slika. Tradicionalni konvolucijski modeli u ovoj su arhitekturi modificirani tako da umjesto slojeva sažimanja imamo slojevima naduzorkovanja (engl. *upsampling*).[17] Cijela arhitekutura zamišljena je kao kombinacija enkodera i dekodera. Enkoder uči semantičke reprezentacije, dok dekoder iz izlaza enkodera stvara novu, semantički segmentiranu sliku. Grafička reprezentacija ove arhitekture vidljiva je na slici 4.6.




**Slika 4.6:** Prikaz arhitekture U-Net.[17]

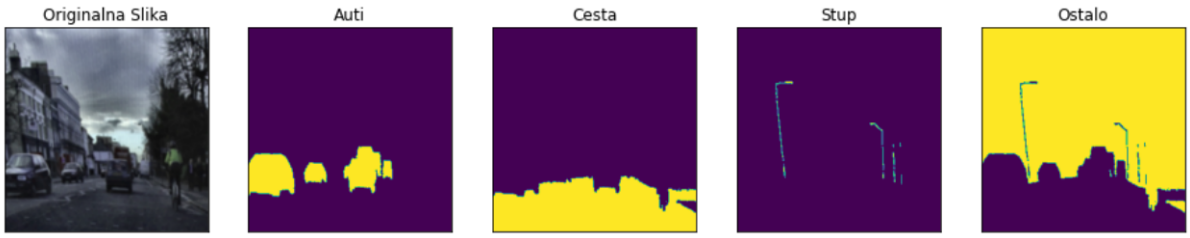
Kako bi se ovaj model prilagodio problemu semantičke segmentacije slika za autonomnu vožnju, s obzirom na razliku u problemu, enkoderski dio je zamjenjen arhitekturom ResNet50 s predtreniranim težinama na skupu podataka ImageNet. Ovo je uobičajena praksa jer rezultira bržom konvergencijom te boljim performansama modela. Kažemo kako je kralježnica (engl. *backbone*) modela ResNet50. ResNet (Residual Network) duboki je model koji koristi tzv. rezidualne blokove.[10] Ova arhitektura pojavila se kao odgovor na problem sve dubljih i dubljih konvolucijskih modela. Kao posljedica dubine modela javlja se nestajući gradijent te dolazi do zasićenja u performansama modela ili čak dovodi do degradacije performansi. Glavna ideja koju ResNet arhitektura uvodi su *skip* poveznice koje povezuju dva sloja u mreži koji nisu jedan uz drugoga. Veza tako preskače (engl. *skip*) slojeve kao što je prikazano na slici 4.7. Kao što samo ime govori, arhitektura koja je iskorištena kao osnova modela duboka je 50 slojeva. Valja napomenuti kako postoje i dublje inačice ove arhitekture sa 100 ili čak i 150 slojeva dubine. S obzirom da je problem semantičke segmentacije znatno složeniji

od problema klasifikacije slika, potrebna je i dublja arhitektura te je stoga ResNet bio logičan odabir.




**Slika 4.7:** Prikaz preskočne poveznice u arhitekturi ResNet.

Za potrebe rastresanje podataka iskorištena je knjižnica Albumentations.[4] Ova knjižnica omogućava brzo izvođenje operacija za rastresanje te se često koristi u području strojnog i dubokog učenja. U sklopu ove knjižnice moguće je primjeniti neku operaciju na sliku, a zatim i na mapu oznaka za semantičku segmentaciju. U eksperimentima su korištene razne transformacije kako bi se povećao skup za učenje: zrcaljenje, afine transformacije, manipulacije bojama, svjetlinom i kontrastom, zamućivanje i izoštravanje slika, dodavanje Gaussovog šuma i slučajno izrezivanje. Na sliku nisu uvijek primjenjivane sve operacije već nasumičnim odabirom dolazimo do nekoliko operacija. Primjer slike nakon operacija rastresanja i pripadajuće oznake prikazani su na slici 4.8.




**Slika 4.8:** Slika nakon operacija rastresanja i pripadne oznake.

Kao funkcija gubitka, odabrana je kombinacija dvaju funkcija. Prva funkcija je *Dice* gubitak koji je baziran na Dice koeficijentu. Taj koeficijent dan je formulom:

$$D = \frac{2\sum_i^N p_i g_i}{\sum_i^N p_i^2 + \sum_i^N g_i^2} \tag{4.3}$$

gdje $p$ označava predikciju, a $g$ pravi razred.

Cilj je ovu funkciju maksimizirati, odnosno funkcija gubitka je obrnuto proporcionalna *Dice* koeficijentu. Druga funkcija koja sačinjava funkciju pogreške je fokalni

gubitak (engl. *Focal loss*) koji težinama koje pridodajemo pojedinim razredima određuje koji su razredi bitniji i na taj način se pokušava doći do boljeg rješenja. Obje ove funkcije imaju svojstvo da dobro funkcioniraju za probleme gdje postoji disbalans što se tiče broja primjera u razredima. Kao što smo to prikazali na slici 4.4, upravo takva svojstva su potrebna u ovome problemu. Ova funkcija dana je formulom:

$$FL(p_t) = \begin{cases} -\alpha(1-p)^{\gamma} log(p), & y = 1. \\ -(1-\alpha)p^{\gamma} log(1-p), & \text{inače.} \end{cases} \tag{4.4}$$

gdje su $\alpha$ i $\gamma$ hiperparametri i mogu se podešavati.

Kada smo definirali ciljni funkciju, odabran je i optimizacijski algoritam. I u ovome slučaju je to Adam. Za praćenje učenja iskoristili smo mjeru srednje vrijednosti IoU, odnosno mIoU te F1 vrijednost. Mjera IoU daje nam dobar uvid u performanse modela za semantičku segmentaciju. Ova mjera (*Intersection over Union*) dana je relacijom:

$$IoU = \frac{TP}{TP + FN + FP} \tag{4.5}$$

gdje TP predstavlja točno klasificirane piksele, FN predstavlja piksele klasificirane da ne pripadaju razredu u pitanju, a zapravo pripadaju. FP predstavlja piksele koji su klasificirani u razred u pitanju, a istinski ne pripadaju. Srednja vrijednost, odnosno mIoU izračunava se kao srednja vrijednost za sve razrede u skupu podataka i dobar je pokazatelj performansi kod semantičke segmentacije.

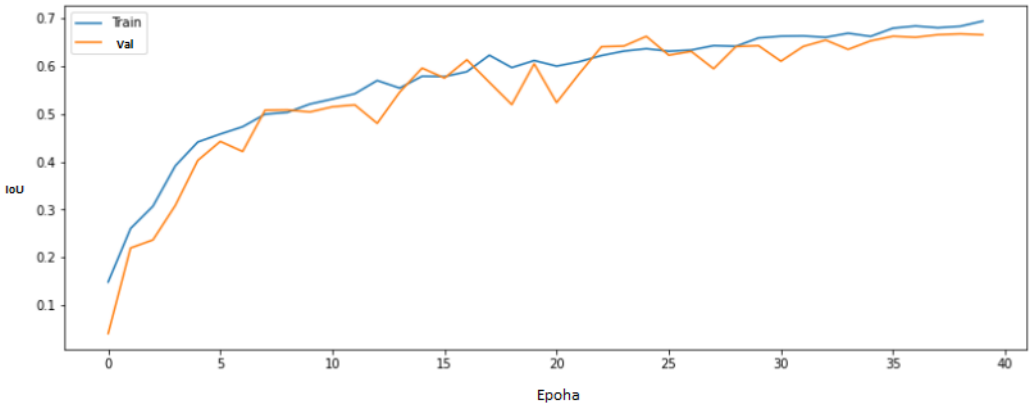
Učenje modela odvijalo se u 40 epoha. Na slici 4.9 prikazan je tijek učenja, odnosno rast IoU mjere na skupu za učenje te na skupu za validaciju. Na slici 4.10 vidiljiv je isti tijek učenja, ali s korištenjem operacija za rastresanje podataka. Već je iz ovih slika vidljivo poboljšanje u odnosu na *baseline*.

Konačni podaci za usporedbu prikazani su u tablici 4.7. Možemo zaključiti kako korištenje operacija za rastresanje poboljšava performanse modela.

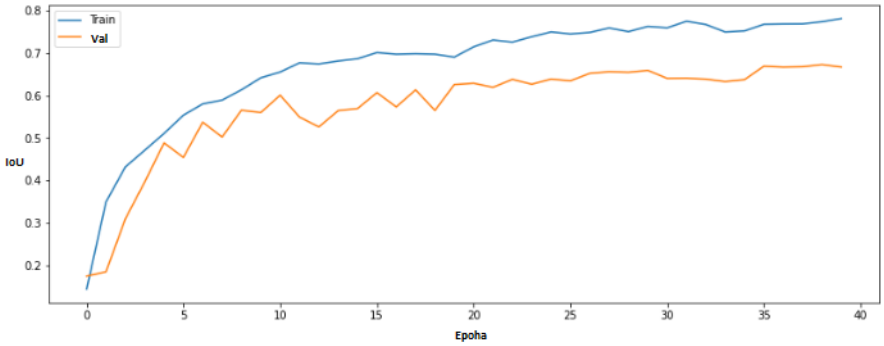
| Model | mIoU | F1 |
|---|---|---|
| baseline | 0.6198 | 0.6982 |
| S operacijama za rastresanje | **0.6346** | **0.7033** |

**Tablica 4.7:** Prikaz performansi modela s i bez operacija za rastresanje podataka.

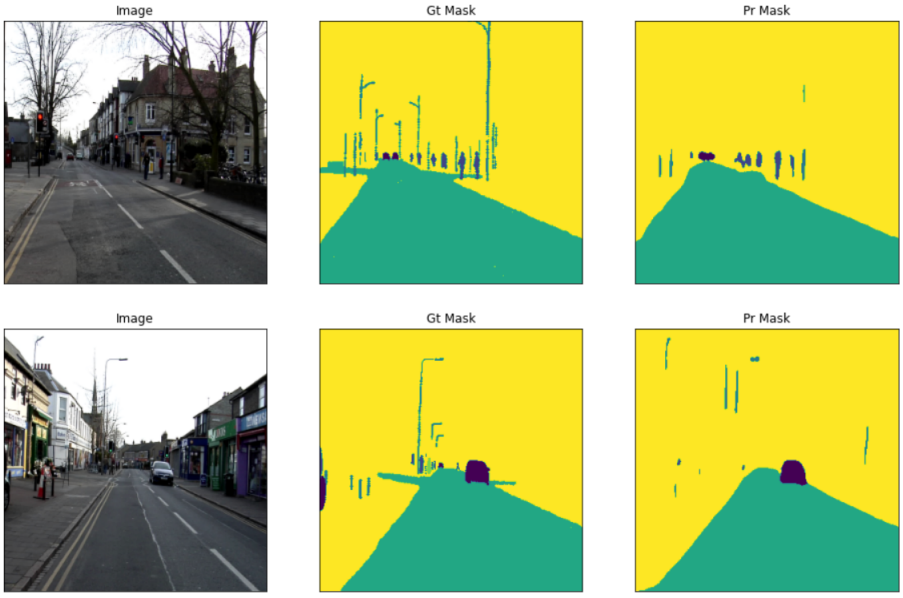
Kako bismo bolje prikazali performanse modela, na slici 4.11su prikazane slike, njihove istinske oznake i predikcije koje je dao model. Vidljivo je da postoje diskrepancije između predikcija i stvarnih oznaka, a to se odražava i na mIoU rezultatu.




**Slika 4.9:** Prikaz rasta mjere IoU tijekom učenja modela bez operacija rastresanja.




**Slika 4.10:** Prikaz rasta mjere IoU tijekom učenja modela s operacijama rastresanja.




**Slika 4.11:** Vizualna usporedba predikcije modela i stvarnih oznaka slika.

Za poboljšanje rezultata, trebalo bi učiti model na više epoha uz možda još veći broj operacija rastresanja. U svakome slučaju, korištenje ovih operacija pokazalo se kao dobra praksa te je poboljšalo performanse modela za nekoliko postotaka.

# 5. Zaključak

Izvođenjem eksperimenata u ovome radu pokazalo se kako je rastresanje podataka dobra alternativa dodatnome sakupljanju i označavanju novih podataka. Prednosti ovih metoda evidentne su - smanjenje troškova i cijene projekta, vremenska ušteda i poboljšanje performansi modela. Iako ne postoji savršena metoda koja će objediniti sve prednosti metoda opisanih u ovome radu, dolazimo do zaključka kako se isplati uložiti malo dodatnog vremena kako bi se odabrala povoljna metoda za rastresanje skupa podataka koji se koristi pri učenju. Metoda AutoAugment pokazala je kako učenje povoljnih transformacija za točno određeni skup podataka ima velike prednosti. Pristupanje svakome skupu podataka na svojstven način ipak je praksa koja nije moguća u svakoj situaciji. Prepreke na koje ova metoda nailazi su povećana računska složenost te nemogućnost regularizacije modela. Stoga je metoda AutoAugment primjenjiva na manje skupove podataka. Metoda RandAugment stoga se pokazala boljom jer je primjenjiva na velike skupove podataka, a zbog formulacije metode moguće je mijenjanjem parametara utjecati na regularizaciju.[6] U konačnici, ako gledamo na problem rastresanja podataka iz perspektive široke primjene, RandAugment se nameće kao bolja metoda što možemo potkrijepiti *state-of-the-art* rezultatima čak i u polunadziranom okruženju.[22] Valja napomenuti kako konačni utjecaj ovih metoda na performanse modela i dalje uvelike ovisi o dubini modela i o veličini početnog skupa podataka i njihovoj raznolikosti i kvaliteti. Korištenjem metoda za rastresanje podataka uvodimo pristranost i dodatni šum u podatke te je potrebno podesiti regularizacijske hiperparametre modela kako bi se došlo do boljih rezultata.

Za probleme semantičke segmentacija problem rastresanja podataka i dalje je djelomično nerješen. Naime, prepreku stvara upravo priroda problema, odnosno činjenica za nam je potrebna oznaka za svaki piksel na slici. Metode temeljene na GAN mrežama spomenute u poglavlju 3.4 prihvatljive su u situacijama kada je moguće odvojiti dodatno vrijeme i računalne resurse kako bi se te mreže naučile da generiraju kvalitetne podatke. I u ovome slučaju to su ponekad nepremostive prepreke. Pri učenju modela za semantičku segmentaciju tako se dakle i dalje najviše koriste klasične me-

tode za rastresanje slika koje se primjenjuju identično na samu sliku i mapu oznaka te slike. Eksperimenti provedeni u ovome radu pokazali su poboljšanje performansi modela kada se koriste metode rastresanja. Kao dodatno poboljšanje moguće je iskoristiti metodu RandAugment u problemu semantičke segmentacije prilagođenu za sam problem.

U konačnici, ovo područje istraživanja i dalje ima puno prostora za napredak. Uvijek će postojati potreba za povećanjem skupa podataka pa je iz tog razloga ovo vrlo zanimljivo područje. Smjer u kojem se kreće ovo područje je mogućnost ugrađivanja ovih metoda u polunadzirane ili čak nenadzirane okoline te iskorištavanje potencijala napredaka u području dubokih konvolucijskih modela.

# LITERATURA


[1] Antreas Antoniou, Amos Storkey, i Harrison Edwards. Data augmentation generative adversarial networks, 2017.

[2] Gabriel J. Brostow, Julien Fauqueur, i Roberto Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, xx(x):xx–xx, 2008.

[3] Gabriel J. Brostow, Jamie Shotton, Julien Fauqueur, i Roberto Cipolla. Segmentation and recognition using structure from motion point clouds. U *ECCV (1)*, stranice 44–57, 2008.

[4] Alexander Buslaev, Vladimir I. Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin, i Alexandr A. Kalinin. Albumentations: Fast and flexible image augmentations. *Information*, 11(2), 2020. ISSN 2078-2489. doi: 10.3390/info11020125. URL https://www.mdpi.com/2078-2489/11/2/125.

[5] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, i Quoc V. Le. Autoaugment: Learning augmentation policies from data, 2018.

[6] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, i Quoc V. Le. Randaugment: Practical automated data augmentation with a reduced search space, 2019.

[7] Cambridge Video Database. Class share across all labeled data: image. URL http://mi.eng.cam.ac.uk/research/projects/VideoRec/CamVid/.

[8] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, i Yoshua Bengio. Generative adversarial networks, 2014.

[9] Benjamin Graham. Fractional max-pooling. *CoRR*, abs/1412.6071, 2014. URL http://arxiv.org/abs/1412.6071.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, i Jian Sun. Deep residual learning for image recognition, 2015.

[11] Diederik P. Kingma i Jimmy Ba. Adam: A method for stochastic optimization, 2014. URL http://arxiv.org/abs/1412.6980.

[12] Damir Kopljar. Konvolucijske neuronske mreže. 2016. URL http://ferko.fer.hr/ferko/EPortfolio!dlFile.action?id=350.

[13] Alex Krizhevsky. The CIFAR-10 dataset. URL https://www.cs.toronto.edu/~kriz/cifar.html.

[14] Shuangting Liu, Jiaqi Zhang, Yuxin Chen, Yifan Liu, Zengchang Qin, i Tao Wan. Pixel level data augmentation for semantic image segmentation using generative adversarial networks. *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2019. doi: 10.1109/icassp.2019.8683590. URL http://dx.doi.org/10.1109/ICASSP.2019.8683590.

[15] Mark Mcdonnell, Migel D Tissera, Tony Vladusich, André van Schaik, i Jonathan Tapson. Fast, simple and accurate handwritten digit classification by training shallow neural network classifiers with the 'extreme learning machine' algorithm. 10:e0134254, 08 2015.

[16] A. Mikołajczyk i M. Grochowski. Data augmentation for improving deep learning in image classification problem. U *2018 International Interdisciplinary PhD Workshop (IIPhDW)*, stranice 117–122, 2018.

[17] Olaf Ronneberger, Philipp Fischer, i Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, stranica 234–241, 2015. ISSN 1611-3349. doi: 10.1007/978-3-319-24574-4_28. URL http://dx.doi.org/10.1007/978-3-319-24574-4_28.

[18] Sebastian Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016. URL http://arxiv.org/abs/1609.04747.

[19] T. Tieleman i G. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

[20] Marko Ćupić. Duboko učenje: Optimizacija parametara modela., 2019. URL http://www.zemris.fer.hr/~ssegvic/du/du3optimization.pdf.

[21] Yulin Wang, Xuran Pan, Shiji Song, Hong Zhang, Cheng Wu, i Gao Huang. Implicit semantic data augmentation for deep networks, 2019.

[22] Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, i Quoc V Le. Unsupervised data augmentation for consistency training. *arXiv preprint arXiv:1904.12848*, 2019.

## Vrednovanje metoda rastresanja podataka za konvolucijske modele


### Sažetak

Duboki konvolucijski modeli su glavni sastojak mnogih praktičnih primjena računalnog vida. Međutim, takvi modeli su ponekad nepraktični jer zahtijevaju velike skupove označenih podataka za učenje. Zbog toga postupci za rastresanje podataka predstavljaju zanimljivo područje istraživanja. U okviru ovog rada proučene su i implementirane metode za rastresanje podataka. Provedeni su eksperimenti i svaka od metoda je vrednovana i uspoređena s rezultatima iz literature.



**Ključne riječi:** rastresanje podataka, duboko učenje, konvolucijski modeli, klasifikacija, semantička segmentacija, cifar-10, camvid


## Evaluation of jittering techniques for convolutional models


### Abstract

Deep convolutional models are the main instrument for many practical applications in the field of computer vision. However, these models are sometimes unpractical because they require great amounts of training data. Therefore, jittering and augmentation techniques impose themselves as interesting filed for reasearch. In this thesis, some of the recent techniques were researched and implemented. Experiments were conducted and each technique was evaluated and compared to results from the literature.



**Keywords:** data augmentation, deep learning, convolutional models, classification, semantic segmentation, cifar-10, camvid