

VELEUČILIŠTE U BJELOVARU
STRUČNI PRIJEDIPLOMSKI STUDIJ RAČUNARSTVO

**KONVOLUCIJSKE NEURONSKE MREŽE ZA
PREPOZNAVANJE VOZILA: ANALIZA I USPOREDBA
MODELA**

Završni rad br. 13/RAČ/2024

Antonio Jungić

Bjelovar, listopad 2024.



Veleučilište u Bjelovaru
Trg E. Kvaternika 4, Bjelovar

1. DEFINIRANJE TEME ZAVRŠNOG RADA I POVJERENSTVA

Student: **Antonio Jungić**

JMBAG: 0314024758

Naslov rada (tema): **Konvolucijske neuronske mreže za prepoznavanje vozila: Analiza i usporedba modela**

Područje: **Tehničke znanosti**

Polje: **Računarstvo**

Grana: **Umjetna inteligencija**

Mentor: **Krešimir Markota, mag. ing. comp.**

zvanje: **predavač**

Članovi Povjerenstva za ocjenjivanje i obranu završnog rada:

1. **Krunoslav Husak, dipl. ing. rač., predsjednik**
2. **Krešimir Markota, mag. ing. comp., mentor**
3. **Ivan Sekovanić, mag. ing. inf. et comm. techn., član**

2. ZADATAK ZAVRŠNOG RADA BROJ: 13/RAČ/2024

U sklopu završnog rada potrebno je:

1. Analizirati važnost prepoznavanja vozila za sigurnost i učinkovitost prometa te istražiti njegova područja primjene.
2. Predobraditi podatkovni skup slika s vozilima u prikladan format za treniranje i testiranje konvolucijske neuronske mreže pomoću Python biblioteka za obradu slike.
3. Izraditi prilagođen model konvolucijske neuronske mreže koja će se trenirati na predobrađenom podatkovnom skupu za prepoznavanje vozila.
4. Usporediti performanse prilagođenog modela konvolucijske neuronske mreže za prepoznavanje vozila s vodećim predefiniranim modelima.
5. Evaluirati i interpretirati matrice zabune za odabrane modele konvolucijske neuronske mreže za prepoznavanje vozila.

Datum: 13. rujna 2024. godine

Mentor: **Krešimir Markota, mag. ing. comp.**



Sadržaj

1. UVOD	1
2. STROJNO UČENJE.....	2
2.1 Nadzirano strojno učenje.....	2
2.2 Nenadzirano strojno učenje.....	3
2.3 Podržano strojno učenje.....	4
3. NEURONSKE MREŽE.....	5
3.1 Konvolucijska neuronska mreža	5
4. KORIŠTENE TEHNOLOGIJE	7
4.1 Python	7
4.2 Tensorflow.....	7
4.3 Keras.....	7
4.4 Google Colab.....	8
4.5 Python biblioteke	8
4.5.1 Pandas	8
4.5.2 NumPy	8
4.5.3 Matplotlib.....	8
4.5.4 Scikit-Learn.....	9
4.5.5 Time.....	9
4.6 Skup podataka.....	9
5. KORIŠTENI MODELI	10
5.1 DenseNet121 model	10
5.1.1 Kreiranje modela.....	10
5.1.2 Treniranje modela	11
5.2 ResNet50 model	12
5.2.1 Kreiranje modela.....	12
5.2.2 Treniranje modela	13
5.3 Custom CNN model.....	13
5.3.1 Kreiranje modela.....	13
5.3.2 Treniranje modela	15
6. TESTIRANJE I REZULTATI.....	16
6.1 Trajanje treniranja.....	16
6.2 Funkcija gubitka	16
6.3 Funkcija točnosti.....	18
6.4 Matrica konfuzije	19
6.5 Evaluacijske metrike	20
7. ZAKLJUČAK.....	23
8. LITERATURA	24
9. OZNAKE I KRATICE	25

10. SAŽETAK.....	26
11. ABSTRACT	27

1. UVOD

Svrha ovog završnog rada je istražiti i analizirati primjenu dubokih neuronskih mreža u zadatku klasifikacije slika. U fokusu rada nalazi se usporedba performansi triju različitih modela konvolucijskih neuronskih mreža: Custom CNN, ResNet50 i DenseNet121. S obzirom na široku primjenu neuronskih mreža u prepoznavanju uzoraka i klasifikaciji slika, ovaj rad nastoji detaljno obraditi proces obrade podataka, treniranja modela i analize njihovih rezultata na CIFAR-10 skupu podataka.

Prema dosadašnjim istraživanjima, duboke neuronske mreže poput ResNet-a i DenseNet-a pokazale su se izuzetno učinkovitima u zadatku klasifikacije slika jer omogućuju prepoznavanje složenih uzoraka i bolju generalizaciju na neviđene podatke. S druge strane, jednostavniji modeli poput Custom CNN-a mogu ponuditi zadovoljavajuće rezultate uz kraće trajanje treniranja i manji broj parametara, što ih čini prikladnim za određene zadatke. Ovaj rad pruža detaljnu usporedbu ovih modela s ciljem razumijevanja njihovih prednosti i nedostataka u kontekstu klasifikacije slika.

U svrhu evaluacije modela korištene su ključne metode strojnog učenja, uključujući obradu podataka, normalizaciju, augmentaciju i treniranje neuronskih mreža. Modeli su evaluirani korištenjem metričkih funkcija poput preciznosti, odaziva, F1-ocjene te matrice konfuzije kako bi se detaljnije analizirale njihove performanse na CIFAR-10 skupu podataka.

Struktura rada je sljedeća: nakon uvoda, poglavlje "Korišteni modeli" detaljno opisuje arhitekturu i implementaciju triju modela, te postupak obrade podataka prije treniranja. Poglavlje "Testiranje i rezultati" prikazuje rezultate evaluacije modela kroz metrike poput funkcije gubitka, točnosti i vremena treniranja. Rad završava zaključkom, gdje su predstavljeni ključni rezultati istraživanja.

2. STROJNO UČENJE

Strojno učenje predstavlja granu umjetne inteligencije koja koristi podatke i algoritme kako bi omogućila sustavima oponašanje ljudskog procesa učenja te postupno poboljšanje točnosti. Prema izvoru [1], ova tehnologija omogućuje računalima da uče iz iskustva bez ručnog programiranja, koristeći podatke poput slika ili senzorskih informacija za treniranje modela koji prepoznaju obrasce i daju predikcije. Korištenje većeg broja podataka povećava točnost modela, dok se dio podataka koristi za provjeru performansi na novim ulazima. Sustav može imati opisne funkcije koje analiziraju prošle događaje. Također, funkcije mogu biti prediktivne, s ciljem predviđanja budućih događaja. Uz to, sustav može imati i preskriptivne funkcije koje daju preporuke za daljnje korake. Tri glavne vrste strojnog učenja uključuju nadzirano, nenadzirano i podržano učenje, prikazane su na slici 2.1.



Slika 2.1: Podjela strojnog učenja [2]

2.1 Nadzirano strojno učenje

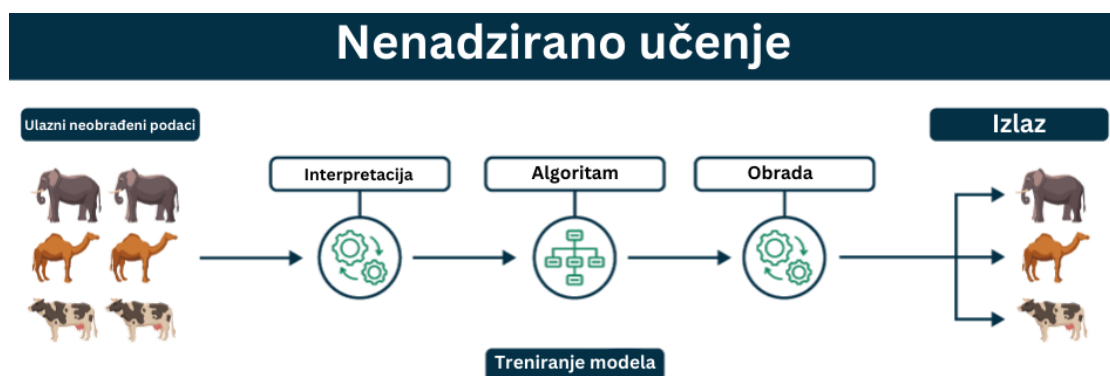
Nadzirano strojno učenje koristi označene podatke za treniranje modela, kako bi se naučili povezivati ulazni podaci s odgovarajućim izlazima. Na slici 2.2 prikazan je osnovni koncept nadziranog učenja. Primjerice, treniranje modela na slikama mačaka i pasa omogućuje razlikovanje tih dviju vrsta. Dvije glavne kategorije uključuju klasifikaciju, koja predviđa diskretne kategorije, kao što je prepoznavanje e-mailova kao neželjenu ili željenu poštu, i regresiju, koja predviđa kontinuirane vrijednosti, poput cijena kuća. Prednosti uključuju visoku preciznost i interpretabilnost modela, dok je nedostatak potreba za velikim količinama označenih podataka. Primjene nadziranog učenja obuhvaćaju prepoznavanje slika, dijagnostiku bolesti i sustave preporuka.



Slika 2.2: Nadzirano strojno učenje [3]

2.2 Nenadzirano strojno učenje

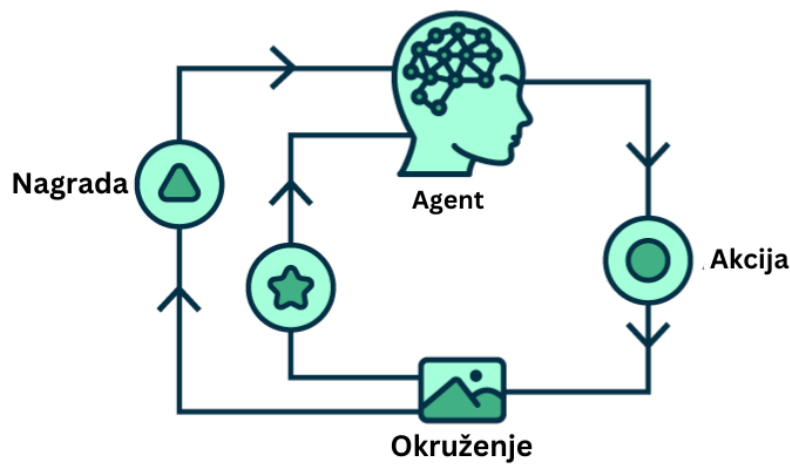
Nenadzirano strojno učenje koristi neoznačene podatke za otkrivanje skrivenih obrazaca ili odnosa u podacima. Klasteriranje, odnosno grupiranje sličnih podataka, i asocijacijsko učenje, koje otkriva povezanosti između stavki, glavne su metode nenadziranog učenja. Prednosti uključuju automatsko otkrivanje obrazaca, mogućnost rada s velikim količinama neoznačenih podataka i smanjenu potrebu za označavanjem podataka. Također, nenadzirano učenje može identificirati neočekivane strukture i odnose u podacima. Međutim, izazov ostaje interpretacija rezultata. Primjene uključuju segmentaciju tržišta, prepoznavanje anomalija i preporučiteljske sustave. Na slici 2.3 prikazan je osnovni koncept nenadziranog učenja.



Slika 2.3: Nenadzirano strojno učenje [3]

2.3 Podržano strojno učenje

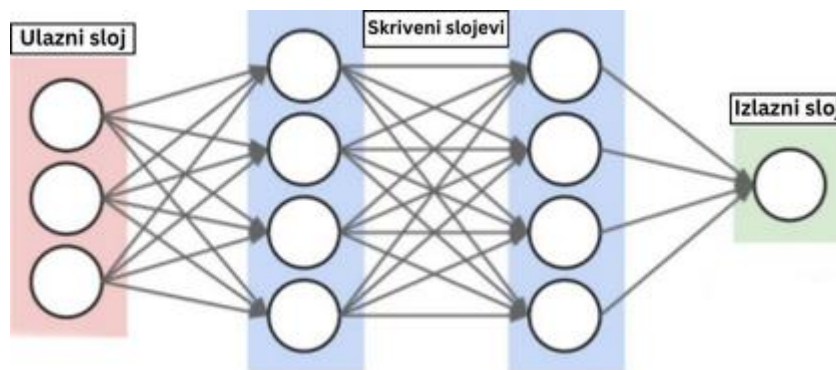
Podržano strojno učenje koristi povratne informacije iz okoline kako bi algoritam poboljšao svoje odluke. Kao što je prikazano na slici 2.4, temeljni principi uključuju pokušaj, pogrešku i odgođenu nagradu. Algoritmi poput Q-learninga, SARSA-e i Deep Q-learninga uče odnose između stanja i akcija kroz povratne informacije. Prema [3], podržano učenje može biti pozitivno, što znači nagrada za poželjnu akciju, ili negativno, što podrazumijeva uklanjanje neugodnog podražaja. Prednosti uključuju autonomno donošenje odluka i rješavanje složenih problema, dok su glavni nedostaci visoki troškovi treninga i potreba za velikim količinama podataka. Primjene uključuju autonomna vozila, robotiku i računalne igre.



Slika 2.4: Podržano strojno učenje [3]

3. NEURONSKE MREŽE

Neuronske mreže, inspirirane radom ljudskog mozga, koriste se za automatsko izdvajanje značajki iz podataka, kao što se navodi u izvoru [3]. Osnovni dijelovi mreže uključuju ulazne podatke, težine, pristranosti, aktivacijske funkcije i pravilo učenja. Mreža iterativno prilagođava svoje parametre tijekom tri faze: primanje ulaza, generiranje izlaza i prilagođavanje. Neuronska mreža sastoji se od ulaznog sloja, jednog ili više skrivenih slojeva, i izlaznog sloja. Struktura neuronske mreže prikazana je na slici 3.1. Aktivacijske funkcije poput ReLU-a ili sigmoidne funkcije uvode nelinearnost, omogućujući prepoznavanje složenih nelinearnih uzoraka. Kvaliteta predikcija mreže mjeri se funkcijom gubitka, poput srednje kvadratne pogreške, koja uspoređuje predviđene i stvarne vrijednosti.

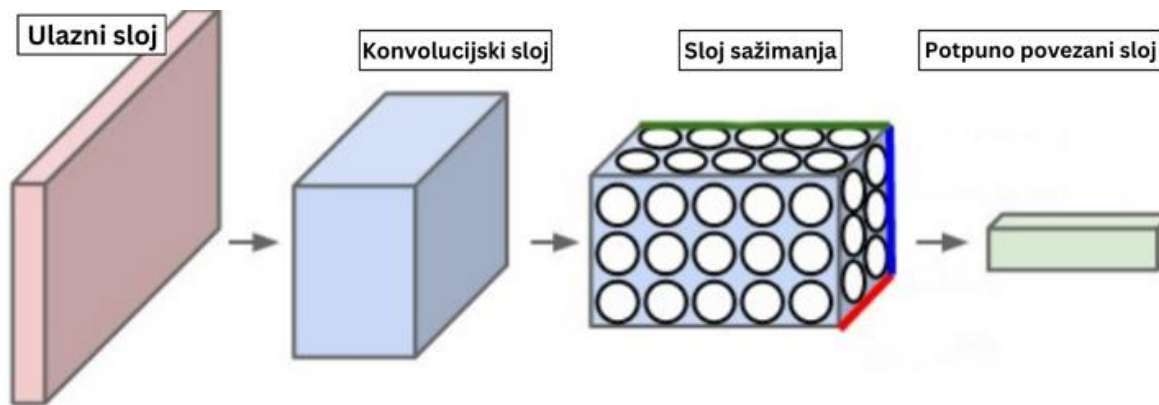


Slika 3.1: Izgled neuronske mreže [5]

Treniranje mreže uključuje fazu unaprijedne propagacije, gdje ulazi prolaze kroz mrežu, i fazu povratne propagacije, gdje se greške koriste za prilagodbu težina. Gradijentni spust prilagođava težine kako bi se smanjila ukupna pogreška. Postupak se ponavlja dok se ne dobije željena razina točnosti. Neuronske mreže primjenjuju se u različitim područjima, uključujući prepoznavanje slika, obradu prirodnog jezika, autonomna vozila i modele predviđanja. Zahvaljujući sposobnosti učenja iz velikih količina podataka, postale su ključne u rješavanju složenih problema koje tradicionalni algoritmi ne mogu učinkovito riješiti.

3.1 Konvolucijska neuronska mreža

Konvolucijske neuronske mreže, prema [6], posebno su učinkovite za prepoznavanje i obradu slika. Njihova arhitektura vidljiva je na slici 3.2, a sastoji se od konvolucijskih slojeva, slojeva sažimanja i potpuno povezanih slojeva, a dizajnirana je da oponaša način na koji mozak obrađuje vizualne informacije.



Slika 3.2: Izgled konvolucijske neuronske mreže [5]

CNN-ovi korisni su za zadatke poput klasifikacije slika, detekcije objekata i segmentacije slika. Navedene vrste neuronskih mreža koriste se u medicinskoj dijagnostici, robotici i analizi videa, osobito za praćenje objekata i prepoznavanje obrazaca. Konvolucijski slojevi koriste filtere za otkrivanje značajki poput rubova i tekstura, dok slojevi sažimanja smanjuju prostorne dimenzije ulaza, čime se smanjuje računski trošak mreže. Treniranje CNN-a uključuje pripremu podataka, mjerenje performansi putem funkcije gubitka te prilagodbu težina pomoću optimizatora. Učinkovitost modela mjeri se metrikama poput točnosti, preciznosti, odziva i F1 mjere, osobito kod neuravnoteženih skupova podataka. Konvolucijski sloj ključan je za CNN jer primjenjuje filtere na ulazne podatke, generirajući aktivacijske mape koje predstavljaju odziv mreže na različite uzorke slike. Dubina izlaznog volumena određuje se brojem filtera, a nadopunjavanje nulama može se koristiti za očuvanje dimenzija ulaza. Slojevi sažimanja smanjuju dimenzije aktivacijskih mapa, što smanjuje složenost mreže. Najčešće se koristi maksimalno sažimanje, koje zadržava najveću vrijednost u određenom području, čime se mreža bolje nosi s promjenama u položaju značajki. Potpuno povezani slojevi dolaze na kraju mreže i povezuju svaki neuron s prethodnim slojem. Ovi slojevi omogućuju mreži da uči složenije obrasce na temelju naučenih značajki iz prethodnih slojeva, donoseći konačne predikcije.

4. KORIŠTENE TEHNOLOGIJE

Ovo poglavlje obuhvaća pregled svih tehnologija korištenih u ovom radu, uključujući programski jezik Python, biblioteke za obradu podataka te Google Colab kao platformu za treniranje modela. Također, opisan je i korišteni skup podataka CIFAR-10.

4.1 Python

Python je interpretirani programski jezik temeljen na objektno orijentiranom pristupu, poznat po jednostavnoj sintaksi i fleksibilnosti, što ga čini popularnim u različitim područjima, uključujući web razvoj, znanost o podacima i strojno učenje, prema izvoru [7]. Njegova bogata standardna biblioteka i podrška za različite alate omogućuju široku primjenu, od razvoja softverskih sustava do analize podataka. U ovom radu, Python je korišten za obradu podataka, treniranje modela i evaluaciju rezultata.

4.2 Tensorflow

Kako se navodi u izvoru [8], TensorFlow je okvir otvorenog koda za strojno učenje razvijen od Googlea, te se koristi za razvoj i treniranje modela strojnog i dubokog učenja. Integriran je s Pythonom, omogućujući jednostavne numeričke izračune i rad s velikim skupovima podataka. TensorFlow nudi dva sučelja: Keras, visoko-razinski API za brzo prototipiranje, i TensorFlow Core, niskorazinski API za veću kontrolu nad modelima. Također, TensorFlow dolazi s ugrađenim alatom TensorBoard za praćenje treniranja i analizu performansi modela. Ove značajke čine ga pogodnim za projekte strojnog učenja korištene u ovom radu.

4.3 Keras

Keras je visoko-razinski open-source API za duboko učenje, koji je, prema izvoru [9], poznat po svojoj jednostavnosti i intuitivnosti. Kao dio TensorFlow-a (tf.keras), omogućuje brzo kreiranje i treniranje neuronskih mreža s minimalnim kodom. Njegove ključne prednosti uključuju fleksibilnost, kompatibilnost s različitim platformama (GPU i CPU), te mogućnost brzog prototipiranja, što ga čini popularnim u istraživačkim projektima. Keras je korišten u širokom rasponu primjena, uključujući računalni vid, obradu prirodnog jezika i vremenske serije.

4.4 Google Colab

Google Colab besplatna je platforma na oblaku koja omogućuje pisanje i izvršavanje Python koda u Jupyter Notebook okruženju. Prema izvoru [10], koristan je za strojno učenje, jer pruža besplatan pristup GPU-ovima, što je važno za treniranje složenih modela. Ne zahtijeva instalaciju ili postavljanje, jer se sve odvija u oblaku, omogućujući brz početak rada. Integracija s Google Driveom olakšava spremanje i dijeljenje bilježnica, dok kolaborativno uređivanje omogućuje više korisnika da rade na istom projektu istovremeno. Colab je kompatibilan s popularnim bibliotekama poput TensorFlowa, PyTorch, NumPya i Matplotliba, čineći ga popularnim izborom među znanstvenicima i analitičarima.

4.5 Python biblioteke

Python biblioteke su zbirke prethodno napisanih kodova koje olakšavaju razvoj aplikacija i algoritama. Umjesto da se često korišteni dijelovi koda pišu ispočetka, koriste se unaprijed definirane funkcije i alati, čime se ubrzava proces razvoja. U strojnom učenju najčešće su korištene biblioteke poput Pandas, Matplotlib, Scikit-learn i NumPy.

4.5.1 Pandas

Pandas je open-source biblioteka koja omogućuje manipulaciju i analizu podataka, posebno tabličnih. Prema izvoru [11], ključan je alat za podatkovne analitičare i znanstvenike koji rade sa strukturiranim podacima u Pythonu, a izgrađen je na NumPy-u. Pandas olakšava čišćenje, spajanje i obradu podataka, kao i rukovanje nedostajućim vrijednostima, čime postaje nezamjenjiv u analizi podataka.

4.5.2 NumPy

NumPy je osnovna Python biblioteka za znanstvene izračune i manipulaciju velikim višedimenzionalnim nizovima, prema izvoru [12]. Osigurava alate za matematičke operacije, linearnu algebru i generiranje slučajnih brojeva, te podržava integraciju s drugim jezicima poput C i Fortrana. Njegova sposobnost brzog rada s numeričkim podacima čini ga temeljnim alatom u znanosti o podacima i strojnom učenju.

4.5.3 Matplotlib

Kako navodi izvor [13], Matplotlib je moćan alat za vizualizaciju podataka koji omogućuje izradu raznovrsnih grafova, uključujući linijske, stupčaste i histogram grafove.

Omogućuje vizualizaciju rezultata i analiza podataka, što olakšava razumijevanje kompleksnih informacija.

4.5.4 Scikit-Learn

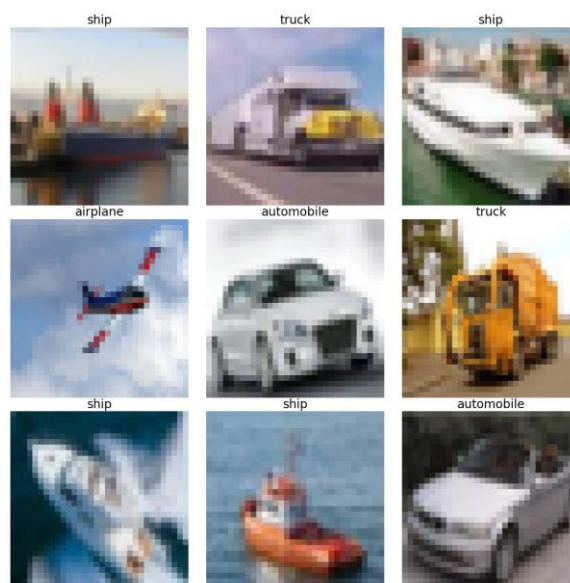
Scikit-learn je biblioteka koja nudi alate za klasifikaciju, regresiju i klasteriranje. Osim toga, prema izvoru [14], omogućuje predobradu podataka, razdvajanje podataka na trening i test setove te skaliranje značajki. Scikit-learn također nudi alate za evaluaciju modela pomoću metrika kao što su točnost i preciznost.

4.5.5 Time

Biblioteka Time, kako je navedeno u izvoru [15], omogućuje rad s vremenskim podacima, uključujući praćenje vremena i mjerenje trajanja. Korisna je u aplikacijama koje zahtijevaju precizno praćenje vremena.

4.6 Skup podataka

Za ovaj rad korišten je CIFAR-10 skup podataka, koji, prema izvoru [16], sadrži 60.000 slika u boji veličine 32x32 piksela podijeljenih u 10 klasa. U ovom slučaju, korištene su samo 4 klase: avion, automobil, brod i kamion. Trening set sadrži 20.000 slika, dok testni set sadrži 4.000 slika, ravnomjerno raspoređenih među klasama, a primjere tih slika moguće je vidjeti na slici 4.1. Ovaj podskup omogućuje efikasno treniranje modela uz zadržavanje dovoljne raznolikosti za točnu klasifikaciju.



Slika 4.1: Slike iz skupa podataka

5. KORIŠTENI MODELI

U ovom poglavlju prikazana je obrada podataka prije treniranja modela te implementacija i treniranje triju neuronskih mreža: Custom CNN, ResNet50 i DenseNet121. Svi modeli su prilagođeni za klasifikaciju slika iz CIFAR-10 skupa podataka. Dio ovog postupka i objašnjenja generiran je uz pomoć jezičnog modela ChatGPT.

Priprema podataka uključivala je:

1. Normalizacija podataka: CIFAR-10 slike imaju piksele u rasponu od 0 do 255. Te vrijednosti su podijeljene s 255 kako bi se skalirale na raspon od 0 do 1, što olakšava treniranje modela, kako je prikazano u programskom kodu 5.1.
2. One-hot kodiranje ciljne varijable: Radilo se s 4 klase, pa je svaka klasa pretvorena u vektor duljine 4, gdje je jedna pozicija označena s 1, a ostale s 0. Ovaj postupak osigurava prepoznavanje vjerojatnosti za svaku klasu umjesto numeričkih oznaka, što možemo vidjeti u programskom kodu 5.1.

Programski kod 5.1: Normalizacija podataka i one-hot kodiranje ciljne varijable

```
X_train_norm = X_train_norm / 255.0
X_test_norm = X_test_norm / 255.0

label_mapping = {0: 0, 1: 1, 8: 2, 9: 3}
y_train_converted = np.vectorize(label_mapping.get)(y_train)
y_test_converted = np.vectorize(label_mapping.get)(y_test)

y_train_onehot = to_categorical(y_train_converted, 4)
y_test_onehot = to_categorical(y_test_converted, 4)
```

5.1 DenseNet121 model

DenseNet121 je duboka konvolucijska neuronska mreža koja se često koristi za zadatke klasifikacije slika. Njegova arhitektura omogućuje međusobno dijeljenje informacija između slojeva, što smanjuje broj parametara i omogućuje učinkovitije učenje. U nastavku je opisan proces kreiranja, kompiliranja i treniranja prilagođenog DenseNet121 modela za klasifikaciju slika iz CIFAR-10 skupa podataka.

5.1.1 Kreiranje modela

Model je kreiran učitavanjem osnove DenseNet121 modela bez završnih slojeva. Iz programskog koda 5.2 možemo vidjeti da je dodan Global Average Pooling sloj kako bi se smanjio broj parametara te potpuno povezani sloj s 512 jedinica i aktivacijom ReLU. Na

kraju je dodan izlazni sloj s 4 jedinica i aktivacijom softmax, kako bi model mogao predvidjeti vjerojatnosti za svaku klasu.

Programski kod 5.2: Kreiranje DenseNet modela

```
densenet_model_base = DenseNet121(weights='imagenet',
include_top=False, input_shape=(32, 32, 3))

model_custom_densenet = Sequential()

model_custom_densenet.add(densenet_model_base)

model_custom_densenet.add(GlobalAveragePooling2D())

model_custom_densenet.add(Dense(512, activation='relu'))
model_custom_densenet.add(Dropout(0.25))

model_custom_densenet.add(Dense(4, activation='softmax'))
```

Adam optimizator, prikazan u programskom kodu 5.3, omogućuje učinkovito treniranje na velikim skupovima podataka uz prilagodljivu stopu učenja. Categorical crossentropy je odabrana kao funkcija gubitka zbog višeklasne prirode klasifikacije.

Programski kod 5.3: Kompilacija DenseNet modela

```
METRICS = [
    'accuracy',
    tf.keras.metrics.Precision(name='precision'),
    tf.keras.metrics.Recall(name='recall')
]

model_custom_densenet.compile(optimizer=Adam(learning_rate=1e-
4), loss='categorical_crossentropy', metrics=METRICS)
```

5.1.2 Treniranje modela

Iz programskog koda 5.4 možemo vidjeti da je treniranje modela provedeno tijekom 20 epoha, koristeći mini-serije podataka veličine 32. Tehnika proširivanja podataka, uključujući horizontalno okretanje i pomicanje po širini i visini, korištena je kako bi se povećala raznolikost podataka i smanjila mogućnost pretreniranosti. Validacija modela provedena je na testnim podacima nakon svake epohe.

```
pocetno_vrijeme = time.time()

history_densenet =
model_custom_densenet.fit(train_generator,
                           epochs=20,

steps_per_epoch=steps_per_epoch,

validation_data=(X_test_norm, y_test_onehot))

zavrsno_vrijeme = time.time()
ukupno_vrijeme = zavrsno_vrijeme - pocetno_vrijeme
print(f"Training Time: {ukupno_vrijeme:.2f} seconds")
```

5.2 ResNet50 model

ResNet50 je duboka konvolucijska neuronska mreža koja koristi "residual connections" kako bi olakšala treniranje dubokih mreža i poboljšala performanse. Residual connections omogućuju prijenos informacija iz ranijih slojeva direktno u kasnije slojeve, što pomaže u izbjegavanju problema s nestajanjem gradijenta. U nastavku je opisan postupak kreiranja, kompiliranja i treniranja ResNet50 modela za klasifikaciju slika iz CIFAR-10 skupa podataka.

5.2.1 Kreiranje modela

ResNet50 model je kreiran bez završnih slojeva, te je model učinjen "širim" povećanjem broja filtera i neurona, što je vidljivo u programskom kodu 5.5. Dodan je Global Average Pooling sloj te potpuno povezani sloj s 512 jedinica aktiviranih ReLU funkcijom. Na kraju je dodan izlazni sloj s 10 jedinica i aktivacijom softmax kako bi se modelom predviđele vjerojatnosti za svaku klasu.

```
resnet_base = ResNet50(weights='imagenet',
include_top=False, input_shape=(32, 32, 3))

model_wide_resnet = Sequential(name="WideResNet")

model_wide_resnet.add(resnet_base)

model_wide_resnet.add(GlobalAveragePooling2D())

model_wide_resnet.add(Dense(512, activation='relu'))
model_wide_resnet.add(Dropout(0.5))

model_wide_resnet.add(Dense(4, activation='softmax'))
```

ResNet50 je kompiliran uz korištenje Adam optimizatora i funkcije gubitka categorical crossentropy. Praćene metrike uključivale su točnost, preciznost i odaziv, kako je prikazano u programskom kodu 5.6.

Programski kod 5.6: Kompilacija ResNet50 modela

```
METRICS = [  
    'accuracy',  
    tf.keras.metrics.Precision(name='precision'),  
    tf.keras.metrics.Recall(name='recall')  
]  
  
model_wide_resnet.compile(optimizer=Adam(learning_rate=1e-  
4), loss='categorical_crossentropy', metrics=METRICS)
```

5.2.2 Treniranje modela

Model je treniran tijekom 20 epoha koristeći mini-serije podataka veličine 32, a tehnika augmentacije, kako je prikazano u programskom kodu 5.7, uključivala je horizontalno okretanje i pomicanje slika. Validacija modela provedena je na testnim podacima.

Programski kod 5.7: Treniranje ResNet50 modela i mjerenje vremena treniranja

```
pocetno_vrijeme = time.time()  
  
history_wide_resnet = model_wide_resnet.fit(train_generator,  
                                           epochs=20,  
  
steps_per_epoch=steps_per_epoch,  
  
validation_data=(X_test_norm, y_test_onehot))  
  
zavrsno_vrijeme = time.time()  
ukupno_vrijeme = zavrsno_vrijeme - pocetno_vrijeme  
  
print(f"Training Time for ResNet: {ukupno_vrijeme:.2f}  
seconds")
```

5.3 Custom CNN model

Custom CNN model dizajniran je za klasifikaciju slika iz CIFAR-10 skupa podataka. Model je izgrađen od konvolucijskih slojeva, slojeva za normalizaciju, slojeva za sažimanje i potpuno povezanih slojeva kako bi mogao izdvojiti ključne značajke iz slika.

5.3.1 Kreiranje modela

Više konvolucijskih slojeva s ReLU aktivacijom i Batch Normalization dodano je kako bi se treniranje učinilo stabilnijim. Kako je prikazano u programskom kodu 5.8, nakon svakog konvolucijskog bloka korišten je Max Pooling za smanjenje prostornih dimenzija

značajki, odnosno širine i visine slike. Ovaj proces omogućuje sažimanje informacija i smanjenje broja parametara, čime se poboljšava efikasnost modela i smanjuje rizik od pretreniranja. Na kraju, model je spljošten i povezan s potpuno povezanim slojem s 256 jedinica. Izlazni sloj koristi softmax za klasifikaciju slika u jednu od 4 klase.

Programski kod 5.8: Kreiranje custom CNN modela

```
CNN_custom = Sequential()

CNN_custom.add(Conv2D(filters=32, kernel_size=KERNEL_SIZE,
input_shape=INPUT_SHAPE, activation='relu', padding='same'))
CNN_custom.add(BatchNormalization())
CNN_custom.add(Conv2D(filters=32, kernel_size=KERNEL_SIZE,
input_shape=INPUT_SHAPE, activation='relu', padding='same'))
CNN_custom.add(BatchNormalization())
CNN_custom.add(MaxPool2D(pool_size=(2, 2)))
CNN_custom.add(Dropout(0.25))

CNN_custom.add(Conv2D(filters=64, kernel_size=KERNEL_SIZE,
input_shape=INPUT_SHAPE, activation='relu', padding='same'))
CNN_custom.add(BatchNormalization())
CNN_custom.add(Conv2D(filters=64, kernel_size=KERNEL_SIZE,
input_shape=INPUT_SHAPE, activation='relu', padding='same'))
CNN_custom.add(BatchNormalization())
CNN_custom.add(MaxPool2D(pool_size=(2, 2)))
CNN_custom.add(Dropout(0.25))

CNN_custom.add(Conv2D(filters=128, kernel_size=KERNEL_SIZE,
input_shape=INPUT_SHAPE, activation='relu', padding='same'))
CNN_custom.add(BatchNormalization())
CNN_custom.add(Conv2D(filters=128, kernel_size=KERNEL_SIZE,
input_shape=INPUT_SHAPE, activation='relu', padding='same'))
CNN_custom.add(BatchNormalization())
CNN_custom.add(MaxPool2D(pool_size=(2, 2)))
CNN_custom.add(Dropout(0.25))

CNN_custom.add(Flatten())
CNN_custom.add(Dense(256, activation='relu'))
CNN_custom.add(Dropout(0.25))
CNN_custom.add(Dense(4, activation='softmax'))
```

Iz programskog koda 5.9 vidljiv je model kompiliran pomoću Adam optimizatora i funkcije gubitka categorical crossentropy, dok su praćene metrike bile točnost, preciznost i odziv.

Programski kod 5.9: Kompilacija custom CNN modela

```
METRICS = [
    'accuracy',
    tf.keras.metrics.Precision(name='precision'),
    tf.keras.metrics.Recall(name='recall')
]
CNN_custom.compile(loss='categorical_crossentropy',
optimizer=Adam(learning_rate=1e-4), metrics=METRICS)
```

5.3.2 Treniranje modela

Treniranje je provedeno tijekom 20 epoha, koristeći tehniku augmentacije za povećanje raznolikosti podataka, kako je prikazano u programskom kodu 5.10. Trajanje treniranja i validacija modela na testnim podacima također su praćeni.

Programski kod 5.10: Treniranje custom CNN modela i mjerenje trajanja treniranja

```
pocetno_vrijeme = time.time()

r = CNN_custom.fit(train_generator,
                    epochs=20,
                    steps_per_epoch=steps_per_epoch,
                    validation_data=(X_test_norm, y_test_onehot),
                    )

zavrsno_vrijeme = time.time()
ukupno_vrijeme = zavrsno_vrijeme - pocetno_vrijeme

print(f"Training Time for Custom CNN: {ukupno_vrijeme:.2f}
seconds")
```

6. TESTIRANJE I REZULTATI

Ovo poglavlje sadrži usporedbu performansi triju modela Custom CNN, ResNet50 i DenseNet121 na skupu podataka CIFAR-10. Glavni fokus stavljen je na usporedbu vremena treniranja, funkcije gubitka, točnosti, te evaluacijskih metrika poput preciznosti, odziva, F1-scorea i matrice zabune za svaki model.

6.1 Trajanje treniranja

Trajanje treniranja može biti ključno za odabir modela, osobito kod složenih arhitektura poput ResNet50 i DenseNet121, koje obično zahtijevaju više vremena zbog većeg broja slojeva i parametara.

Tablica 6.1: Trajanje treniranja modela

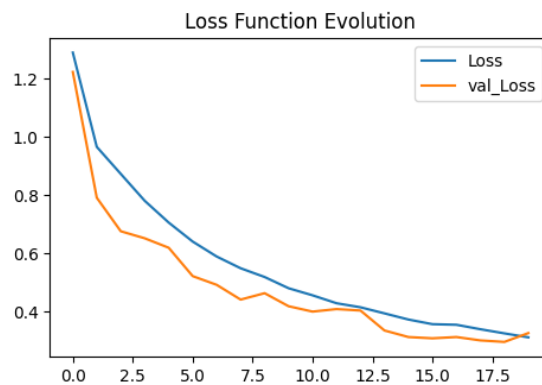
Model	Trajanje treniranja (s)
Custom CNN	509.75
ResNet50	2461.33
DenseNet121	2933.40

ResNet50 i DenseNet121 zahtijevaju značajno više vremena za treniranje u usporedbi s Custom CNN modelom, kako je vidljivo u tablici 6.1, što je očekivano zbog složenijih arhitektura.

6.2 Funkcija gubitka

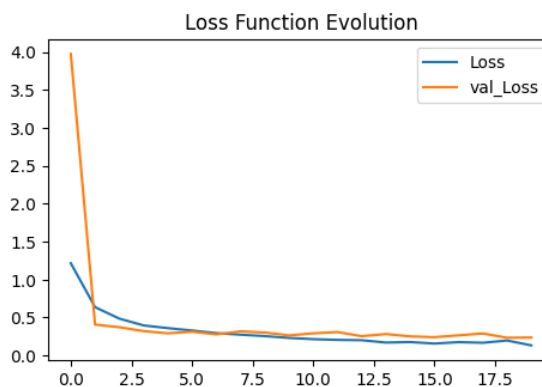
Funkcija gubitka mjeri koliko su predikcije modela udaljene od stvarnih vrijednosti tijekom treniranja. Na slikama 6.1, 6.2 i 6.3 prikazana je evolucija funkcije gubitka za sva tri modela – Custom CNN, ResNet50 i DenseNet121. Svaka slika prikazuje gubitak na treniranju prikazan plavom linijom i gubitak na validacijskom skupu prikazan narančastom linijom tijekom epoha. Custom CNN model pokazuje stabilan pad funkcije gubitka tijekom epoha, kako na treniranju, tako i na validacijskom skupu. Validacijski gubitak nije značajno

veći od trenirajućeg, što ukazuje na dobro generaliziranje modela bez znakova pretreniranosti, kako je prikazano na slici 6.1.



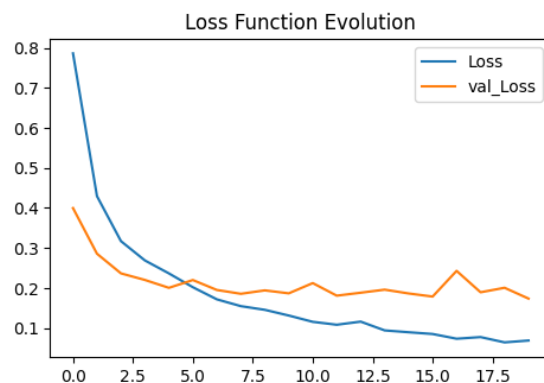
Slika 6.1: Funkcija gubitka za Custom CNN model

Za ResNet50 model primijećen je brzi pad gubitka u prvim epohama, s kasnijom stabilizacijom. Mala razlika između gubitka na trening skupu i gubitka na validacijskom skupu ukazuje na vrlo učinkovito učenje i dobru generalizaciju modela, kako je prikazano na slici 6.2.



Slika 6.2: Funkcija gubitka za ResNet50 model

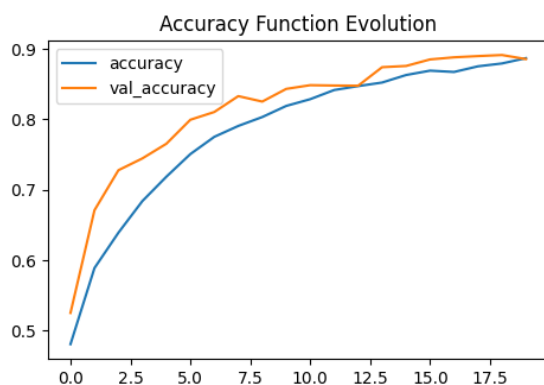
DenseNet121 model pokazuje sličan pad gubitka tijekom epoha, no s manjim oscilacijama u validacijskom gubitku. Ove varijacije su očekivane i ne ukazuju na pretreniranost. Model uspješno generalizira na validacijske podatke, iako bi smanjenje stope učenja moglo smanjiti varijabilnost, kako je prikazano na slici 6.3.



Slika 6.3: Funkcija gubitka za DenseNet model

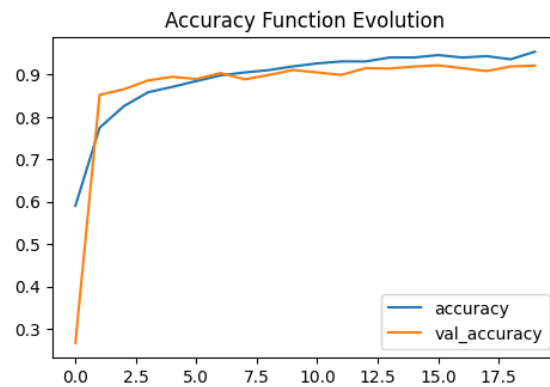
6.3 Funkcija točnosti

Funkcija točnosti mjeri postotak ispravno klasificiranih uzoraka tijekom treniranja modela. Slike 6.4, 6.5 i 6.6 prikazuju evoluciju točnosti tijekom epoha. Plava linija prikazuje točnost na trenirajućem skupu, dok narančasta linija prikazuje točnost na validacijskom skupu. Kod Custom CNN modela zabilježen je postepeni rast točnosti, s validacijskom točnošću koja doseže oko 90% prema kraju treniranja, kako je prikazano na slici 6.4. Ovaj rezultat ukazuje na dobru generalizaciju modela.



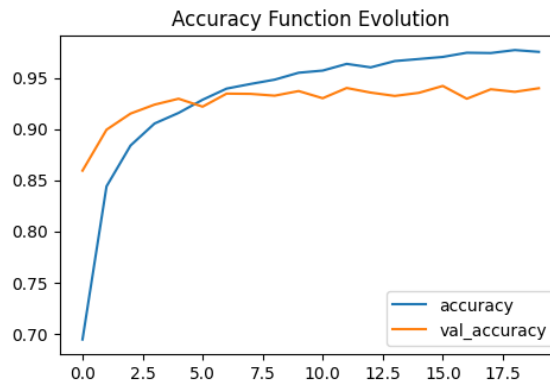
Slika 6.4: Funkcija točnosti za Custom CNN model

ResNet50 model pokazuje brz rast točnosti u ranim epohama, te stabilizaciju s točnošću iznad 90%, kako je prikazano na slici 6.5. Trenirajuća i validacijska točnost su gotovo identične, što potvrđuje dobru generalizaciju modela.



Slika 6.5: Funkcija točnosti za ResNet50 model

Za DenseNet121 model zabilježen je kontinuirani rast točnosti, pri čemu točnost treniranja dostiže 95%, kako je prikazano na slici 6.6, dok točnost validacije ostaje stabilna i visoka, oko 90-95%. Male oscilacije u validacijskoj točnosti su normalne i ne ukazuju na probleme s pretreniranjem.



Slika 6.6: Funkcija točnosti za DenseNet121 model

6.4 Matrica konfuzije

Matrica konfuzije je evaluacijska metrika koja pruža detaljan pregled performansi modela na razini svake klase. Prikazuje koliko je model točno klasificirao pozitivne i negativne primjere, ali i koliko je puta pogriješio klasificirajući uzorke u pogrešne klase. Svaka ćelija matrice prikazuje broj klasifikacija za kombinaciju predviđene i stvarne klase. Ova analiza ključna je za razumijevanje područja gdje model najviše griješi i gdje bi mogla biti potrebna poboljšanja. Custom CNN model pokazuje određene poteškoće u razlikovanju sličnih klasa. Najviše pogrešaka primijećeno je kod klasifikacije između automobila i kamiona, gdje su ti uzorci zamijenjeni u 149 slučajeva, kako je vidljivo iz tablice 6.2. Ovi rezultati sugeriraju da model ne prepoznaje dovoljno jasno značajke između tih dviju klasa, što bi se moglo dodatno poboljšati finim podešavanjem modela ili dodavanjem više podataka za treniranje.

Tablica 6.2: Matrica konfuzije za Custom CNN model

	Avion	Automobil	Brod	Kamion
Avion	774	16	107	103
Automobil	3	847	16	134
Brod	18	10	926	46
Kamion	7	15	13	965

ResNet50 pokazuje bolju sposobnost razlikovanja klasa, s minimalnim brojem pogrešaka između sličnih klasa. Primijećeno je samo 108 pogrešnih klasifikacija između automobila i kamiona, kako je prikazano u tablici 6.3, što sugerira da se neke značajke između tih klasa preklapaju. Iako su ove pogreške minimalne, dodatno smanjenje grešaka moguće je finim podešavanjem modela ili dodatnom obradom podataka. Općenito, ResNet50 pokazuje iznimno visoku točnost i stabilnost klasifikacije.

Tablica 6.3: Matrica konfuzije za ResNet50 model

	Avion	Automobil	Brod	Kamion
Avion	936	27	21	13
Automobil	7	961	7	25
Brod	44	31	909	16
Kamion	28	83	10	879

DenseNet121 postiže najbolje rezultate među sva tri modela, s najmanjim brojem pogrešaka u klasifikaciji. Zabilježene su minimalne pogreške između automobila i kamiona, pri čemu je došlo do 81 zamjene, dok model pokazuje gotovo savršenu točnost u prepoznavanju aviona, kako je prikazano u tablici 6.4. Rezultati ukazuju na to da DenseNet121 bolje prepoznaje značajke među klasama u usporedbi s drugim modelima, što ga čini najpreciznijim modelom u ovoj usporedbi.

Tablica 6.4: Matrica konfuzije za DenseNet121 model

	Avion	Automobil	Brod	Kamion
Avion	961	9	21	9
Automobil	12	942	7	39
Brod	47	13	928	12
Kamion	28	42	15	915

6.5 Evaluacijske metrike

Ovdje su detaljno objašnjene ključne evaluacijske metrike korištene za procjenu performansi modela, uključujući preciznost, odaziv i F1-ocjenu. Ove metrike omogućuju razumijevanje točnosti klasifikacije modela i njihove sposobnosti za rukovanje različitim

klasama unutar CIFAR-10 skupa podataka. Preciznost je omjer točno predviđenih pozitivnih primjera u odnosu na ukupni broj pozitivno klasificiranih primjera. Visoka preciznost znači da model rijetko klasificira negativne primjere kao pozitivne, što je važno kada želimo smanjiti broj lažno pozitivnih rezultata. Na primjer, u klasifikaciji automobila, visoka preciznost znači da model rijetko greškom označava brod ili avion kao automobil. Odaziv pokazuje koliko je točno prepoznatih pozitivnih primjera u odnosu na ukupan broj stvarnih pozitivnih slučajeva. Visoka vrijednost odaziva označava mogućnost prepoznavanja većinu stvarnih pozitivnih primjera. U kontekstu klasifikacije, visoki odaziv bi značio da model prepoznaje većinu automobila u skupu podataka, čak i ako ponekad greškom uključi nekoliko brodova. F1-ocjena je harmonijska sredina između preciznosti i odaziva, i koristi se kada je važno balansirati između njih. Ona daje bolji pregled općeg učinka modela, pogotovo kada su preciznost i odaziv u sukobu – ako je preciznost visoka, ali je odaziv nizak, ili obratno. F1-ocjena pokazuje ukupni balans između te dvije metrike.

Custom CNN model ostvario je sljedeće prosječne rezultate:

- Prosječna preciznost: 89,5%
- Prosječni odaziv: 88,5%
- Prosječna F1-ocjena: 88,5%

Ovi rezultati pokazuju da model postiže zadovoljavajuću preciznost i odaziv, iako postoji prostor za poboljšanje u prepoznavanju određenih klasa.

ResNet50 model je postigao sljedeće prosječne rezultate:

- Prosječna preciznost: 92%
- Prosječni odaziv: 92%
- Prosječna F1-ocjena: 92%

ResNet50 model pokazuje izvrsne performanse, s uravnoteženom preciznošću i odazivom. Ovo ukazuje na dobru sposobnost modela da točno klasificira slike unutar svih klasa.

DenseNet121 model ostvario je najbolje rezultate među modelima:

- Prosječna preciznost: 94%
- Prosječni odaziv: 94,24%
- Prosječna F1-ocjena: 94%

Ovi rezultati pokazuju da je DenseNet121 model najprecizniji i najpouzdaniji, s najboljom sposobnošću klasifikacije i prepoznavanja uzoraka iz CIFAR-10 skupa podataka.

7. ZAKLJUČAK

Svrha rada bila je napraviti usporedbu triju različitih modela dubokih neuronskih mreža: Custom CNN, ResNet50 i DenseNet121, u zadatku klasifikacije slika na CIFAR-10 skupu podataka. U radu je prikazan proces obrade podataka, treniranja modela i analize rezultata, uz fokus na ključne evaluacijske metrike poput preciznosti, odziva i F1-ocjene.

Rezultati su pokazali da se Custom CNN model najbrže trenirao, što ga čini pogodnim za zadatke u kojima je brzina treniranja ključna, čak i ako je dopuštena nešto niža točnost. Međutim, ovaj model je dao najlošije rezultate u usporedbi s druga dva modela, što se očituje kroz niže ocjene preciznosti i F1-ocjene. S druge strane, modeli ResNet50 i DenseNet121, unatoč dužem vremenu treniranja, postigli su bolje rezultate. ResNet50 se istaknuo kao izuzetno učinkovit model s visokim rezultatima već nakon nekoliko epoha, dok je DenseNet121 model pokazao najbolju ukupnu točnost i preciznost, iako je njegovo treniranje trajalo najduže.

Jedno od ključnih ograničenja ovog istraživanja jest činjenica da su korištene samo četiri klase iz CIFAR-10 skupa podataka. Primjena na složenije ili veće skupove podataka mogla bi otkriti dodatne izazove, osobito u pogledu vremena treniranja i skalabilnosti modela. Budući rad bi mogao uključivati eksperimentiranje s različitim tehnikama optimizacije i regularizacije, poput smanjenja stope učenja ili korištenja naprednijih metoda augmentacije podataka.

U zaključku, svaki model pokazuje određene prednosti i nedostatke, ovisno o specifičnom kontekstu primjene. Custom CNN omogućuje brzu obradu uz zadovoljavajuće rezultate, dok su ResNet50 i DenseNet121 modeli primjereniji za zadatke koji zahtijevaju visoku točnost i preciznost, uz veći vremenski trošak. Odabir modela ovisi o specifičnim zahtjevima primjene i dostupnim resursima.

8. LITERATURA

- [1] 3 Types of Machine Learning You Should Know, dostupno na: <https://www.coursera.org/articles/types-of-machine-learning>, [24.9.2024.]
- [2] Marko Josipović, Postupci strojnog učenja za popravljjanje točnosti klasifikacije manjinskih klasa kod nebalansiranih skupova podataka, 2019., [24.9.2024.]
- [3] Anushka Jain, Types of Machine Learning, dostupno na <https://www.geeksforgeeks.org/types-of-machine-learning/>, [24.9.2024.]
- [4] Kinza Yasar, What is a neural network, dostupno na <https://www.techtarget.com/searchenterpriseai/definition/neural-network>, [24.9.2024.]
- [5] Krešimir Kralj, Klasifikacija slika dubokim konvolucijskim modelima, dostupno na <https://www.zemris.fer.hr/~ssegvic/project/pubs/kralj17bs.pdf>, 2017., [24.9.2024.]
- [6] CNN [online], Convolutional Neural Network (CNN) in Machine Learning, [24.9.2024.]
- [7] Python [online] , wikipedia.org, dostupno na: [https://en.wikipedia.org/wiki/Python_\(programming_language\)](https://en.wikipedia.org/wiki/Python_(programming_language)), [24.9.2024.]
- [8] TensorFlow [online], wikipedia.org, dostupno na: <https://en.wikipedia.org/wiki/TensorFlow>, [24.9.2024.]
- [9] Keras Documentation [online], dostupno na: <https://keras.io/>, [24.9.2020.]
- [10] Google colab [online], dostupno na: <https://www.geeksforgeeks.org/getting-started-with-google-colab/>, [26.10.2024.]
- [11] Panadas [online], dostupno na: <https://pandas.pydata.org/>, [29.9.2024.]
- [12] NumPy [online], dostupno na <https://numpy.org/>, [29.9.2024.]
- [13] Matplotlib [online], dostupno na : <https://matplotlib.org/stable/project/history.html>, [29.9.2024.]
- [14] Scikit-learn [online] dostupno na: <https://scikit-learn.org/stable/>, [29.9.2024.]
- [15] Time [online], dostupno na: <https://realpython.com/python-time-module/>, [29.9.2024.]
- [16] Cifar-10 [online], dostupno na: <https://www.cs.toronto.edu/~kriz/cifar.html>, [29.9.2024.]

9. OZNAKE I KRATICE

API – Sučelje za programiranje aplikacija

Adam – Optimizator

CIFAR-10 – Skup podataka Kanadskog instituta za napredna istraživanja s 10 klasa slika

CNN – Konvolucijska neuronska mreža

CPU – Središnja procesorska jedinica

Deep Q-learning – Duboki algoritam za učenje kroz pojačanje

DenseNet121 – Duboka konvolucijska mreža s 121 slojem

F1-score – Harmonična sredina između preciznosti i odaziva

GPU – Grafička procesorska jedinica

Global Average Pooling – Tehnika smanjenja dimenzionalnosti u neuronskim mrežama

Q-learning – Algoritam za učenje kroz pojačanje

ReLU – Aktivacijska funkcija

ResNet50 – Duboka konvolucijska mreža s 50 slojeva i rezidualnim blokovima

SARSA – Algoritam za pojačano učenje

max pooling – Maksimalno sažimanje

10. SAŽETAK

Naslov: KONVOLUCIJSKE NEURONSKE MREŽE ZA PREPOZNAVANJE VOZILA: ANALIZA I USPOREDBA MODELA

Ovaj završni rad bavi se usporedbom triju različitih neuronskih mreža za klasifikaciju slika na skupu podataka CIFAR-10. Glavni cilj rada bio je analizirati performanse modela Custom CNN, ResNet50 i DenseNet121 u smislu vremena treniranja, točnosti, funkcije gubitka te evaluacijskih metrika poput preciznosti, odziva i F1-ocjene. Korišteni podaci unaprijed su obrađeni kroz normalizaciju i one-hot kodiranje, dok su modeli trenirani uz primjenu tehnike augmentacije podataka radi poboljšanja generalizacije. Rezultati su pokazali da je Custom CNN model najbrži u treniranju, ali s nešto slabijim performansama u usporedbi s ResNet50 i DenseNet121 modelima. DenseNet121 model, iako najzahtjevniji u smislu vremena treniranja, postigao je najbolje rezultate s najvišom točnošću i preciznošću. Ovi rezultati ukazuju na to da složeniji modeli pružaju bolje rezultate, ali uz veću vremensku cijenu treniranja.

Ključne riječi: neuronske mreže, CIFAR-10, klasifikacija slika, strojno učenje, Python.

11. ABSTRACT

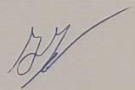
Title: CONVOLUTIONAL NEURAL NETWORKS FOR VEHICLE RECOGNITION: ANALYSIS AND MODEL COMPARISON

This thesis presents a comparison of three different neural networks for image classification using the CIFAR-10 dataset. The primary objective was to analyze the performance of the Custom CNN, ResNet50, and DenseNet121 models in terms of training time, accuracy, loss function, and evaluation metrics such as precision, recall, and F1-score. The data were preprocessed through normalization and one-hot encoding, and the models were trained using data augmentation techniques to enhance generalization. The results indicate that the Custom CNN model is the fastest to train but exhibits slightly lower performance compared to the ResNet50 and DenseNet121 models. The DenseNet121 model, while the most time-consuming in terms of training, achieved the best results with the highest accuracy and precision. These findings suggest that more complex models yield better results but require longer training times.

Keywords: neural networks, CIFAR-10, image classification, machine learning, Python.

IZJAVA O AUTORSTVU ZAVRŠNOG RADA

Pod punom odgovornošću izjavljujem da sam ovaj rad izradio/la samostalno, poštujući načela akademske čestitosti, pravila struke te pravila i norme standardnog hrvatskog jezika. Rad je moje autorsko djelo i svi su preuzeti citati i parafraze u njemu primjereno označeni.

Mjesto i datum	Ime i prezime studenta/ice	Potpis studenta/ice
U Bjelovaru, <u>9.10.2024.</u>	Antonio Jungić	

U skladu s čl. 58, st. 5 Zakona o visokom obrazovanju i znanstvenoj djelatnosti, Veleučilište u Bjelovaru dužno je u roku od 30 dana od dana obrane završnog rada objaviti elektroničke inačice završnih radova studenata Veleučilišta u Bjelovaru u nacionalnom repozitoriju.

Suglasnost za pravo pristupa elektroničkoj inačici završnog rada u nacionalnom repozitoriju

Antonio Jungić
ime i prezime studenta/ice

Dajem suglasnost da tekst mojeg završnog rada u repozitorij Nacionalne i sveučilišne knjižnice u Zagrebu bude pohranjen s pravom pristupa (zaokružiti jedno od ponuđenog):

- a) Rad javno dostupan
- b) Rad javno dostupan nakon _____ (upisati datum)
- c) Rad dostupan svim korisnicima iz sustava znanosti i visokog obrazovanja RH
- ☒ d) Rad dostupan samo korisnicima matične ustanove (Veleučilište u Bjelovaru)
- e) Rad nije dostupan

Svojim potpisom potvrđujem istovjetnost tiskane i elektroničke inačice završnog rada.

U Bjelovaru, 9.10.2024.

[potpis]
potpis studenta/ice