Procjena modela za otkrivanja objekata korištenjem matrice konfuzije te srednje prosječne preciznosti (mAP)

Njirić, Ana

Undergraduate thesis / Završni rad

2021

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Mechanical Engineering and Naval Architecture / Sveučilište u Zagrebu, Fakultet strojarstva i brodogradnje**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:235:907886

Rights / Prava: Attribution 4.0 International/Imenovanje 4.0 međunarodna

Download date / Datum preuzimanja: **2024-04-25**

Repository / Repozitorij:

Repository of Faculty of Mechanical Engineering and Naval Architecture University of Zagreb

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Ana Njirić

Zagreb, 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mentori:

doc. dr. sc. Tomislav Stipančić, dipl. ing.

Student:

Ana Njirić

Zagreb, 2021.

Izjavljujem da sam ovaj rad izradila samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Zahvaljujem se mentoru doc. dr. sc Tomislavu Stipančiću na pristupačnosti i pruženoj pomoći tijekom izrade rada.

Također zahvaljujem svojoj obitelji i prijateljima na razumijevanju i podršci tijekom preddiplomskog studija.

Ovaj rad posvećujem svojoj mami Leni koja me čuva i prati s neba. Znam da bi bila najponosnija.

Ana Njirić

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za završne ispite studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala i mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa: **602 - 04 / 21 - 6 / 1**	
Ur.broj: **15 - 1703 - 21 -**	

ZAVRŠNI ZADATAK

Student: **Ana Njirić** Mat. br.: 0035216876

Naslov rada na hrvatskom jeziku: **Procjena modela za otkrivanja objekata korištenjem matrice konfuzija te srednje prosječne preciznosti (mAP)**

Naslov rada na engleskom jeziku: **Evaluating object detection models using confusion matrix and mean average precision (mAP)**

Opis zadatka:

Alati za evaluaciju rada modela strojnog učenja pružaju uvid u skrivene zakonitosti među parametrima neuronske mreže te omogućuju fina podešavanja kako bi model ostvarivao bolje rezultate klasifikacije ili detekcije.

U radu je potrebno trenirati konvolucijsku neuronsku mrežu na CIFAR-100 bazi slika te evaluirati rad mreže. Posebno je potrebno proučiti i navesti neke od evaluacijskih tehnika analize modela strojnog učenja. Razvijeni model konvolucijske neuronske mreže ostvaren u Python programskom jeziku potrebno je evaluirati te dati analizu s obzirom na funkcije matrice konfuzija te srednje prosječne preciznosti (mAP).

U radu je potrebno dati osvrt na moguća poboljšanja rada mreže s obzirom na korištene evaluacijske tehnike, te navesti korištenu literaturu i eventualno dobivenu pomoć.

Zadatak zadan:
30. studenoga 2020.

Zadatak zadao:
Doc. dr. sc. Tomislav Stipančić

Datum predaje rada:
1. rok: 18 veljače 2021.
2. rok (izvanredni): 5. srpnja 2021.
3. rok: 23. rujna 2021.

Predviđeni datumi obrane:
1. rok: 22.2. – 26.2.2021.
2. rok (izvanredni): 9.7.2021.
3. rok: 27.9. – 1.10.2021.

Predsjednik Povjerenstva:
Prof. dr. sc. Branko Bauer

SADRŽAJ

SADRŽAJ I
POPIS SLIKA II
POPIS TABLICA III
POPIS OZNAKA IV
SAŽETAK V
SUMMARY VI
1. UVOD 1
2. TEORIJSKA OSNOVA RADA 2
2.1. Strojno učenje 2
2.2. Duboko učenje 3
2.3. Umjetne neuronske mreže 4
2.4. Konvolucijske neuronske mreže 5
2.5. Evaluacijske tehnike za analizu modela strojnog učenja 9
2.5.1. Matrica konfuzije za binarnu klasifikaciju 11
2.5.2. Matrica konfuzije za klasifikaciju više klasa 13
2.5.3. Točnost, preciznost i odziv 14
2.5.4. Krivulja preciznost- odziv 15
2.5.5. Prosječna preciznost 16
2.5.6. Križanje preko unije (IoU) 16
2.5.7. Srednja prosječna preciznost (mAP) 17
3. IZVEDBA ZADATKA 18
3.1. CIFAR-100 18
3.2. Programski jezik Python 19
3.2.1. TensorFlow 19
3.2.2. Keras 19
3.2.3. Matplotlib 20
3.2.4. NumPy 20
3.2.5. Sklearn ili Scikit-learn 20
3.2.6. Seaborn 20
3.2.7. Pandas 21
3.3. Učitavanje potrebnih biblioteka i modula 21
3.4. Učitavanje skupa podataka 22
3.5. Izrada konvolucijske neuronske mreže 24
4. ANALIZA REZULTATA 28
4.1. Usporedba rezultata s postojećim člancima 35
5. ZAKLJUČAK 36
LITERATURA 37
PRILOZI 39

POPIS SLIKA

Slika 2.1 Usporedba dubokog i strojnog učenja. [8] 3
Slika 2.2 Građa biološkog neurona. [10] 4
Slika 2.3 Model umjetnog neurona. [12] 5
Slika 2.4 Prikaz konvolucijske neuronske mreže. [14] 6
Slika 2.5 Primjer konvolucijskog sloja. [15] 6
Slika 2.6 Prikaz sloja sažimanja. [14] 7
Slika 2.7 ReLU aktivacijska funkcija. [16] 8
Slika 2.8 Prikaz potpuno povezanog sloja. [17] 8
Slika 2.9 Skup metode izdvajanja. [19] 10
Slika 2.10 Primjer k-struka unakrsne provjere valjanosti. [20] 10
Slika 2.11 Matrica konfuzije. [21] 12
Slika 2.12 Primjer krivulje preciznost-odziv. [22] 15
Slika 2.13 Jednadžba izračuna IoU iznosa. [23] 16
Slika 2.14 Primjer IoU. [24] 17
Slika 3.1. Učitavanje potrebnih biblioteka i modula. 21
Slika 3.2. Učitavanje CIFAR-100 skupa podataka 22
Slika 3.3. Kod za prikaz slika iz skupa podataka. 22
Slika 3.4. Prikaz 25 slika iz skupa podataka. 23
Slika 3.5. Priprema ulaznih podataka. 23
Slika 3.6. Kreiranje modela konvolucijske neuronske mreže. 24
Slika 3.7. Sažetak modela. 25
Slika 3.8. Prikaz treniranja modela. 25
Slika 3.9. Prikaz epoha kod treniranja mreže 1 od 3. 26
Slika 3.10. Prikaz epoha kod treniranja mreže 2 od 3. 26
Slika 3.11. Prikaz epoha kod treniranja mreže 3 od 3. 27
Slika 4.1. Grafovi točnosti i gubitka. 28
Slika 4.2. Grafovi preciznosti i odziva. 28
Slika 4.3. Kod za izračun matrice konfuzije. 29
Slika 4.4. Kod za prikaz matrice konfuzije. 29
Slika 4.5. Matrica konfuzije. 30
Slika 4.6. Izvještaj o klasifikaciji 1 od 3. 31
Slika 4.7. Izvještaj o klasifikaciji 2 od 3. 31
Slika 4.8. Izvještaj o klasifikaciji 3 od 3. 32
Slika 4.9. Preciznost-odziv krivulje klasa. 32
Slika 4.10. Zajednička preciznost-odziv krivulja klasa. 33
Slika 4.11. Srednja prosječna preciznost. 33
Slika 4.12. Kod za prikaz pogrešno predviđenih oznaka slika. 34
Slika 4.13. Slike sa točnim i predviđenim oznakama. 34

POPIS TABLICA

Tablica 2.1 Sličnosti između biološkog i umjetnog neurona. 5
Tablica 2.2 Usporedba istinitih i predviđenih oznaka. 11
Tablica 2.3 Primjer matrice konfuzije. 12
Tablica 2.4 Usporedba istinitih i predviđenih oznaka za klasifikaciju više klasa. 13
Tablica 2.5 Primjer matrice konfuzija za crvenu klasu. 13
Tablica 3.1 Superklase i klase CIFAR-100 skupa podataka. 18

POPIS OZNAKA

Oznaka	**Opis**
AP	prosječna preciznost (average precision)
CNN	konvolucijska neuronska mreža (convolutional neural network)
IoU	križanje preko unije (intersection over union)
mAP	srednja prosječna preciznost (mean average precision)
NumPy	Numerical Python
ReLU	ispravljena linearna jedinica (rectified linear unit)

SAŽETAK

Tema ovog rada je evaluacija modela konvolucijske neuronske mreže (CNN) na CIFAR-100 bazi slika. Model mreže je izrađen u *Python* programskom jeziku koristeći biblioteku *TensorFlow* uz razne dodatne biblioteke. Procjena modela je odrađena korištenjem matrice konfuzija te srednje prosječne preciznosti (mAP). U prvom dijelu teorijski će te biti upoznati s konvolucijski neuronskim mrežama i alatima za evaluaciju rada tih modela. Nakon toga slijedi dio u kojem će biti prikazano i objašnjeno kodiranje evaluacije u programu *Python*. Na kraju se prikazuju i analiziraju rezultati te navode moguća poboljšanja rada mreže.

Ključne riječi: konvolucijska neuronska mreža, CIFAR-100, matrica konfuzija, srednja prosječna preciznost (mAP), *Python*.

SUMMARY

The topic of this paper is the evaluation of the convolutional neural network model (CNN) on a CIFAR-100 image base. The network model is developed in *Python* programming language using *TensorFlow* library with various additional libraries. Model evaluation was done using confusion matrix and mean average precision (mAP). In the first part you will theoretically be informed about convolutional neural networks and tools for evaluating the operation of these models. This is followed by the part presenting and explaining the coding of the evaluation in *Python* program. Finally, the results are presented and analysed and possible improvements in network operation are indicated.

Key words: convolutional neural network, CIFAR-100, confusion matrix, mean average precision (mAP), *Python*.

1. UVOD

Umjetna inteligencija doprinosi razvoju tehnologija svakodnevno. Omogućuje računalima da obavljaju zadatke kao što to rade ljudi. Jedna od bitnih primjena umjetne inteligencije je računalni vid. On omogućuje računalima i sustavima da analiziraju i klasificiraju ulazne podatke kao što su slike, videozapisi i drugo. Važan dio računalnog vida za ovaj rad je prepoznavanje slika. Prvo je potrebno prikupiti skup podataka i sastaviti ga kao osnovu za treniranje. Zatim je potrebno podacima dodijeliti oznake, tj.klase. Nakon što je skup podataka spreman, moguće je prijeći na treniranje. Cilj treninga je da algoritam može napraviti predviđanja nakon analize slike i dodijeliti mu klasu. [1]

Na početku rada teorijski će te biti upoznati s pojmovima dubokog učenja, neuronskih mreža te evaluacijskim tehnikama analize modela strojnog učenja.

Nakon toga je na CIFAR-100 skupu podataka kreiran i treniran model konvolucijske neuronske mreže. Evaluacijskim tehnikama analize modela mreže prikazani su odnosi ulaznih podataka i podataka na kojima je model treniran.

Nakon analize rezultata tih tehnika vidimo koliko mreža može točno predvidjeti podatke, u ovom slučaju klase slika. Uspoređuju se rezultati s nekim već postojećim istraživanjima i navode moguća poboljšanja rada mreže.

2. TEORIJSKA OSNOVA RADA

2.1. Strojno učenje

Strojno učenje je grana umjetne inteligencije (AI) koja sustavima omogućava automatsko učenje i poboljšanje iz iskustva bez da su za to izričito programirani. Sustav učenja algoritma strojnog učenja može biti podijeljen u tri glavna koraka [2] :

- Postupak odlučivanja – Potrebno je odabrati skup ulaznih podataka koje će algoritam obrađivati kako bi izradio procjenu o uzorku podataka. Ulazni podaci mogu biti označeni (engl. labeled) ili neoznačeni (engl. unlabeled).
- Funkcija pogreške – Služi za procjenu predviđanja modela. Ako postoje poznati primjeri, funkcija pogreške može izvršiti usporedbu kako bi se procijenila točnost modela.
- Postupak optimizacije modela – Postupak učenja se ponavlja, varijabla prolazi kroz algoritam i uspoređuju s potrebnim rezultatima. Težinske vrijednosti algoritma se prilagođavaju prolaskom svake varijable kako bi se dobili točniji rezultati. Postupak se ponavlja sve dok rezultati ne budu zadovoljavajući.

Strojno učenje prema [3] možemo podijeliti u tri skupine: nadgledano, nenadgledano i pojačano učenje.

Nadgledano učenje (engl. *supervised learning*) koristi označene skupove podataka za obučavanje algoritama za točnu klasifikaciju podataka ili točno predviđanje ishoda. Skup podataka uključuje ulazne i izlazne podatke. Algoritam mjeri svoju točnost pomoću funkcije gubitka (engl. *loss function*), prilagođavajući je sve dok greška nije dovoljno smanjena. [3] Primjer nadgledanog učenja je klasifikacija neželjene pošte u zasebnu mapu od ulazne pošte.

Nenadgledano učenje (engl. *unsupervised learning*) koristi algoritme strojnog učenja za analizu, klasteriranje (grupiranje) i povezivanje neoznačenih skupova podataka te smanjenje dimenzionalnosti. Algoritmi otkrivaju sličnosti i razlike u informacijama te to čini nenadgledano učenje idealnim rješenjem za istraživačku analizu podataka i prepoznavanje slika. [4]

Pojačano učenje (engl. *reinforcement learning*) je model učenja kod kojeg je cilj sustava učenje iz vlastitih postupaka i iskustva . Razlikuje se od nadgledanog učenja po tome što u nadgledanom učenju izlazni podatak (rezultat) ovisi o ulaznom podatku zadanom na početku te

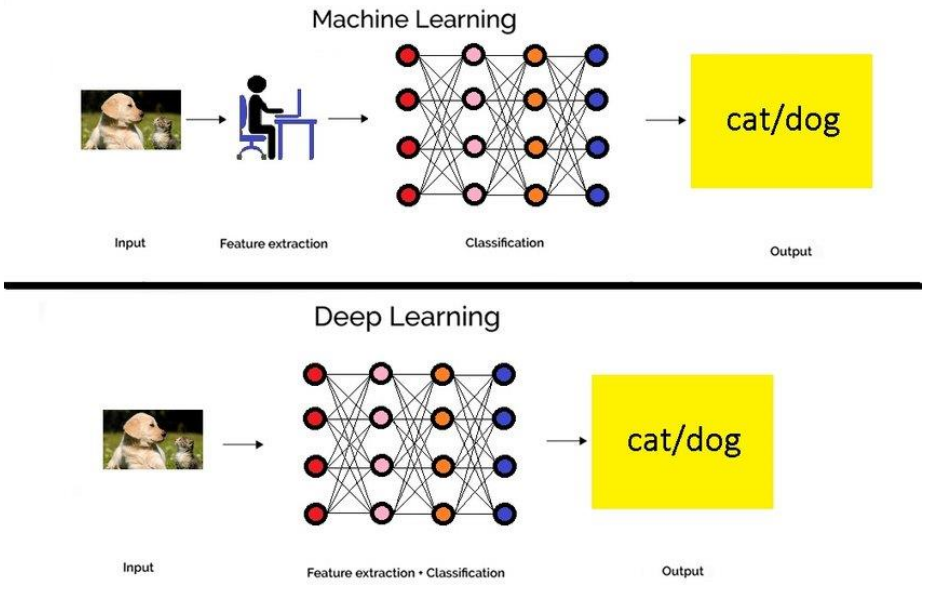
su odluke neovisne jedna o drugoj pa se oznake dodjeljuju svakoj odluci, a kod pojačanog učenja odluke se donose uzastopno, izlaz ovisi o stanju trenutnog ulaza i sljedeći ulaz ovisi o izlazu prethodnog ulaza te se oznake davaju nizovima ovisnih odluka. Primjer je igra šaha. [5]

2.2. Duboko učenje

Duboko učenje je podskup strojnog učenja, koje je u biti neuronska mreža s 3 ili više slojeva te je zbog toga drugi naziv duboka neuronska mreža. Neuronske mreže pokušavaju simulirati ponašanje ljudskog mozga dopuštajući modelu da samostalno uči i donosi odluke. Dodatni skriveni slojevi u neuronskoj mreži pomažu u optimizaciji i poboljšanju radi točnosti predviđanja. Tehnologija dubokog učenja se koristi u svakodnevnim proizvodima i uslugama, primjer su autonomni automobili (automobili koji sami upravljaju) te daljinski upravljači s mogućim glasovnim pretraživanjem. [6]

Osnovna razlika između dubokog i strojnog učenja je prikazana na slici [Slika 2.1]

Kod strojeva programiranih strojnim učenjem čovjek mora sudjelovati u ispravljanju grešaka koje je stroj napravio, dok kod dubokog učenja strojevi pomoću neuronske mreže sami mogu utvrditi da li je zadatak uspješno odrađen. Strojno učenje zahtijeva manje računalne snage, a duboko učenje zahtijeva manje ljudske intervencije. [7]

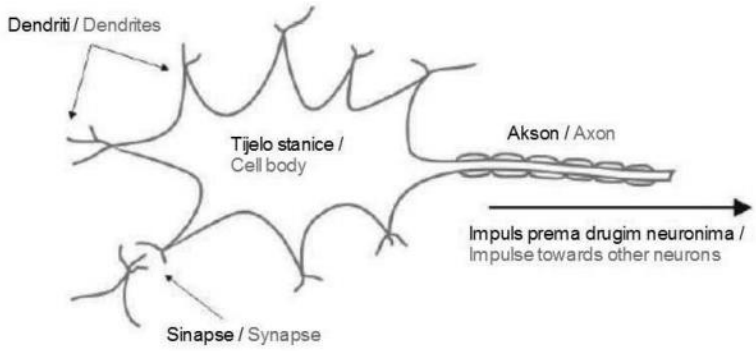


Slika 2.1 Usporedba dubokog i strojnog učenja. [8]

2.3. Umjetne neuronske mreže

Umjetna neuronska mreža (engl. *artificial neural network*) dizajnirana je da simulira način na koji ljudski mozak analizira i obrađuje podatke. Podskup je strojnog učenja i u središtu je algoritama dubokog učenja.

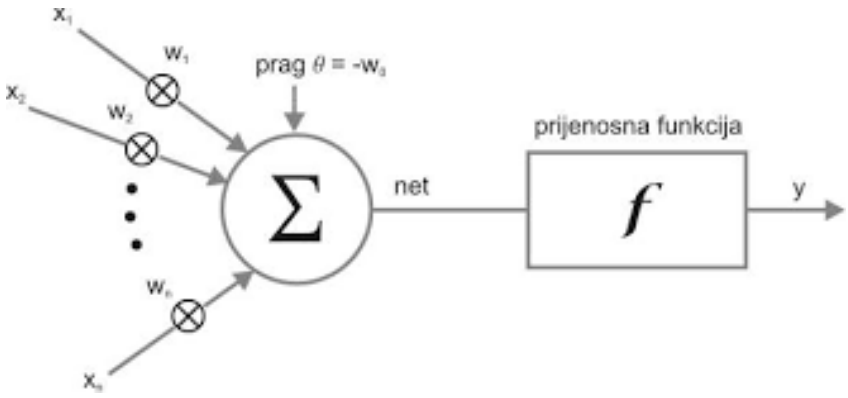
Umjetne neuronske mreže su izgrađene poput ljudskog mozga, sa neuronskim čvorovima isprepletenim poput mreže. Ljudski mozak ima stotine milijardi neurona. Svaki neuron sastoji se od staničnog tijela (soma), dendrita, aksona i sinapse. [9] Na slici [Slika 2.2] je prikazana građa biološkog neurona.



Slika 2.2 Građa biološkog neurona. [10]

Dendriti dovode signale izvana, soma je odgovorna za obradu ulaznih signala i odlučivanje treba li neuron aktivirati izlazne signale, akson je odgovoran za dobivanje obrađenih signala (impulsa) od neurona, a sinapsa je veza između aksona i drugih neuronskih dendrita i prenosi impulse s jednog neurona na drugi. [11]

Umjetna neuronska mreža ima stotine ili tisuće umjetnih neurona (perceptron). Umjetni neuron je matematička funkcija temeljena na modelu bioloških neurona. Na slici [Slika 2.3] je prikazan model umjetnog neurona. Signali su opisani numeričkim iznosom i na ulazu u neuron se množe težinskim faktorom, zatim se sumiraju te ako je iznos iznad definiranog praga, neuron daje izlazni signal. [12]



Slika 2.3 Model umjetnog neurona. [12]

U tablici [Tablica 2.1.] su navedene sličnosti između biološkog i umjetnog neurona.

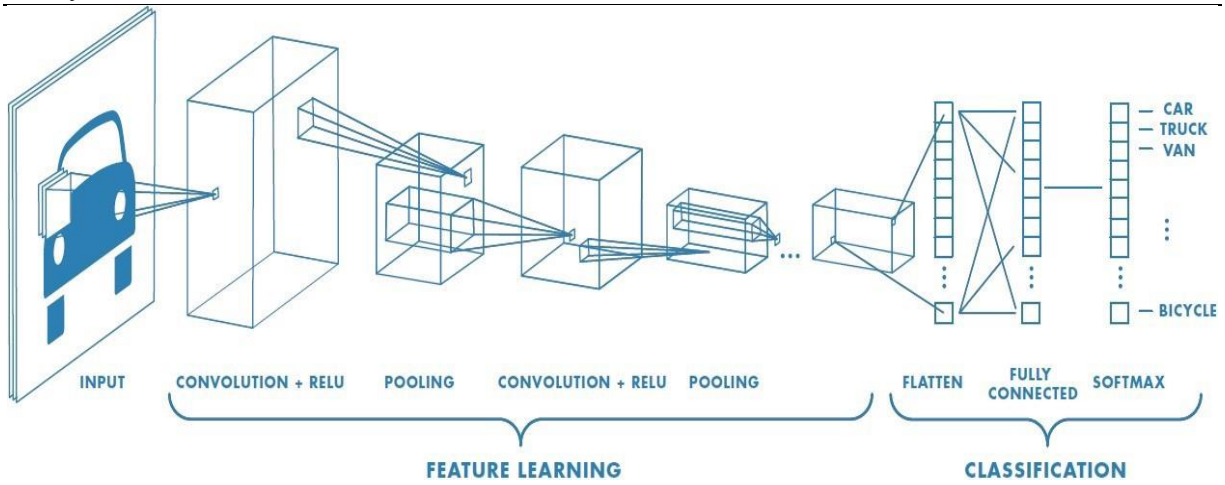
Tablica 2.1 Sličnosti između biološkog i umjetnog neurona.

Biološki neuron	**Umjetni neuron**
Ulazni signal prima putem dendrita	Ulazi su određeni težinskim faktorima
Signal se obrađuje u somi	Obrada ulaza; prag se dodaje sumi ulaza
Akson pretvara obrađeni ulaz u izlaz	Prijenosna funkcija pretvara ulaze u izlaz
Sinapsa prenosi informacije od neurona do ostalih neurona s kojima je povezan	Prenosi informacije do ostalih neurona

2.4. Konvolucijske neuronske mreže

Konvolucijske neuronske mreže (ConvNets ili CNN-s) su neuronske mreže koje se najčešće koriste za zadatke klasifikacije i računalnog vida. Pružaju skalabilniji pristup klasifikaciji slike i zadacima prepoznavanja objekata. Oslanjaju se na principe linearne algebre, posebno množenja matrica, za identifikaciju uzoraka slika. Razlikuju se od ostalih neuronskih mreža po boljoj obradi ulaznih slika, govora ili audio signala. Sastoje se od ulaznog sloja, skrivenih slojeva i izlaznog sloja. Tri glavne vrste skrivenih slojeva su konvolucijski sloj (engl. *convolutional layer*), sloj sažimanja (engl. *pooling layer*) i potpuno povezani sloj (engl. *fully-connected layer*). [13]

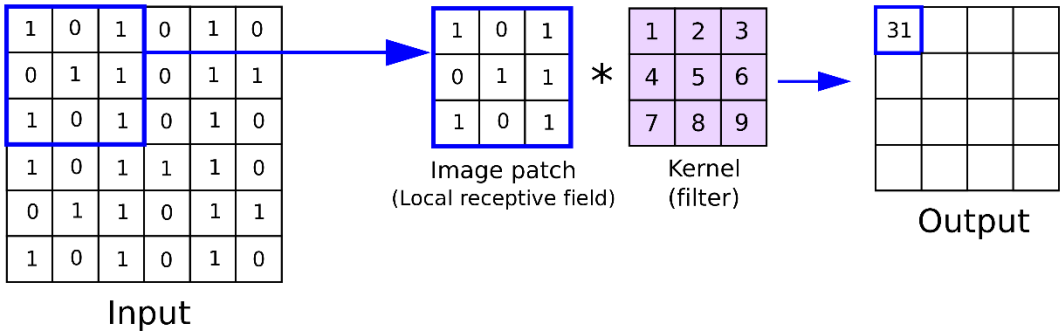
Na slici [Slika 2.4] je prikazana konvolucijska neuronska mreža.



Slika 2.4 Prikaz konvolucijske neuronske mreže. [14]

U mrežu ulazi ulazni volumen slike. Dimenzija dubine odgovara kanalima boje RGB (crvena, zelena, plava). Za nastavak pretpostavimo sliku u boji, volumena 32x32x3 (širina x visina x 3 kanala boja).

U konvolucijskom sloju filter (engl. *kernel*) se kreće po receptivnim poljima slike te je skenira. Veličina filtra je uglavnom matrica 3x3. Filtar se zatim primjenjuje na područje slike te se stvara proizvod s točkama proračunavanjem podataka ulaznih piksela slike i filtera. Taj proizvod s točkama se unosi u izlazni niz i od njega nastaje mapa značajki (engl. *feature map*). Na slici [Slika 2.5] je prikazan primjer konvolucijskog sloja s ulaznom slikom, receptivnim poljima (engl. *local receptive field*), filterom te izlaznim nizom. Konvolucijski sloj pretvara sliku u numeričke vrijednosti dopuštajući neuronskoj mreži da izdvaja i promatra bitne uzorke. [13]

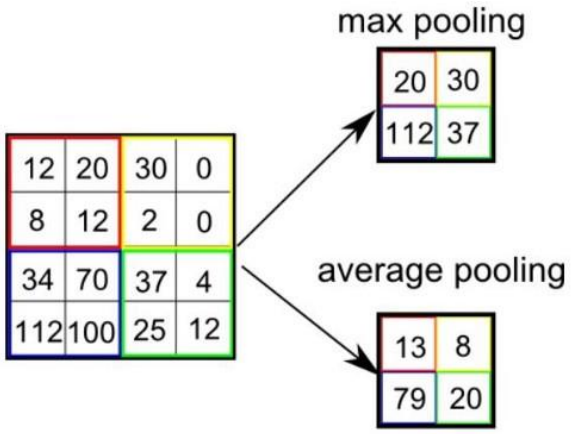


Slika 2.5 Primjer konvolucijskog sloja. [15]

Slojevi sažimanja još se nazivaju i smanjenja uzorkovanja (engl. *downsampling*) jer provode smanjenje dimenzionalnosti, smanjujući broj ulaznih parametara. Filter se kreće po cijeloj

ulaznoj slici kao i kod konvolucijskog sloja, ali ovaj filter koristi agregacijske funkcije (funkcije sakupljanja) na vrijednosti unutar receptivnog (prijemnog) polja popunjavajući izlazni niz. Postoje 2 vrste sažimanja : maksimalno sažimanje (engl. *max pooling*) i prosječno sažimanje (engl. *average pooling*). Kod maksimalnog sažimanja filter dok se pomiče po slici, odabire piksel s najvećom vrijednosti i šalje ga u izlazni niz. Kod prosječnog sažimanja filter izračunava prosječnu vrijednost unutar receptivnog polja te je šalje u izlazni niz. Ovaj sloj pomaže mreži u smanjenju složenosti i poboljšanju učinkovitosti. [13]

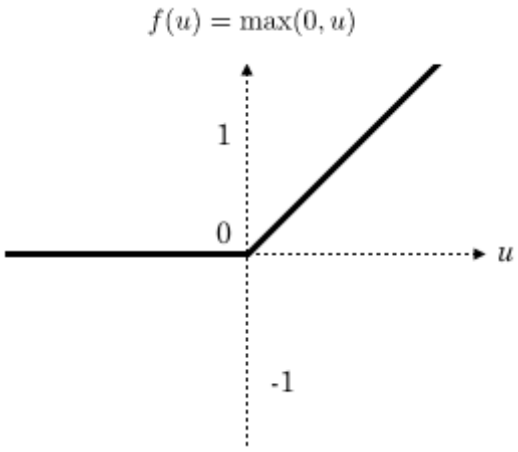
Na slici [Slika 2.6] je prikazan primjer sloja sažimanja.



Slika 2.6 Prikaz sloja sažimanja. [14]

Nakon slojeva sažimanja slijede ReLU slojevi. „ReLU“ označava „ispravljenu linearnu jedinicu“ (engl. *rectified linear unit*). Primjenjuje se na mapi značajki, unoseći nelinearnost u model. Nelinearnost unosi na način da zamijeni negativne vrijednosti piksela s nulama.

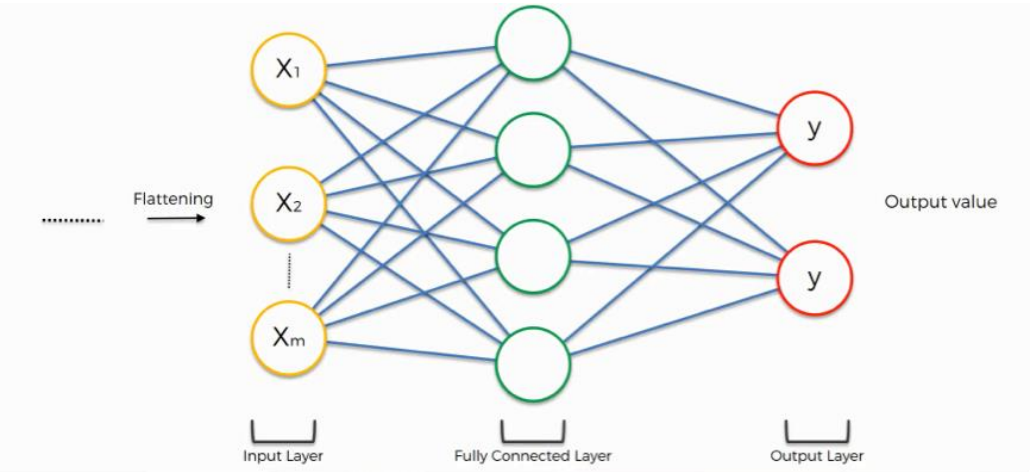
Na slici [Slika 2.7] je prikazana ReLU aktivacijska funkcija.



Slika 2.7 ReLU aktivacijska funkcija. [16]

U potpuno povezanim slojevima (engl. *fully-connected layers*) svaki čvor prima izlaz od svakog čvora prethodnog sloja. Obavlja klasifikaciju na temelju značajki izdvojenih iz prethodnih slojeva i njihovih filtera. Dok konvolucijski i slojevi sažimanja uglavnom koriste ReLU funkcije, potpuno povezani slojevi koriste aktivacijsku funkciju *Softmax* za klasifikaciju ulaza koja stvara vjerojatnost od 0 do 1. [13]

Na slici [Slika 2.8] je prikazan primjer potpuno povezanog sloja.



Slika 2.8 Prikaz potpuno povezanog sloja. [17]

2.5. Evaluacijske tehnike za analizu modela strojnog učenja

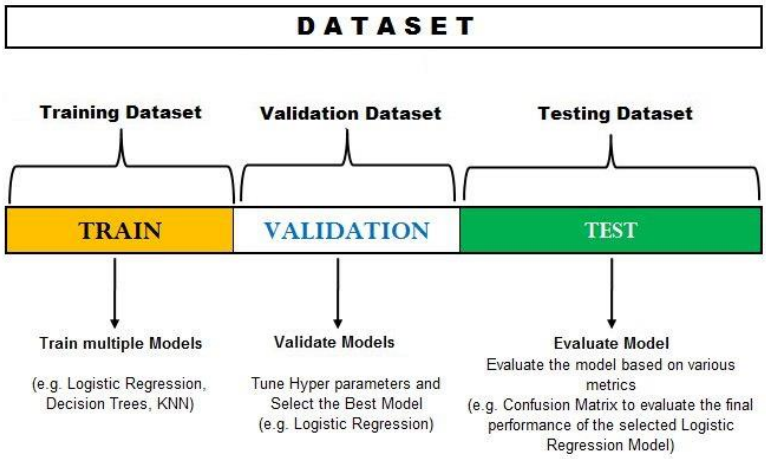
Modeli strojnog učenja bi trebali moći dati točna predviđanja kako bi stvorili stvarnu vrijednost za danu organizaciju. Ključni korak je treniranje modela ali jednako je važan i način na koji se model generalizira na neviđenim podacima. Cilj evaluacije modela je procijeniti tu generalizacijsku točnost modela na neviđenim podacima. Metode evaluacije izvedbe modela se mogu podijeliti u dvije kategorije: izdvajanje (engl. *holdout*) i unakrsna provjera valjanosti (engl. *cross-validation*). Metode koriste testni skup (engl. *test set*) za procjenu izvedbe modela. Testni skup čine podaci koje model ne vidi te nije preporučeno koristiti već poznate podatke za procjenu modela. [18]

Metoda izdvajanja želi testirati model na različitim podacima od onih na kojima se obučavao. Skup podataka je podijeljen u tri podskupa:

- Skup za treniranje (engl. *training set*) je podskup skupa podataka koji se koristi za izradu modela predviđanja.
- Skup provjere valjanosti (engl. *validation set*) je podskup skupa podataka koji se koristi za procjenu izvedbe modela nastalog u fazi treniranja.
- Testni skup (engl. *test set*) je podskup skupa podataka koji se koristi za procjenu budućih izvedbi modela. Ako se model bolje uklapa u skup za treniranje nego u testni skup, vjerojatno je uzrok prekomjerno uklapanje (engl. *overfitting*).

Pristup izdvajanja je koristan, fleksibilan i brz, ali često može doći do velike varijabilnosti (promjenljivosti) jer razlike u skupu podataka za obuku i testnom skupu mogu rezultirati velikim razlikama u procjeni točnosti. [18]

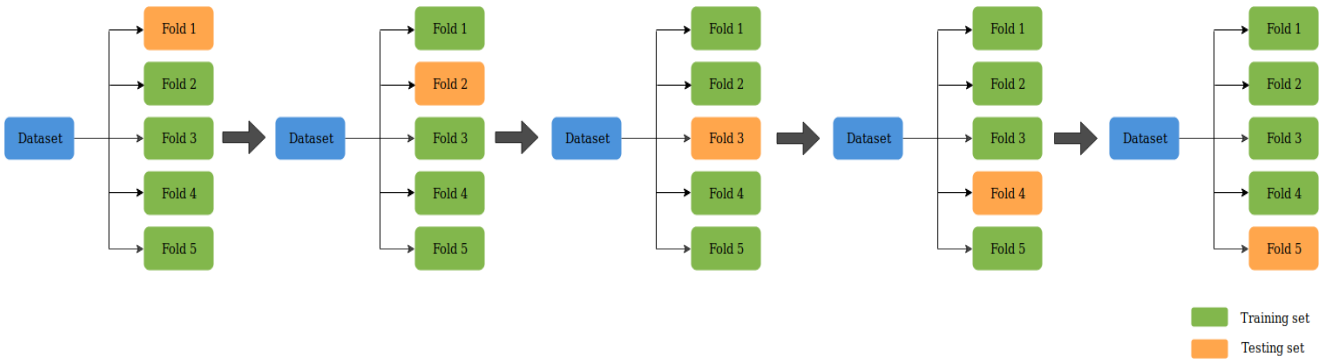
Na slici [Slika 2.9] je prikazan skup podataka metode izdvajanja.



Slika 2.9 Skup metode izdvajanja. [19]

Unakrsna provjera valjanosti dijeli izvorni skup podataka promatranja na skup za treniranje i neovisni skup za procjenu analize. Najčešća tehnika je k-struka unakrsna provjera valjanosti (engl. *k-fold cross validation*). Kod te tehnike izvorni skup podataka je podijeljen na *k* poduzorka jednake veličine, nazvanih preklopi (engl. *folds*). Ponavljamo postupak *k* puta s pomicanjem testnog skupa. Ako smo u prvoj iteraciji testirali model na prvom preklopu i na ostalim preklopima trenirali model, u sljedećoj ćemo iteraciji testirati model na drugom preklopu i trenirati na ostalima. Proces se ponavlja sve dok svaki preklop ne bude korišten kao testni skup. Ukupna učinkovitost modela se dobije prosječnom procjenom pogreške na svim *k* preklopima. [20]

Na slici [Slika 2.10] je prikazan primjer ove metode na skupu podataka s *k*=5 preklopa.



Slika 2.10 Primjer k-struka unakrsne provjere valjanosti. [20]

Za kvantificiranje izvedbi modela su potrebne metrike evaluacije modela. Izbor metrika (mjernih podataka) za procjenu modela ovisi o danom zadatku strojnog učenja, kao što su klasifikacija, regresija, grupiranje, rangiranje. Klasifikacija čini većinu aplikacija za strojno učenje. [18]

U nastavku će biti detaljnije objašnjene neke metrike koje se koriste za klasifikaciju problema.

2.5.1. Matrica konfuzije za binarnu klasifikaciju

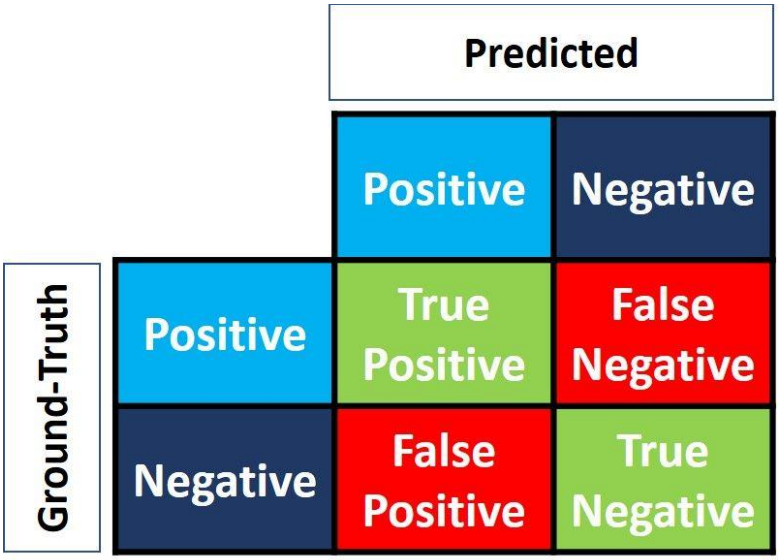
U binarnoj klasifikaciji svaki ulazni uzorak je dodijeljen jednoj od moguće dvije klase. Te dvije klase su označene oznakama poput 1 i 0, ili pozitivno i negativno. Na primjer, dvije oznake razreda mogu biti uspjeh ili neuspjeh studenta, zloćudni ili dobroćudni rak. [21]

Pretpostavit ćemo da postoji problem klasifikacije s klasama pozitivno i negativno. Pet uzoraka koji se koriste za obuku modela imaju oznake: pozitivno, negativno, negativno, pozitivno, pozitivno. Te oznake se nazivaju prizemno istinite oznake (engl. *ground-truth labels*). Za model je važan brojčani rezultat. Kada se uzorak dodava modelu, model ne vraća uvijek oznaku klase nego rezultat. Kada se ovih pet uzoraka unese u model dobiju se njihovi rezultati: 0.5, 0.8, 0.3, 0.25, 0.7 . Ove rezultate pretvorimo u oznake pomoću praga (engl. *threshold*). Prag može biti bilo koji broj koji odredi korisnik. Ovdje ćemo odabrati da je prag 0.5 i tada se svakom uzorku koji je iznad ili jednak broju 0.5 dodaje pozitivna oznaka, a manjima negativna. Za ovaj primjer su predviđene oznake: pozitivno (0.5), pozitivno (0.8), negativno (0.3), negativno (0.25), pozitivno (0.7). U tablici [Tablica 2.2] je prikazana usporedba prizemno istinitih oznaka i predviđenih oznaka. Možemo vidjeti da imam tri točna i dva pogrešna predviđanja.

Tablica 2.2 Usporedba istinitih i predviđenih oznaka.

Istinite oznake	pozitivno	negativno	negativno	pozitivno	pozitivno
Predviđene oznake	pozitivno	pozitivno	negativno	negativno	pozitivno

Dalje za dobivanje više informacija o izvedbi modela koristimo matricu konfuzije. Na slici [Slika 2.11] je prikazana matrica. Oblika je 2x2, oznake dva retka i dva stupca su pozitivno i negativno jer predstavljaju dvije oznake klasa. Oznake redaka predstavljaju istinite oznake, a oznake stupaca predviđene oznake. Četiri elementa matrice predstavljaju četiri metrike koje broje broj točnih i pogrešnih predviđanja modela. Svaki element ima oznaku od dvije riječi: istinito ili lažno, pozitivan ili negativan. Cilj je maksimirati mjerne podatke s riječi istinito (istinski pozitivno i istinski negativno), a minimizirati ostale (lažno pozitivno i lažno negativno).



Slika 2.11 Matrica konfuzije. [21]

Možemo izračunati ove četiri metrike na primjeru s pet uzoraka. Rezultati su prikazani u tablici [Tablica 2.3].

Tablica 2.3 Primjer matrice konfuzije.

		Predviđene oznake	
		Pozitivno	Negativno
Istinite oznake	Pozitivno	2	1
	Negativno	1	1

2.5.2. Matrica konfuzije za klasifikaciju više klasa

Slučaj računanja matrice konfuzije kada imamo problem s više klasa ćemo prikazati u primjeru. Pretpostavit ćemo da imamo deset uzoraka koji se mogu svrstati u tri klase: crvena, zelena, plava. Istinite oznake za ovih devet uzoraka su: crvena, plava, crvena, zelena, zelena, plava, crvena, crvena, plava, plava. Kada se unesu u model dobiju se predviđene oznake: plava, plava, crvena, crvena, zelena, zelena, plava, crvena, plava, zelena. U tablici [Tablica 2.4] je prikazana usporedba tih oznaka.

Tablica 2.4 Usporedba istinitih i predviđenih oznaka za klasifikaciju više klasa.

Istinite oznake	crvena	plava	crvena	zelena	zelena	plava	crvena	crvena	plava	plava
Predviđene oznake	plava	plava	crvena	crvena	zelena	zelena	plava	crvena	plava	zelena

Zatim moramo odrediti ciljnu klasu koja će biti označena kao pozitivna. Uzet ćemo da je to crvena. Ostale klase su negativne. Sada opet imamo dvije klase, pozitivno i negativno. U tablici [Tablica 2.5] je prikazana matrica konfuzije za crvenu klasu.

Tablica 2.5 Primjer matrice konfuzija za crvenu klasu.

		Predviđene oznake	
		Pozitivno	Negativno
Istinite oznake	Pozitivno	2	2
	Negativno	1	5

2.5.3. Točnost, preciznost i odziv

Točnost (engl. *accuracy*) je metrika koja opisuje izvedbu modela u svim klasama. Računa se kao omjer između broja točnih predviđanja i ukupnog broja predviđanja. Ta metrika nije baš dobar pokazatelj izvedbe modela. Uzmimo za primjer da model ima 530 točnih predviđanja za pozitivnu klasu od mogućih 550, a samo 5 točnih za negativnu klasu od mogućih 50. Tada je ukupna točnost $\frac{530+50}{600} = \mathbf{0,8917}$. To znači da je model 89,17% točan i trebao bi imati točna predviđanja u 89,17% slučajeva, a to nije istina.

Preciznost (engl. *precision*) je metrika koja se izračunava kao omjer između broja točno klasificiranih pozitivnih uzoraka i ukupnog broja uzoraka klasificiranih kao pozitivnih. Preciznost prikazuje točnost modela u klasifikaciji uzorka kao pozitivnog. Njen iznos je visok kada model napravi puno točnih klasifikacija uzoraka kao pozitivnih i kada napravi malo pogrešnih klasifikacija uzoraka kao pozitivnih.

Odziv (engl. *recall*) se izračunava kao omjer između broja točno klasificiranih uzoraka kao pozitivnih i ukupnog broja pozitivnih uzoraka. On mjeri sposobnost modela da uoči pozitivne uzorke. Neovisan je o tome kako su negativni uzorci klasificirani. Ako model klasificira sve pozitivne uzorke kao pozitivne, odziv će biti 100% čak i ako su svi negativni uzorci pogrešno klasificirani kao pozitivni. [21]

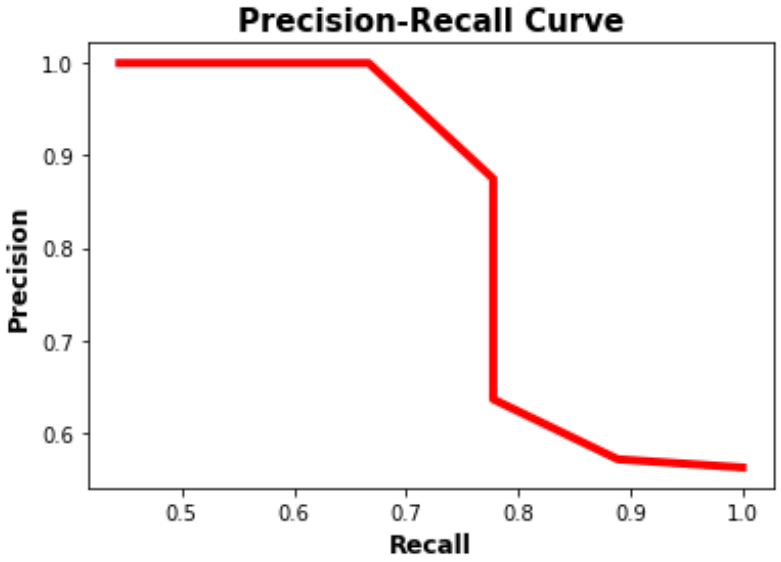
Odziv ovisi samo o pozitivnim uzorcima, dok preciznost ovisi i o negativnim i o pozitivnim uzorcima. Kada model ima visoku sposobnost odziva, ali nisku preciznost, tada on ispravno klasificira većinu pozitivnih uzoraka, ali ima puno lažno pozitivnih rezultata.

Ako želite pronaći sve pozitivne uzorke i nije bitno ako neki negativni uzorak bude pogrešno klasificiran kao pozitivan, onda je bolje koristiti odziv. Preciznost je bolje koristiti kada je problem osjetljiv na pogrešnu klasifikaciju negativnog uzorka kao pozitivnog.

2.5.4. Krivulja preciznost- odziv

Krivulja preciznost-odziv (engl. *precision-recall curve*) prikazuje kompromis između vrijednosti preciznosti i odziva za različite pragove. Ona pomaže u odabiru najboljeg praga za maksimiranje oba mjerna podatka. Za stvaranje krivulje potrebni su ulazi: istinite oznake, predviđeni rezultati uzoraka i pragovi za pretvaranje predviđenih rezultata u oznake klase.

Na slici [Slika 2.12] je prikazan primjer krivulje preciznog odziva. Povećanjem odziva preciznost se smanjuje. Kada se broj pozitivnih uzoraka povećava, točnost ispravne klasifikacije svakog uzorka se smanjuje. Na slici najbolja točka je (preciznost, odziv) = (0.778, 0.875) .



Slika 2.12 Primjer krivulje preciznost-odziv. [22]

Bolji način za odabir najboljih vrijednosti preciznosti i odziva je korištenje metrike koja se naziva $f1$ rezultat. Računa se pomoću jednadžbe $f1 = 2 * \frac{preciznost * odziv}{preciznost + odziv}$. Njen iznos predstavlja ravnotežu između preciznosti i odziva. Kada je iznos f 1 visok, znači da su i preciznost i odziv visoki. [22]

2.5.5. Prosječna preciznost

Prosječna preciznost (engl. *average precision*) je način za sažimanje krivulje preciznog odziva u jednu vrijednost koja predstavlja prosjek svih preciznosti. [22]

Računa se jednadžbom (1).

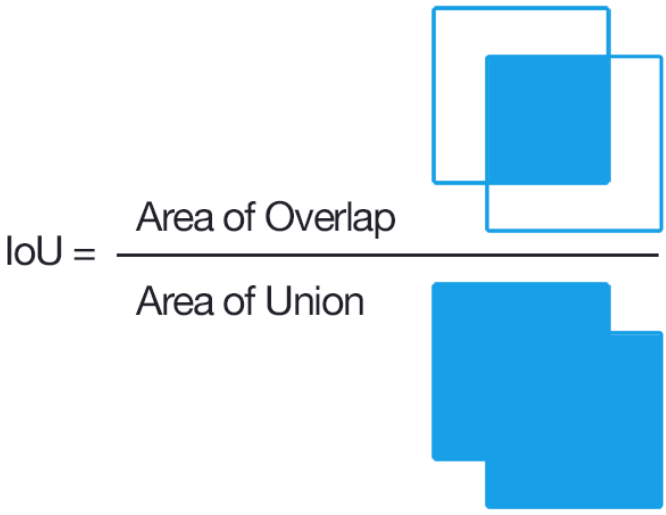
$$\boldsymbol{AP} = \sum_{\boldsymbol{k=0}}^{\boldsymbol{k=n-1}} [\boldsymbol{Odziv(k) - Odziv(k+1)}] * \boldsymbol{Preciznost(k)}$$

$$\boldsymbol{Odziv(n) = 0, Preciznost(n) = 1, n = broj\ pragova} \quad \textbf{(1)}$$

2.5.6. Križanje preko unije (IoU)

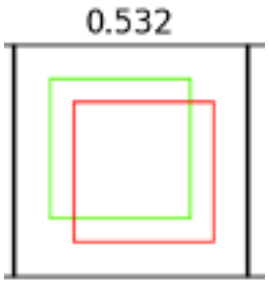
Križanje preko unije (engl. *intersection over union*, IoU) je način na koji otkrivanje objekta generira predviđene rezultate. Postoje 2 ulaza za uvježbavanje modela: slika i granični istiniti okviri za svaki objekt na slici. Model predviđa granične okvire otkrivenih objekata i očekivano je da se predviđeni okvir neće poklapati sa istinitim okvirom. [22]

Križanje preko unije je kvantitativna mjera za izračun usklađenosti istinitih i predviđenih okvira. Na slici [Slika 2.13] je prikazano kako se računa. Dijeli se površina područja križanja između 2 kutije sa površinom njihove unije (spoja). Predviđanje je bolje što je veći iznos.



Slika 2.13 Jednadžba izračuna IoU iznosa. [23]

Na slici [Slika 2.14] je prikazan primjer. Navedeni iznos IoU je 0.532 i to znači da 53,2% istinitog i predviđenog okvira se preklapa.



Slika 2.14 Primjer IoU. [24]

Za objektivnu procjenu je li model točno predvidio mjesto okvira koriste se pragovi. Ako model predviđa okvir s IoU rezultatom većim ili jednakim pragu, znači da postoji veliko preklapanje između predviđenog okvira i jednog od istinitih te je model uspješno otkrio objekt. Otkriveno područje se klasificira kao pozitivno (sadrži objekt). [22] To možemo prikazati jednadžbom (2).

$$klasa(IoU) = \begin{cases} Pozitivno \rightarrow IoU \geq prag \\ Negativno \rightarrow IoU < prag \end{cases}$$

(2)

2.5.7. Srednja prosječna preciznost (mAP)

Izračun srednje prosječne preciznosti (engl. *mean average precision*, mAP) započinje računanjem prosječne preciznosti za svaku klasu. Srednja vrijednost prosječne preciznosti svih klasa je mAP. Izračun se može prikazati jednadžbom (3) .

$$mAP = \frac{1}{n}\sum_{k=1}^{k=n} AP_k$$

$$AP_k = prosječna\ preciznost\ klasa\ k$$

$$n = broj\ klasa$$

(3)

3. IZVEDBA ZADATKA

3.1. CIFAR-100

U ovom radu će se koristiti CIFAR-100 skup podataka (engl. *dataset*). Skup se sastoji od 100 klasa od kojih svaka ima po 600 slika. U svakoj klasi ima 500 slika za treniranje i 100 slika za testiranje. Klase su grupirane u 20 superklasa. Svaka slika ima „finu“ oznaku (engl. *fine label*) koja označava klasu kojoj pripada i „grubu“ oznaku (engl. *coarse label*) koja označava superklasu kojoj pripada. U tablici [Tablica 3.1] su prikazane superklase i klase slika. [25]

Tablica 3.1 Superklase i klase CIFAR-100 skupa podataka.

Superklasa	**Klasa**
Vodeni sisavci	Dabar, dupin, vidra, tuljan, kit
Ribe	Akvarijske ribe, plosnate ribe, raže, morski pas, pastrva
Cvijeće	Orhideje, mak, ruže, suncokret, tulipani
Posude za hranu	Boce, zdjele, limenke, šalice, tanjuri
Voće i povrće	Jabuke, gljive, naranče, kruške, slatke paprike
Kućni električni uređaji	Sat, računalna tipkovnica, lampa, telefon, televizor
Kućni namještaj	Krevet, stolica, kauč, stol, ormar
Insekti	Pčela, buba, leptir, gusjenica, žohar
Veliki mesožderi	Medvjed, leopard, lav, tigar, vuk
Velike vanjske stvari koje je napravio čovjek	Most, dvorac, kuća, cesta, neboder
Velike prirodne scene na otvorenom	Oblak, šuma, planina, ravnica, more
Veliki svejedi i biljojedi	Deve, goveda, čimpanze, slon, klokan
Sisavci srednje veličine	Lisica, dikobraz, oposum, rakun, tvor
Beskičmenjaci koji nisu insekti	Rak, jastog, puž, pauk, crv
Ljudi	Beba, dječak, djevojčica, muškarac, žena
Gmazovi	Krokodil, dinosaur, gušter, zmija, kornjača
Mali sisavci	Hrčak, miš, zec, rovka, vjeverica
Drveća	Javor, hrast, palma, bor, vrba
Vozila 1	Bicikl, autobus, motocikl, kamion, vlak
Vozila 2	Kosilica, raketa, tramvaj, cisterna, traktor

3.2. Programski jezik Python

Python je objektno orijentirani, interpreterski, svestrani programski jezik. Jednostavan je za korištenje i može se koristiti za sve, od web razvoja do razvoja softvera i znanstvenih aplikacija.

Sadrži module, iznimke, dinamičke tipove podataka i visoke razine dinamičkih tipova odataka i klasa. Podržava i proceduralno i funkcionalno programiranje. Ima sučelja za mnoge sistemske pozive i biblioteke te je proširiv u C ili C++. Python radi na mnogim Unix verzijama kao što su Linux, macOS i Windows. [26]

Za ovaj rad je odabran Python izdanja 3.9 jer je besplatan i jednostavan s velikim mogućnostima. Sadrži ugrađene neke programske biblioteke i jednostavno se instaliraju ostale biblioteke.

Biblioteka sadrži prethodno napisane kodove koji se mogu koristiti i tako smanjuju vrijeme programiranja. Instaliraju se na način da se u terminal upiše 'pip install [paket]'. U nastavku su opisani paketi koji su korišteni u ovom radu.

3.2.1. TensorFlow

TensorFlow je biblioteka otvorenog koda za strojno učenje velikih razmjera i numeričko računanje. Razvio ju je Google Brain tim. Olakšava proces prikupljanja podataka, modela obuke, predviđanja i poboljšanja budućih rezultata. Spaja niz modela i algoritama strojnog učenja i dubokog učenja (neuronskih mreža). TensorFlow može trenirati i pokretati duboke neuronske mreže za klasifikaciju ručno napisanih znamenki, prepoznavanje slika, umetanje riječi i obradu prirodnog jezika. [27]

U ovom radu je korišten TensorFlow izdanja 2.6.

3.2.2. Keras

Keras je pristupačno, visoko produktivno programsko sučelje koje se nalazi unutar paketa TensorFlow. Smanjuje kognitivno opterećenje programera kako bi se fokusirao na probleme koji su zaista bitni. Fleksibilan je, jednostavni tijekovi rada trebali bi biti brzi i laki, a napredni tijekovi puta bi trebali biti mogući jasnim putem koji se nadovezuje na već naučeno.

Omogućuje potpunu iskoristivost mogućnosti platformi programa TensorFlow tako što se može pokrenuti na velikim grupama grafičkih procesora i Keras modeli mogu raditi u preglednicima i na mobilnim uređajima. [28]

3.2.3. Matplotlib

Matplotlib je opsežna biblioteka za stvaranje statičkih, animiranih i interaktivnih vizualizacija u Pythonu. Temeljena je na NumPy tipu podataka polja. Strukturirana je tako da je nekoliko redaka koda sve što je potrebno u većini slučajeva za generiranje grafičkog prikaza podataka. [29]

3.2.4. NumPy

NumPy (Numerical Python) je Python biblioteka koja se koristi za rad s poljima (engl. array). Biblioteka je otvorenog koda i može se besplatno koristiti, pristupačna je i produktivna za sve programere. Pruža sveobuhvatne matematičke funkcije, generatore slučajnih brojeva, Fourierove transformacije i još puno toga. NumPy koncepti vektorizacije, indeksiranja i emitiranja su brzi i svestrani. [30]

3.2.5. Sklearn ili Scikit-learn

Sklearn je Python biblioteka otvorenog koda koja podržava nadgledano i nenadgledano strojno učenje. Povezana je s Python bibliotekama NumPy, SciPy i matplotlib. Želi pružiti jednostavna i učinkovita rješenja problema učenja koji se mogu koristiti u različitim kontekstima. [31]

Modul sklearn.metrics uključuje funkcije bodovanja, izvedbu mjernih podataka i proračune udaljenosti. U radu će se koristiti za računanje mjernih podataka u matrici konfuzije. Također se koristi za računanje točnosti, preciznosti i odziva modela.

3.2.6. Seaborn

Seaborn je Python biblioteka za vizualizaciju podataka koja se temelji na matplotlibu. Pruža sučelje za crtanje informativnih statističkih grafika na visokoj razini. Blisko se integrira sa pandas strukturama podataka. [32]

3.2.7. Pandas

Pandas je Python biblioteka za rad sa skupovima podataka. Omogućuje nam analizu velikih podataka i donošenje zaključaka na temelju statističkih teorija. Može očistiti neuredne skupove podataka i učiniti ih čitljivima i relevantnima. [33]

3.3. Učitavanje potrebnih biblioteka i modula

Prvo u Python skriptu učitavamo potrebne biblioteke i module što je prikazano na slici [Slika 3.1].

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import sklearn.metrics
import seaborn as sns
import pandas as pd
```

```
from tensorflow import keras
from tensorflow.keras import datasets, layers, models
from tensorflow.keras.datasets import cifar100
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D
from tensorflow.keras.losses import categorical_crossentropy
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.metrics import Precision, Recall
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import  average_precision_score, precision_score, recall_score, precision_recall_curve
```

Slika 3.1. Učitavanje potrebnih biblioteka i modula.

Iz biblioteke *Tensorflow* uvozi se modul *keras*. Iz modula keras.datasets uvozi se baza podataka CIFAR-100, a iz modula keras.models uvozi se model *Sequential* (). Iz modula keras.layers uvoze se slojevi potrebni za izgradnju modela. Iz keras.losses uvozi se funkcija gubitka *categorical_crossentropy* za klasifikacijski model s više klasa. Iz keras.optimizers uvozi se optimizator *Adam* koji implementira Adam algoritam za tehniku optimizacije gradijentnog spuštanja. *ImageDataGenerator* iz modula keras.preprocessing.image umjetno povećava podatke za obuku (engl. *training dataset*).

Iz biblioteke *Sklearn* uvoze se metrike za procjenu izvedbe modela, a biblioteke *matplotlib.pyplot, seaborn* i *pandas* služe za vizualizaciju tih metrika.

3.4. Učitavanje skupa podataka

Učitavaju se podaci iz CIFAR-100 skupa podataka i odvajaju u dijelove za treniranje i testiranje što je i prikazano na slici [Slika 3.2].

```
#Učitavanje CIFAR-100 skupa podataka
(x_train, y_train), (x_test, y_test) = cifar100.load_data()

print(f"x_train shape: {x_train.shape}")
print(f"y_train shape: {y_train.shape}")
print(f"x_test shape: {x_test.shape}")
print(f"y_test shape: {y_test.shape}")
```

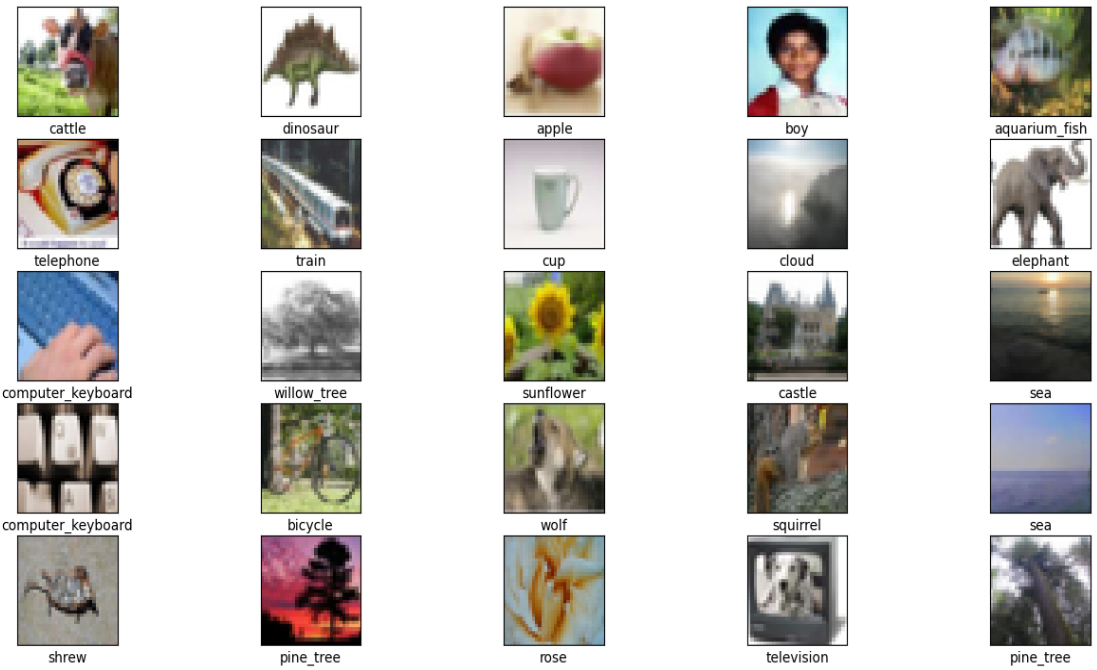
```
x_train shape: (50000, 32, 32, 3)
y_train shape: (50000, 1)
x_test shape: (10000, 32, 32, 3)
y_test shape: (10000, 1)
```

Slika 3.2. Učitavanje CIFAR-100 skupa podataka.

Za provjeru da je skup učitan na slici [Slika 3.3] je prikazan kod učitavanja slika, a na slici [Slika 3.4] je prikazano 25 slika iz skupa podataka.

```
class_names=['apple','aquarium_fish','baby','bear','beaver','bed','bee','beetle','bicycle','bottle','bowl',
             'boy','bridge','bus','butterfly','camel','can','castle','caterpillar','cattle','chair','chimpanzee',
             'clock','cloud','cockroach','couch','crab','crocodile','cup','dinosaur','dolphin','elephant',
             'flatfish','forest','fox','girl','hamster','house','kangaroo','computer_keyboard','lamp',
             'lawn_mower','leopard','lion','lizard','lobster','man','maple_tree','motorcycle','mountain',
             'mouse','mushroom','oak_tree','orange','orchid','otter','palm_tree','pear','pickup_truck',
             'pine_tree','plain','plate','poppy','porcupine','possum','rabbit','raccoon','ray','road',
             'rocket','rose','sea','seal','shark','shrew','skunk','skyscraper','snail','snake','spider',
             'squirrel','streetcar','sunflower','sweet_pepper','table','tank','telephone','television',
             'tiger','tractor','train','trout','tulip','turtle','wardrobe','whale','willow_tree','wolf',
             'woman','worm',]

superclass_names={'aquatic mammals': ['beaver', 'dolphin', 'otter', 'seal', 'whale'],
    'fish': ['aquarium_fish', 'flatfish', 'ray', 'shark', 'trout'],
    'flowers': ['orchid', 'poppy', 'rose', 'sunflower', 'tulip'],
    'food containers': ['bottle', 'bowl', 'can', 'cup', 'plate'],
    'fruit and vegetables': ['apple', 'mushroom', 'orange', 'pear',
                             'sweet_pepper'],
    'household electrical device': ['clock', 'computer_keyboard', 'lamp',
                                    'telephone', 'television'],
    'household furniture': ['bed', 'chair', 'couch', 'table', 'wardrobe'],
    'insects': ['bee', 'beetle', 'butterfly', 'caterpillar', 'cockroach'],
    'large carnivores': ['bear', 'leopard', 'lion', 'tiger', 'wolf'],
    'large man-made outdoor things': ['bridge', 'castle', 'house', 'road',
                                      'skyscraper'],
    'large natural outdoor scenes': ['cloud', 'forest', 'mountain', 'plain',
                                     'sea'],
    'large omnivores and herbivores': ['camel', 'cattle', 'chimpanzee',
                                       'elephant', 'kangaroo'],
    'medium-sized mammals': ['fox', 'porcupine', 'possum', 'raccoon', 'skunk'],
    'non-insect invertebrates': ['crab', 'lobster', 'snail', 'spider', 'worm'],
    'people': ['baby', 'boy', 'girl', 'man', 'woman'],
    'reptiles': ['crocodile', 'dinosaur', 'lizard', 'snake', 'turtle'],
    'small mammals': ['hamster', 'mouse', 'rabbit', 'shrew', 'squirrel'],
    'trees': ['maple_tree', 'oak_tree', 'palm_tree', 'pine_tree',
              'willow_tree'],
    'vehicles 1': ['bicycle', 'bus', 'motorcycle', 'pickup_truck', 'train'],
    'vehicles 2': ['lawn_mower', 'rocket', 'streetcar', 'tank', 'tractor']}

plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(x_train[i])
    plt.xlabel(class_names[y_train[i][0]])
plt.show()
```

Slika 3.3. Kod za prikaz slika iz skupa podataka.



Slika 3.4. Prikaz 25 slika iz skupa podataka.

Na slici [Slika 3.5] je prikazana priprema ulaznih podataka. Prvo pretvaramo podatke iz cijelih brojeva (integer) u decimalne brojeve (float) i normaliziramo vrijednosti piksela slika u raspon [0,1] tako što dijelimo trenutnu vrijednost piksela s najvećom vrijednošću. Zatim pretvaramo vektor klase u binarnu matricu klase, za upotrebu s funkcijom gubitka categorical_crossentropy. Određuje se ulazni oblik podataka: visina 32, širina 32 i tri kanala boja (crvena, zelena, plava).

```
#Normaliziranje podataka u raspon [-1,1]
x_train = x_train.astype('float32') /255
x_test = x_test.astype('float32') /255

#Pretvaranje vektora klase u matricu klase
y_train = to_categorical(y_train, 100)
y_test = to_categorical(y_test, 100)

#Određivanje ulaznog oblika podataka
img_width, img_height, img_no_channels = 32, 32, 3
input_shape = (img_width, img_height, img_no_channels)
```

Slika 3.5. Priprema ulaznih podataka.

3.5. Izrada konvolucijske neuronske mreže

Za izradu modela konvolucijske neuronske mreže (CNN) prvo moramo odrediti hiperparametre za treniranje modela koji su prikazani na slici [Slika 3.6]. Epoha uključuje jedan prolaz kroz skup podataka za obuku i sastoji se od 50 serija (engl. *batch*). U ovom slučaju imamo 50 epoha. Model računa predviđanja za svaki uzorak u seriji.

Definiramo prethodno istrenirani model Sequential i dodajemo dodatne slojeve. Dodajemo tri dvodimenzionalna konvolucijska sloja Conv2D i tri sloja sažimanja MaxPooling2D. Koristimo aktivacijsku funkciju ReLU. Zatim dodajemo sloj Flatten() koji podatke poravna u jednodimenzionalni sloj. Dense je pravilno povezani sloj koji opskrbljuje sve izlaze iz prethodnog sloja svojim neuronima, pri čemu svaki neuron daje jedan izlaz sljedećem sloju. Aktivacijska funkcija potpuno povezanog sloja je Softmax koja postavlja izlaze neuronske mreže na vrijednosti između 0 i 1 ,te zbroj svih izlaza je jednak 1.

```
#Konfiguracija modela
no_classes = 100  #broj klasa
no_epochs = 50  #broj epoha (ponavljanja) za trening
optimizer = keras.optimizers.Adam(learning_rate=0.001) #metoda kojom ažuriramo težine neuronske mreže
validation_split = 0.2 #20% podataka za trening se koristi u svrhu provjere valjanosti
verbosity = 1 #opširnost, 1=True, 0=False
batch_size = 50 #veličina serije = količina uzoraka koji odjednom idu u model i nakon njih se računa gubitak
loss_function = categorical_crossentropy #funkcija gubitka za usporedbu predviđanja s istinom
```

```
#Kreiranje modela
model = Sequential()
model.add(Conv2D(32, kernel_size=(3,3), activation='relu', input_shape=input_shape, padding="same"))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(64, kernel_size=(3,3), activation='relu', padding="same"))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(128, kernel_size=(3,3), activation='relu', padding="same"))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Flatten()) #pretvara višedimenzionalne podatke u 1D

model.add(Dense(256, activation='relu')) #omogućavaju klasifikaciju
model.add(Dense(128, activation='relu'))
model.add(Dense(no_classes, activation='softmax'))

model.summary()
```

Slika 3.6. Kreiranje modela konvolucijske neuronske mreže.

Na slici [Slika 3.7] je prikazan sažetak modela.

Nakon toga sastavljamo model i kreće obuka (treniranje modela) [Slika 3.8]. ImageDataGenerator povećava raznolikost podataka koji se koriste za treniranje modela.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 32, 32, 32)        896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 16, 16, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 8, 8, 64)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 8, 8, 128)         73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 4, 4, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 2048)              0
_________________________________________________________________
dense (Dense)                (None, 256)               524544
_________________________________________________________________
dense_1 (Dense)              (None, 128)               32896
_________________________________________________________________
dense_2 (Dense)              (None, 100)               12900
=================================================================
Total params: 663,588
Trainable params: 663,588
Non-trainable params: 0
_________________________________________________________________
```

Slika 3.7. Sažetak modela.

```
#Sastavljanje modela

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

model.compile(loss= loss_function,optimizer= optimizer, metrics=metrics)

data_generator = ImageDataGenerator(width_shift_range=0.1, height_shift_range=0.1,fill_mode='nearest',horizontal_flip=True,
                                    rotation_range=10)

train_generator = data_generator.flow(x_train, y_train)

steps_per_epoch = x_train.shape[0]// batch_size

history = model.fit(train_generator, steps_per_epoch = steps_per_epoch, validation_data=(x_test,y_test), verbose=1,
                    epochs=no_epochs, shuffle=True)
```

Slika 3.8. Prikaz treniranja modela.

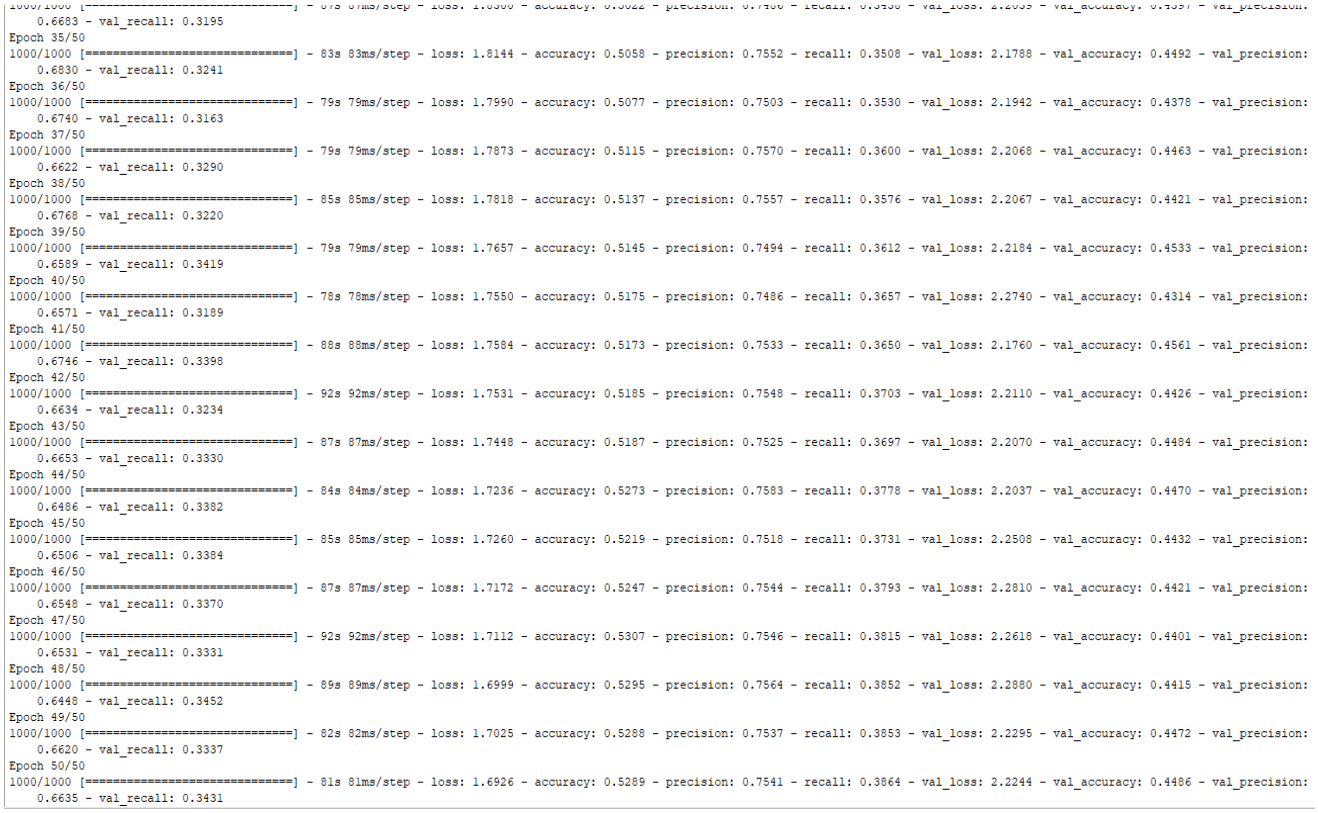
Na slikama [Slika 3.9], [Slika 3.10] i [Slika 3.11] su prikazane epohe kod treniranja neuronske mreže.

```
Epoch 1/50
1000/1000 [==============================] - 100s 100ms/step - loss: 4.0818 - accuracy: 0.0651 - precision: 0.4695 - recall: 0.0024 - val_loss: 3.6667 - val_accuracy: 0.1297 - val_precision:
    0.5818 - val_recall: 0.0096
Epoch 2/50
1000/1000 [==============================] - 99s 99ms/step - loss: 3.4751 - accuracy: 0.1599 - precision: 0.6002 - recall: 0.0230 - val_loss: 3.2291 - val_accuracy: 0.2111 - val_precision:
    0.6009 - val_recall: 0.0402
Epoch 3/50
1000/1000 [==============================] - 93s 93ms/step - loss: 3.1570 - accuracy: 0.2195 - precision: 0.6344 - recall: 0.0513 - val_loss: 2.9724 - val_accuracy: 0.2672 - val_precision:
    0.6682 - val_recall: 0.0850
Epoch 4/50
1000/1000 [==============================] - 85s 85ms/step - loss: 2.9530 - accuracy: 0.2598 - precision: 0.6667 - recall: 0.0782 - val_loss: 2.8368 - val_accuracy: 0.2892 - val_precision:
    0.6785 - val_recall: 0.1112
Epoch 5/50
1000/1000 [==============================] - 85s 85ms/step - loss: 2.7977 - accuracy: 0.2918 - precision: 0.6887 - recall: 0.1049 - val_loss: 2.6787 - val_accuracy: 0.3180 - val_precision:
    0.7117 - val_recall: 0.1237
Epoch 6/50
1000/1000 [==============================] - 86s 86ms/step - loss: 2.6747 - accuracy: 0.3174 - precision: 0.6903 - recall: 0.1252 - val_loss: 2.6205 - val_accuracy: 0.3376 - val_precision:
    0.7033 - val_recall: 0.1496
Epoch 7/50
1000/1000 [==============================] - 87s 87ms/step - loss: 2.5679 - accuracy: 0.3395 - precision: 0.7005 - recall: 0.1476 - val_loss: 2.6525 - val_accuracy: 0.3272 - val_precision:
    0.6885 - val_recall: 0.1545
Epoch 8/50
1000/1000 [==============================] - 80s 80ms/step - loss: 2.4906 - accuracy: 0.3534 - precision: 0.7125 - recall: 0.1637 - val_loss: 2.4969 - val_accuracy: 0.3651 - val_precision:
    0.7032 - val_recall: 0.1891
Epoch 9/50
1000/1000 [==============================] - 81s 81ms/step - loss: 2.4317 - accuracy: 0.3701 - precision: 0.7204 - recall: 0.1799 - val_loss: 2.4581 - val_accuracy: 0.3782 - val_precision:
    0.6785 - val_recall: 0.2146
Epoch 10/50
1000/1000 [==============================] - 86s 86ms/step - loss: 2.3702 - accuracy: 0.3828 - precision: 0.7192 - recall: 0.1942 - val_loss: 2.4149 - val_accuracy: 0.3765 - val_precision:
    0.6859 - val_recall: 0.2107
Epoch 11/50
1000/1000 [==============================] - 84s 84ms/step - loss: 2.3188 - accuracy: 0.3908 - precision: 0.7182 - recall: 0.2035 - val_loss: 2.4103 - val_accuracy: 0.3848 - val_precision:
    0.6763 - val_recall: 0.2294
Epoch 12/50
1000/1000 [==============================] - 82s 82ms/step - loss: 2.2691 - accuracy: 0.4034 - precision: 0.7230 - recall: 0.2164 - val_loss: 2.3824 - val_accuracy: 0.3891 - val_precision:
    0.7010 - val_recall: 0.2305
Epoch 13/50
1000/1000 [==============================] - 83s 83ms/step - loss: 2.2335 - accuracy: 0.4115 - precision: 0.7338 - recall: 0.2295 - val_loss: 2.3371 - val_accuracy: 0.4032 - val_precision:
    0.6843 - val_recall: 0.2503
Epoch 14/50
1000/1000 [==============================] - 82s 82ms/step - loss: 2.2022 - accuracy: 0.4189 - precision: 0.7368 - recall: 0.2380 - val_loss: 2.3459 - val_accuracy: 0.3951 - val_precision:
    0.6977 - val_recall: 0.2433
Epoch 15/50
1000/1000 [==============================] - 83s 83ms/step - loss: 2.1654 - accuracy: 0.4258 - precision: 0.7347 - recall: 0.2482 - val_loss: 2.4019 - val_accuracy: 0.3877 - val_precision:
    0.6827 - val_recall: 0.2520
Epoch 16/50
1000/1000 [==============================] - 80s 80ms/step - loss: 2.1363 - accuracy: 0.4305 - precision: 0.7328 - recall: 0.2523 - val_loss: 2.2942 - val_accuracy: 0.4073 - val_precision:
    0.6873 - val_recall: 0.2629
```

Slika 3.9. Prikaz epoha kod treniranja mreže 1 od 3.

```
Epoch 17/50
1000/1000 [==============================] - 82s 82ms/step - loss: 2.1061 - accuracy: 0.4409 - precision: 0.7370 - recall: 0.2647 - val_loss: 2.3800 - val_accuracy: 0.3862 - val_precision:
    0.6559 - val_recall: 0.2386
Epoch 18/50
1000/1000 [==============================] - 87s 87ms/step - loss: 2.0657 - accuracy: 0.4491 - precision: 0.7374 - recall: 0.2715 - val_loss: 2.3235 - val_accuracy: 0.4040 - val_precision:
    0.6755 - val_recall: 0.2667
Epoch 19/50
1000/1000 [==============================] - 83s 83ms/step - loss: 2.0518 - accuracy: 0.4493 - precision: 0.7329 - recall: 0.2769 - val_loss: 2.3062 - val_accuracy: 0.4071 - val_precision:
    0.6740 - val_recall: 0.2619
Epoch 20/50
1000/1000 [==============================] - 81s 81ms/step - loss: 2.0316 - accuracy: 0.4553 - precision: 0.7383 - recall: 0.2818 - val_loss: 2.1989 - val_accuracy: 0.4247 - val_precision:
    0.7023 - val_recall: 0.2824
Epoch 21/50
1000/1000 [==============================] - 89s 89ms/step - loss: 2.0082 - accuracy: 0.4603 - precision: 0.7412 - recall: 0.2897 - val_loss: 2.2035 - val_accuracy: 0.4325 - val_precision:
    0.7001 - val_recall: 0.2754
Epoch 22/50
1000/1000 [==============================] - 88s 88ms/step - loss: 1.9805 - accuracy: 0.4674 - precision: 0.7458 - recall: 0.2955 - val_loss: 2.1816 - val_accuracy: 0.4367 - val_precision:
    0.6879 - val_recall: 0.3106
Epoch 23/50
1000/1000 [==============================] - 85s 85ms/step - loss: 1.9894 - accuracy: 0.4647 - precision: 0.7347 - recall: 0.2936 - val_loss: 2.3152 - val_accuracy: 0.4086 - val_precision:
    0.6694 - val_recall: 0.2661
Epoch 24/50
1000/1000 [==============================] - 86s 86ms/step - loss: 1.9435 - accuracy: 0.4785 - precision: 0.7489 - recall: 0.3104 - val_loss: 2.1935 - val_accuracy: 0.4436 - val_precision:
    0.6721 - val_recall: 0.3243
Epoch 25/50
1000/1000 [==============================] - 86s 86ms/step - loss: 1.9433 - accuracy: 0.4737 - precision: 0.7428 - recall: 0.3102 - val_loss: 2.2435 - val_accuracy: 0.4265 - val_precision:
    0.6728 - val_recall: 0.3011
Epoch 26/50
1000/1000 [==============================] - 87s 87ms/step - loss: 1.9170 - accuracy: 0.4812 - precision: 0.7439 - recall: 0.3176 - val_loss: 2.1574 - val_accuracy: 0.4452 - val_precision:
    0.7055 - val_recall: 0.3064
Epoch 27/50
1000/1000 [==============================] - 89s 89ms/step - loss: 1.9076 - accuracy: 0.4850 - precision: 0.7478 - recall: 0.3215 - val_loss: 2.2226 - val_accuracy: 0.4356 - val_precision:
    0.6814 - val_recall: 0.3077
Epoch 28/50
1000/1000 [==============================] - 91s 91ms/step - loss: 1.8930 - accuracy: 0.4871 - precision: 0.7386 - recall: 0.3198 - val_loss: 2.2129 - val_accuracy: 0.4375 - val_precision:
    0.6736 - val_recall: 0.3157
Epoch 29/50
1000/1000 [==============================] - 128s 128ms/step - loss: 1.8708 - accuracy: 0.4913 - precision: 0.7485 - recall: 0.3311 - val_loss: 2.1606 - val_accuracy: 0.4482 - val_precision
    0.6909 - val_recall: 0.3114
Epoch 30/50
1000/1000 [==============================] - 109s 109ms/step - loss: 1.8650 - accuracy: 0.4931 - precision: 0.7501 - recall: 0.3332 - val_loss: 2.2587 - val_accuracy: 0.4357 - val_precision
    0.6586 - val_recall: 0.3195
Epoch 31/50
1000/1000 [==============================] - 85s 85ms/step - loss: 1.8439 - accuracy: 0.4972 - precision: 0.7508 - recall: 0.3404 - val_loss: 2.2295 - val_accuracy: 0.4325 - val_precision:
    0.6723 - val_recall: 0.3098
Epoch 32/50
1000/1000 [==============================] - 86s 86ms/step - loss: 1.8429 - accuracy: 0.4978 - precision: 0.7472 - recall: 0.3354 - val_loss: 2.3277 - val_accuracy: 0.4226 - val_precision:
    0.6434 - val_recall: 0.3114
Epoch 33/50
```

Slika 3.10. Prikaz epoha kod treniranja mreže 2 od 3.

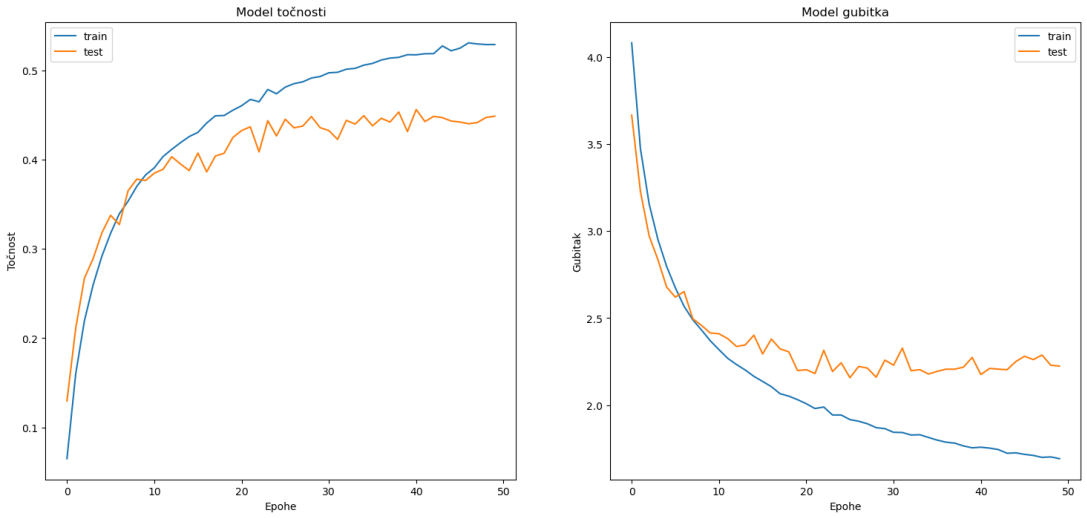


Slika 3.11. Prikaz epoha kod treniranja mreže 3 od 3.

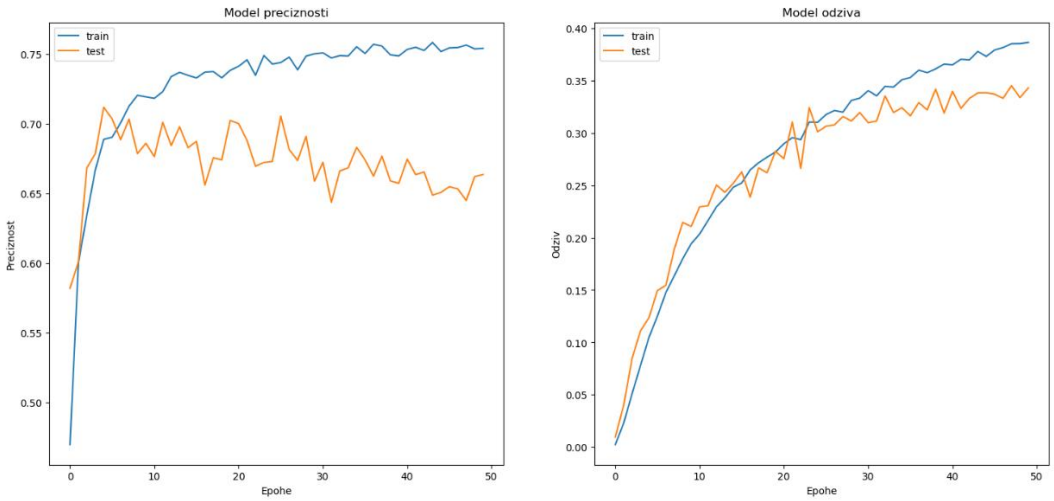
4. ANALIZA REZULTATA

Na slici [Slika 4.1] su prikazani grafovi točnosti i gubitka CNN modela. Na grafu točnosti je prikazana točnost prepoznavanja klasa kroz epohe učenja. Postignuta točnost testiranja mreže je 44.86%. Graf gubitka se dobiva iz funkcije gubitka (engl. *loss function*) koja se izračunava za svaku stavku podataka i želi minimizirati dobivene vrijednosti svakom epohom. Na oba grafa krivulje su zadovoljavajuće.

Na slici [Slika 4.2] su prikazani grafovi preciznosti i odziva modela. Graf preciznosti prikazuje koliko su točna predviđanja klasa po epohama učenja. Preciznost testiranja je 46.84%, a odziv testiranja je 44.86%. Odziv prikazuje sposobnost modela da uoči pozitivne uzorke.



Slika 4.1. Grafovi točnosti i gubitka.



Slika 4.2. Grafovi preciznosti i odziva.

Matrica konfuzije pokazuje ovisnost točnih klasa i predviđenih klasa. Glavna dijagonala matrice prikazuje ispravno predviđene klase. Cilj je imati što manju raspodjelu vrijednosti izvan glavne dijagonale. Na slici [Slika 4.3] je prikazan kod za izračun matrice, a na slici [Slika 4.4] kod za prikaz matrice.

Matrica je prikazana na slici [Slika 4.5]. Nepregledna je zbog prevelikog broja klasa (100), zato izrađujemo i izvještaj o klasifikaciji.

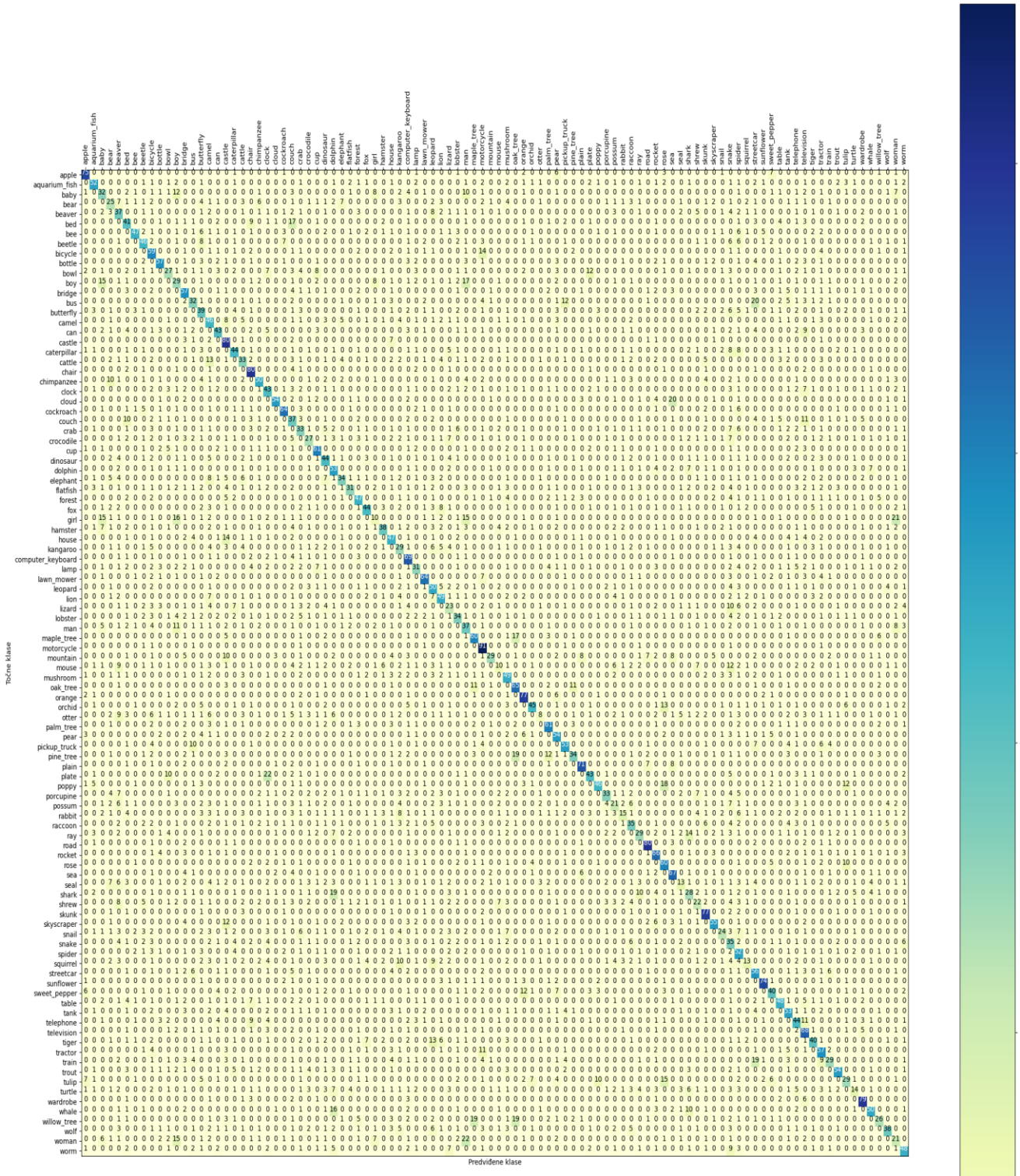
U izvještaju o klasifikaciji preglednije se mogu vidjeti točnost, preciznost, odziv i *f*1 rezultat raspoređeni po klasama. Prikazan je na slikama [Slika 4.6], [Slika 4.7] i [Slika 4.8].

```
#Predviđene oznake
y_pred = model.predict(x_test)
y_pred = np.argmax(y_pred, axis=1)

print(y_pred)

#Točne oznake
y_true = np.argmax(y_test, axis=1)

print(y_true)

#Matrica konfuzije
labels=class_names

matrica_konfuzije = confusion_matrix(y_true, y_pred)
print(matrica_konfuzije)
```

Slika 4.3. Kod za izračun matrice konfuzije.

```
threshold = matrica_konfuzije.max() / 2.

def heatmap(data, row_labels, col_labels, ax=None, cbar_kw={}, cbarlabel="", **kwargs):
    if not ax:
        ax = plt.gca()
    im = ax.imshow(data, **kwargs)

    cbar = ax.figure.colorbar(im, ax=ax, **cbar_kw)
    cbar.ax.set_ylabel(cbarlabel, rotation=-90, va="bottom")

    ax.tick_params(top=True, bottom=False, labeltop=True, labelbottom=False)

    ax.set_xticks(np.arange(data.shape[1]))
    ax.set_yticks(np.arange(data.shape[0]))

    ax.set_xticklabels(col_labels, rotation=90)
    ax.set_yticklabels(row_labels)

    ax.set_xlabel('Predviđene klase')
    ax.set_ylabel('Točne klase')

    return im, cbar

def annotate_heatmap(im, data=None, fmt="d", threshold=None):
    texts=[]
    for i in range(data.shape[0]):
        for j in range(data.shape[1]):
            text=im.axes.text(j, i, format(data[i,j], fmt), horizontalalignment="center",
                              color="white" if data[i,j] > threshold else "black")
            texts.append(text)
    return texts


fig,ax = plt.subplots(figsize=(25,25))
im, cbar = heatmap(matrica_konfuzije, labels, labels, ax=ax, cmap="YlGnBu", cbarlabel="broj predviđanja")
texts=annotate_heatmap(im, data=matrica_konfuzije, threshold=threshold)

fig.tight_layout()
plt.show()
```

Slika 4.4. Kod za prikaz matrice konfuzije.



Slika 4.5. Matrica konfuzije.

	precision	recall	f1-score	support
0	0.79	0.65	0.71	100
1	0.67	0.46	0.54	100
2	0.45	0.23	0.30	100
3	0.38	0.12	0.18	100
4	0.29	0.16	0.21	100
5	0.39	0.48	0.43	100
6	0.63	0.54	0.58	100
7	0.68	0.48	0.56	100
8	0.57	0.43	0.49	100
9	0.66	0.52	0.58	100
10	0.28	0.27	0.28	100
11	0.28	0.22	0.25	100
12	0.47	0.51	0.49	100
13	0.32	0.47	0.38	100
14	0.42	0.36	0.39	100
15	0.24	0.50	0.33	100
16	0.48	0.40	0.43	100
17	0.61	0.68	0.64	100
18	0.56	0.34	0.42	100
19	0.34	0.38	0.36	100
20	0.50	0.76	0.60	100
21	0.74	0.51	0.60	100
22	0.36	0.44	0.39	100
23	0.65	0.64	0.65	100
24	0.80	0.56	0.66	100
25	0.37	0.28	0.32	100
26	0.28	0.42	0.33	100
27	0.27	0.36	0.31	100
28	0.70	0.62	0.66	100
29	0.41	0.34	0.37	100
30	0.38	0.56	0.45	100
31	0.61	0.27	0.38	100
32	0.47	0.47	0.47	100
33	0.48	0.53	0.50	100
34	0.37	0.25	0.30	100
35	0.28	0.31	0.29	100
36	0.48	0.46	0.47	100
37	0.36	0.53	0.43	100
38	0.26	0.24	0.25	100
39	0.46	0.72	0.56	100
40	0.55	0.36	0.44	100
41	0.67	0.63	0.65	100
42	0.29	0.44	0.35	100
43	0.43	0.51	0.47	100
44	0.19	0.18	0.18	100
45	0.20	0.24	0.22	100
46	0.34	0.26	0.30	100
47	0.75	0.39	0.51	100

Slika 4.6. Izvještaj o klasifikaciji 1 od 3.

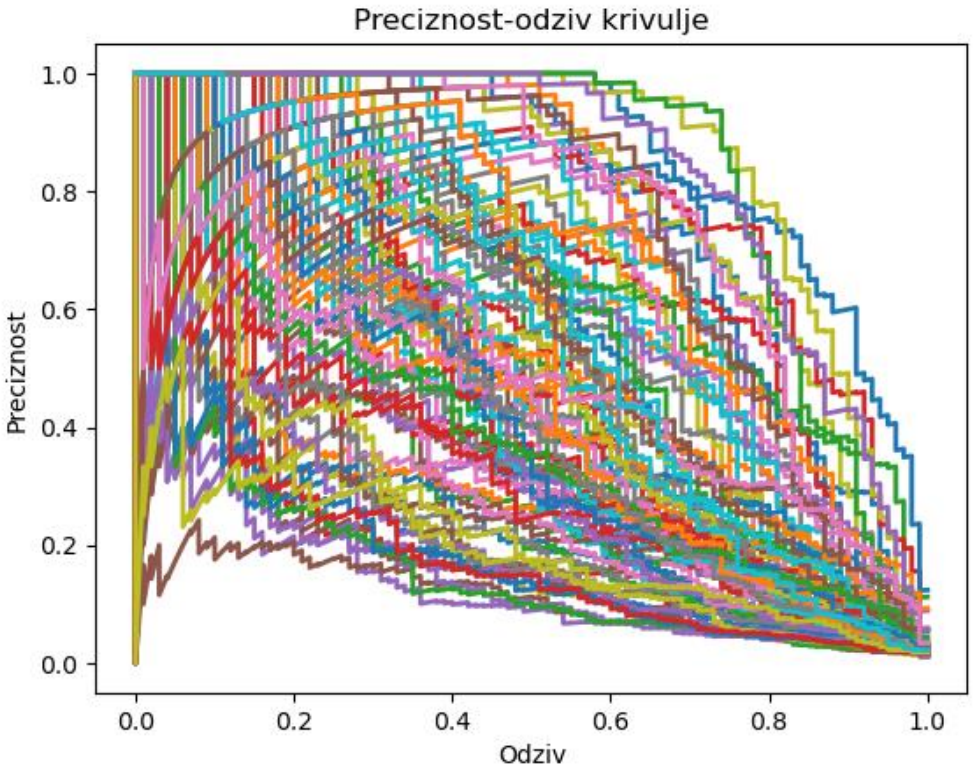
47	0.75	0.39	0.51	100
48	0.49	0.82	0.61	100
49	0.59	0.62	0.60	100
50	0.22	0.15	0.18	100
51	0.34	0.57	0.43	100
52	0.54	0.70	0.61	100
53	0.59	0.81	0.68	100
54	0.46	0.67	0.55	100
55	0.14	0.04	0.06	100
56	0.59	0.69	0.64	100
57	0.61	0.46	0.53	100
58	0.53	0.36	0.43	100
59	0.52	0.32	0.40	100
60	0.74	0.80	0.77	100
61	0.42	0.47	0.45	100
62	0.69	0.37	0.48	100
63	0.54	0.36	0.43	100
64	0.24	0.11	0.15	100
65	0.36	0.21	0.27	100
66	0.22	0.63	0.33	100
67	0.48	0.29	0.36	100
68	0.92	0.67	0.77	100
69	0.55	0.69	0.61	100
70	0.55	0.53	0.54	100
71	0.63	0.62	0.62	100
72	0.24	0.12	0.16	100
73	0.44	0.26	0.33	100
74	0.24	0.30	0.27	100
75	0.58	0.68	0.63	100
76	0.81	0.56	0.66	100
77	0.45	0.23	0.30	100
78	0.20	0.37	0.26	100
79	0.45	0.42	0.43	100
80	0.32	0.09	0.14	100
81	0.50	0.54	0.52	100
82	0.93	0.69	0.79	100
83	0.36	0.38	0.37	100
84	0.37	0.46	0.41	100
85	0.33	0.69	0.45	100
86	0.47	0.45	0.46	100
87	0.51	0.52	0.51	100
88	0.41	0.61	0.49	100
89	0.46	0.57	0.51	100
90	0.38	0.48	0.42	100
91	0.43	0.55	0.48	100
92	0.43	0.34	0.38	100
93	0.39	0.16	0.23	100
94	0.68	0.77	0.72	100
95	0.65	0.49	0.56	100

Slika 4.7. Izvještaj o klasifikaciji 2 od 3.

96	0.51	0.43	0.46	100
97	0.33	0.54	0.41	100
98	0.28	0.20	0.23	100
99	0.58	0.52	0.55	100
accuracy			0.45	10000
macro avg	0.47	0.45	0.44	10000
weighted avg	0.47	0.45	0.44	10000

Slika 4.8. Izvještaj o klasifikaciji 3 od 3.

Preciznost-odziv krivulja se može prikazati posebno za svaku klasu, kao što je na slici [Slika 4.9].Takav prikaz je nepregledan, te je bolji način za odabir najboljih vrijednosti preciznosti i odziva računanje *f*1 rezultata. Krivulja se može prikazati i kao zajednička za sve klase. Na slici [Slika 4.10] je prikazana ta krivulja te mikro-prosječna preciznost svih klasa, AP = 48%.



Slika 4.9. Preciznost-odziv krivulje klasa.

Slika 4.10. Zajednička preciznost-odziv krivulja klasa.

Srednja prosječna preciznost (*mAP)* izračunata je s kodom [Slika 4.11]. Njena vrijednost je 58.14%.

```
#Srednja prosječna preciznost

mAP = np.mean(average_precision[i])

print('mAP:', mAP)
```

```
mAP: 0.5814210650206724
```

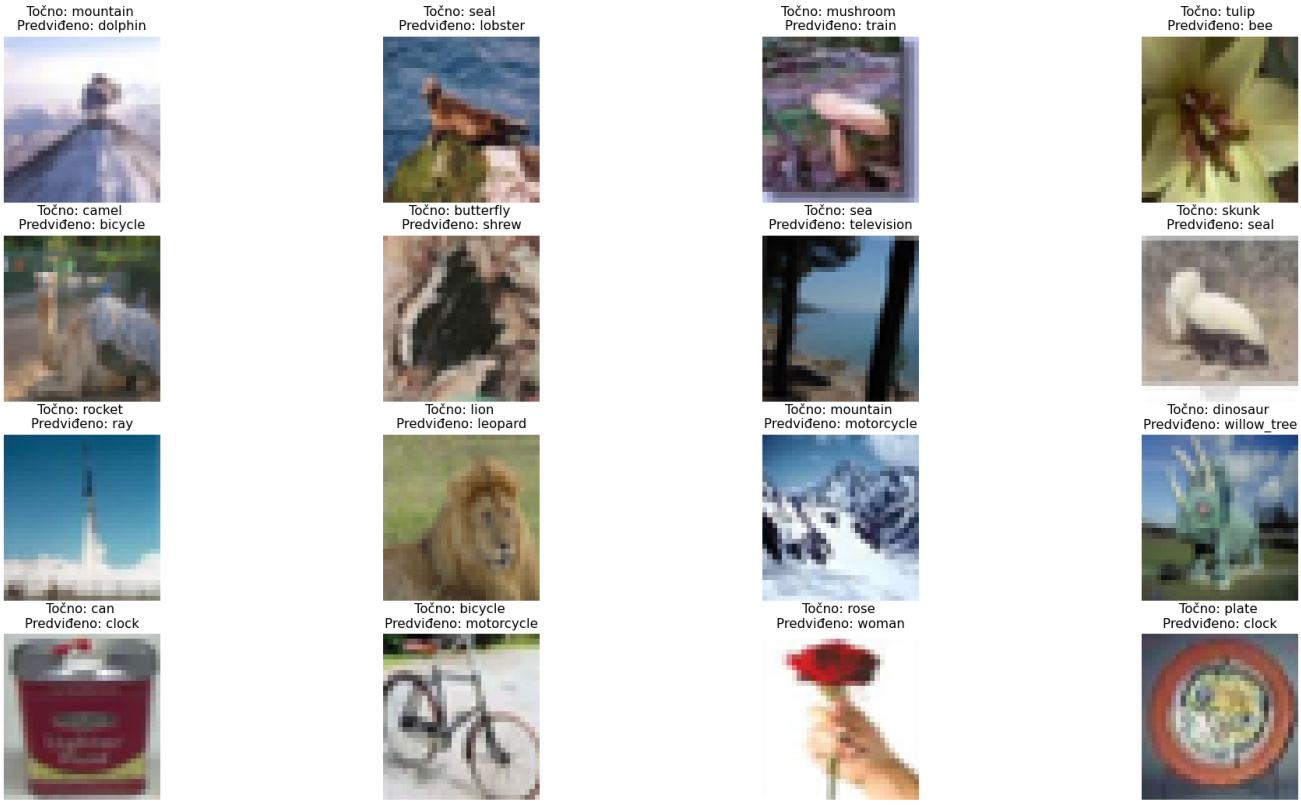
Slika 4.11. Srednja prosječna preciznost.

Pomoću korištenih evaluacijskih tehnika zaključujemo da se najtočnije predviđaju klase: jabuka, stolica, krokodil, muškarac, bor, raža i vjeverica. Matrica konfuzije prikazuje da je najveći broj predviđanja za klase: motocikl, dvorac, stolica, jabuka.

Na slici [Slika 4.12] se nalazi kod s kojim je prikazano 16 slika sa pogrešnim predviđenim oznakama. Uz slike su navedene točne i predviđene oznake [Slika 4.13].

```
row = 4
column = 4
fig, axes = plt.subplots(row, column, figsize=(22,12))
axes = axes.ravel()

misclassified_idx = np.where(y_pred != y_true)[0]
for i in np.arange(0, row*column):
    axes[i].imshow(x_test[misclassified_idx[i]])
    axes[i].set_title("Točno: %s \nPredviđeno: %s" % (labels[y_true[misclassified_idx[i]]],
                                                     labels[y_pred[misclassified_idx[i]]]))
    axes[i].axis('off')
    plt.subplots_adjust(wspace=1)

plt.show()
```

Slika 4.12. Kod za prikaz pogrešno predviđenih oznaka slika.



Slika 4.13. Slike sa točnim i predviđenim oznakama.

4.1. Usporedba rezultata s postojećim člancima

Dobivena točnost ovim radom od približno 45% nije loš rezultat s obzirom da se testira rad mreže od 60 000 slika podijeljenih u 100 klasa. Po klasi ima 500 slika za treniranje što je premalo za izradu mreže koja bi vrlo precizno predvidila klase. Uspoređujući rad sa člankom [34], točnost dobivena njihovim radom mreže je za 10% manja. Razlog tome je što nisu uveli ImageDataGenerator koji zamjenjuje seriju slika novom, nasumično transformiranom serijom i time povećava točnost testiranja mreže.

U članku [35] korišten je suvremeni model EfficientNet-B0 prethodno treniran na skupu podataka ImageNet. Taj model povećava točnost rada mreže i poboljšava učinkovitost modela smanjenjem broja parametara. Dobivena je točnost testiranja 81.79% i time primjećujemo da uvođenje modela EfficientNet značajno povećava točnost.

ResNet je umjetna neuronska mreža koja se koristi kao model u konvolucijski neuronskim mrežama. U članku [36] koristi se model ResNet34 i dobivena je točnost 66% .

5. ZAKLJUČAK

U ovom radu je prikazan rad konvolucijske neuronske mreže na skupu podataka CIFAR-100. Opisana je teorijska osnova potrebna za razumijevanje izrade mreže. Mreža je izrađena i testirana na temelju potrebnih Python biblioteka. Pomoću evaluacijskih alata za analizu rada mreže određena je njena točnost od 45%. U matrici konfuzije i izvještaju klasifikacije se može vidjeti koliko je ispravno predviđenih klasa te njihova preciznost i odziv. Prikazan je primjer slika sa točnim i predviđenim oznakama.

Na temelju rezultata evaluacijskih tehnika i usporedbe sa rezultatima već postojećih istraživanja zaključeno je koja su moguća poboljšanja. Povećanju točnosti testiranja rada mreže pridonosi povećanje broja epoha za treniranje modela na stotine ili čak tisuće epoha, ali time se ujedno i povećava trajanje treniranja, zatim povećanje slikovnih podataka pomoću ImageDataGenerator te korištenje prethodno istreniranih modela, kao što su EfficientNet, ResNet, VGG-16,LeNet-5 i AlexNet. Moguće su i promjene ostalih hiperparametara, ali to ostaje za buduća istraživanja.

LITERATURA

[1] https://deepomatic.com/en/what-is-image-recognition, dostupno dana 21.kolovoza 2021.

[2] https://www.ibm.com/cloud/learn/machine-learning, dostupno dana 21.kolovoza 2021.

[3] https://www.ibm.com/cloud/learn/supervised-learning, dostupno dana 21.kolovoza 2021.

[4] https://www.ibm.com/cloud/learn/unsupervised-learning, dostupno dana 21.kolovoza 2021.

[5] https://www.geeksforgeeks.org/what-is-reinforcement-learning/, dostupno dana 21.kolovoza 2021.

[6] https://www.ibm.com/cloud/learn/deep-learning, dostupno dana 21.kolovoza 2021.

[7] https://flatironschool.com/blog/deep-learning-vs-machine-learning, dostupno dana 21.kolovoza 2021.

[8] Machine-Learning-Vs-Deep-Learning.png, dostupno dana 21.kolovoza 2021.

[9] https://www.investopedia.com/terms/a/artificial-neural-networks-ann.asp, dostupno dana 21.kolovoza 2021.

[10] Tačković, K., Nikolovski, S., Kratkoročno prognoziranje opterećenja …, Energija, god. 57(2008), No. 5, pp. 560-579

[11]https://smhatre59.medium.com/what-is-the-relation-between-artificial-and-biological-neuron-18b05831036, dostupno dana 21.kolovoza 2021.

[12] B.D.Bašić, M.Čupić, J.Šnajder, "Umjetne neuronske mreže", Zagreb: Fakultet elektrotehnike i računarstva, 2008.

[13] https://www.ibm.com/cloud/learn/convolutional-neural-networks, dostupno dana 21.kolovoza 2021.

[14]https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53, dostupno dana 21.kolovoza 2021.

[15] https://anhreynolds.com/blogs/cnn.html, dostupno dana 21.kolovoza 2021.

[16] https://www.researchgate.net/figure/ReLU-activation-function_fig7_333411007, dostupno dana 21.kolovoza 2021.

[17] https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-4-full-connection, dostupno dana 21.kolovoza 2021.

[18]https://heartbeat.fritz.ai/introduction-to-machine-learning-model-evaluation-fa859e1b2d7f, dostupno dana 22.kolovoza 2021.

[19] https://www.datavedas.com/holdout-cross-validation/, dostupno dana 22.kolovoza 2021.

[20] https://medium.datadriveninvestor.com/k-fold-cross-validation-6b8518070833, dostupno

dana 22. kolovoza 2021.

[21] https://blog.paperspace.com/deep-learning-metrics-precision-recall-accuracy/, dostupno dana 22.kolovoza 2021.

[22] https://blog.paperspace.com/mean-average-precision/, dostupno dana 22.kolovoza 2021.

[23]https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/, dostupno dana 22.kolovoza 2021.

[24] http://ronny.rest/tutorials/module/localization_001/iou/, dostupno dana 22.kolovoza 2021.

[25] https://www.cs.toronto.edu/~kriz/cifar.html, dostupno dana 23.kolovoza 2021.

[26] https://docs.python.org/3/faq/general.html#what-is-python, dostupno dana 23.kolovoza 2021.

[27] https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html, dostupno dana 23.kolovoza 2021.

[28] https://keras.io/about/, dostupno dana 23.kolovoza 2021.

[29] https://www.activestate.com/resources/quick-reads/what-is-matplotlib-in-python-how-to-use-it-for-plotting/, dostupno dana 23.kolovoza 2021.

[30] https://numpy.org/, dostupno dana 23.kolovoza 2021.

[31] https://scikit-learn.org/stable/index.html, dostupno dana 23.kolovoza 2021.

[32] https://seaborn.pydata.org/introduction.html, dostupno dana 30.kolovoza 2021.

[33] https://www.w3schools.com/python/pandas/pandas_intro.asp, dostupno dana 30.kolovoza 2021.

[34] https://www.machinecurve.com/index.php/2020/02/09/how-to-build-a-convnet-for-cifar-10-and-cifar-100-classification-with-keras/, dostupno dana 2.rujna 2021.

[35]https://towardsdatascience.com/cifar-100-transfer-learning-using-efficientnet-ed3ed7b89af2, dostupno dana 2.rujna 2021.

[36] https://blog.jovian.ai/classifying-cifar-100-with-resnet-5860a9c2c13f, dostupno dana 2.rujna 2021.

PRILOZI

I. Python kod

I. Python kod

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import sklearn.metrics
import seaborn as sns
import pandas as pd

from tensorflow import keras
from tensorflow.keras import datasets, layers, models
from tensorflow.keras.datasets import cifar100
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D
from tensorflow.keras.losses import categorical_crossentropy
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.metrics import Precision, Recall
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import precision_recall_curve, precision_score, recall_score, average_precision_score


#Učitavanje CIFAR-100 skupa podataka
(x_train, y_train), (x_test, y_test) = cifar100.load_data()

print(f"x_train shape: {x_train.shape}")
print(f"y_train shape: {y_train.shape}")
print(f"x_test shape: {x_test.shape}")
print(f"y_test shape: {y_test.shape}")


#Prikaz slika iz skupa podataka (provjera da je skup učitan)
class_names=['apple','aquarium_fish','baby','bear','beaver','bed','bee','beetle','bicycle','bottle','bowl',
             'boy','bridge','bus','butterfly','camel','can','castle','caterpillar','cattle','chair','chimpanzee',
             'clock','cloud','cockroach','couch','crab','crocodile','cup','dinosaur','dolphin','elephant',
             'flatfish','forest','fox','girl','hamster','house','kangaroo','computer_keyboard','lamp',
             'lawn_mower','leopard','lion','lizard','lobster','man','maple_tree','motorcycle','mountain',
             'mouse','mushroom','oak_tree','orange','orchid','otter','palm_tree','pear','pickup_truck',
             'pine_tree','plain','plate','poppy','porcupine','possum','rabbit','raccoon','ray','road',
             'rocket','rose','sea','seal','shark','shrew','skunk','skyscraper','snail','snake','spider',
             'squirrel','streetcar','sunflower','sweet_pepper','table','tank','telephone','television',
             'tiger','tractor','train','trout','tulip','turtle','wardrobe','whale','willow_tree','wolf',
             'woman','worm',]

superclass_names={'aquatic mammals': ['beaver', 'dolphin', 'otter', 'seal', 'whale'],
    'fish': ['aquarium_fish', 'flatfish', 'ray', 'shark', 'trout'],
    'flowers': ['orchid', 'poppy', 'rose', 'sunflower', 'tulip'],
    'food containers': ['bottle', 'bowl', 'can', 'cup', 'plate'],
    'fruit and vegetables': ['apple', 'mushroom', 'orange', 'pear',
                             'sweet pepper'],
```

```
    'household furniture': ['bed', 'chair', 'couch', 'table', 'wardrobe'],
    'insects': ['bee', 'beetle', 'butterfly', 'caterpillar', 'cockroach'],
    'large carnivores': ['bear', 'leopard', 'lion', 'tiger', 'wolf'],
    'large man-made outdoor things': ['bridge', 'castle', 'house', 'road',
                                      'skyscraper'],
    'large natural outdoor scenes': ['cloud', 'forest', 'mountain', 'plain',
                                     'sea'],
    'large omnivores and herbivores': ['camel', 'cattle', 'chimpanzee',
                                       'elephant', 'kangaroo'],
    'medium-sized mammals': ['fox', 'porcupine', 'possum', 'raccoon', 'skunk'],
    'non-insect invertebrates': ['crab', 'lobster', 'snail', 'spider', 'worm'],
    'people': ['baby', 'boy', 'girl', 'man', 'woman'],
    'reptiles': ['crocodile', 'dinosaur', 'lizard', 'snake', 'turtle'],
    'small mammals': ['hamster', 'mouse', 'rabbit', 'shrew', 'squirrel'],
    'trees': ['maple_tree', 'oak_tree', 'palm_tree', 'pine_tree',
              'willow_tree'],
    'vehicles 1': ['bicycle', 'bus', 'motorcycle', 'pickup_truck', 'train'],
    'vehicles 2': ['lawn_mower', 'rocket', 'streetcar', 'tank', 'tractor']}

plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(x_train[i])
    plt.xlabel(class_names[y_train[i][0]])
plt.show()


#Normaliziranje podataka u raspon [-1,1]
x_train = x_train.astype('float32') /255
x_test = x_test.astype('float32') /255

#Pretvaranje vektora klase u matricu klase
y_train = to_categorical(y_train, 100)
y_test = to_categorical(y_test, 100)

#Odredivanje ulaznog oblika podataka
img_width, img_height, img_no_channels = 32, 32, 3
input_shape = (img_width, img_height, img_no_channels)


#Konfiguracija modela
no_classes = 100  #broj klasa
no_epochs = 20  #broj epoha (ponavljanja) za trening
optimizer = keras.optimizers.Adam(learning_rate=0.001) #metoda kojom ažuriramo težine neuronske mreže
validation_split = 0.2 #20% podataka za trening se koristi u svrhu provjere valjanosti
verbosity = 1 #opširnost, 1=True, 0=False
```

```
batch_size = 50 #veličina serije = količina uzoraka koji odjednom idu u model i nakon njih se računa gubitak
loss_function = categorical_crossentropy #funkcija gubitka za usporedbu predviđanja s istinom

#Kreiranje modela
model = Sequential()
model.add(Conv2D(32, kernel_size=(3,3), activation='relu', input_shape=input_shape, padding="same"))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(64, kernel_size=(3,3), activation='relu', padding="same"))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(128, kernel_size=(3,3), activation='relu', padding="same"))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Flatten()) #pretvara višedimenzionalne podatke u 1D

model.add(Dense(256, activation='relu')) #omogućavaju klasifikaciju
model.add(Dense(128, activation='relu'))
model.add(Dense(no_classes, activation='softmax'))

model.summary()


#Sastavljanje modela


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


model.compile(loss= loss_function,optimizer= optimizer, metrics=metrics)

data_generator = ImageDataGenerator(width_shift_range=0.1, height_shift_range=0.1,fill_mode='nearest',horizontal_flip=True, rotation_range=10)

train_generator = data_generator.flow(x_train, y_train)

steps_per_epoch = x_train.shape[0]// batch_size

history = model.fit(train_generator, steps_per_epoch = steps_per_epoch, validation_data=(x_test,y_test), verbose=1,
                    epochs=no_epochs, shuffle=True)

#Evaluacija modela
score = model.evaluate(x_test, y_test)
print(" Točnost testiranja:", +score[1])
```

```
#Graf točnosti
plt.figure(figsize=(18,8))

plt.subplot(1,2,1)

plt.plot(history.history['accuracy'], label='train')
plt.plot(history.history['val_accuracy'], label='test')
plt.xlabel('Epohe')
plt.ylabel('Točnost')
plt.legend()
plt.title('Model točnosti')


#Graf gubitka
plt.subplot(1,2,2)

plt.plot(history.history['loss'], label='train')
plt.plot(history.history['val_loss'], label='test')
plt.xlabel('Epohe')
plt.ylabel('Gubitak')
plt.legend()
plt.title('Model gubitka')


plt.show()

#Graf preciznosti
plt.figure(figsize=(18,8))

plt.subplot(1,2,1)

plt.plot(history.history['precision'], label='train')
plt.plot(history.history['val_precision'], label='test')
plt.xlabel('Epohe')
plt.ylabel('Preciznost')
plt.legend()
plt.title('Model preciznosti')

#Graf odziva
plt.subplot(1,2,2)

plt.plot(history.history['recall'], label='train')
plt.plot(history.history['val_recall'], label='test')
plt.xlabel('Epohe')
plt.ylabel('Odziv')
plt.legend()
plt.title('Model odziva')

plt.show()
```

```
#Predviđene oznake
y_pred = model.predict(x_test)
y_pred = np.argmax(y_pred, axis=1)

print(y_pred)

#Točne oznake
y_true = np.argmax(y_test, axis=1)

print(y_true)

#Matrica konfuzije
labels=class_names

matrica_konfuzije = confusion_matrix(y_true, y_pred)
print(matrica_konfuzije)

threshold = matrica_konfuzije.max() / 2.

def heatmap(data, row_labels, col_labels, ax=None, cbar_kw={}, cbarlabel="", **kwargs):
    if not ax:
        ax = plt.gca()
    im = ax.imshow(data, **kwargs)

    cbar = ax.figure.colorbar(im, ax=ax, **cbar_kw)
    cbar.ax.set_ylabel(cbarlabel, rotation=-90, va="bottom")

    ax.tick_params(top=True, bottom=False, labeltop=True, labelbottom=False)

    ax.set_xticks(np.arange(data.shape[1]))
    ax.set_yticks(np.arange(data.shape[0]))

    ax.set_xticklabels(col_labels)
    ax.set_yticklabels(row_labels)

    ax.set_xlabel('Predviđene klase')
    ax.set_ylabel('Točne klase')

    return im, cbar

def annotate_heatmap(im, data=None, fmt="d", threshold=None):
    texts=[]
    for i in range(data.shape[0]):
        for j in range(data.shape[1]):
            text=im.axes.text(j, i, format(data[i,j], fmt), horizontalalignment="center",
                              color="white" if data[i,j] > threshold else "black")
            texts.append(text)
    return texts
```

```
fig,ax = plt.subplots(figsize=(25,25))
im, cbar = heatmap(matrica_konfuzije, labels, labels, ax=ax, cmap="YlGnBu", cbarlabel="broj predviđanja")
texts=annotate_heatmap(im, data=matrica_konfuzije, threshold=threshold)

fig.tight_layout()
plt.show()

#Izrada izvještaja klasifikacije
print(classification_report(y_true, y_pred))

#Preciznost-odziv krivulja

y_score = model.predict(x_test)

precisions = dict()
recalls = dict()
average_precision = dict()

for i in range(no_classes):
    precisions[i], recalls[i], _ = precision_recall_curve(y_test[:,i],
                                                          y_score[:,i])
    average_precision[i] = average_precision_score(y_test[:, i], y_score[:, i])
    plt.plot(recalls[i], precisions[i], lw=2, label='klasa{}'.format(i))

plt.xlabel("Odziv")
plt.ylabel("Preciznost")
plt.legend(loc="best")
plt.title("Preciznost-odziv krivulje")
plt.show()

#Prosječna preciznost

print('Prosječna preciznost po klasama:', average_precision)

precisions["micro"], recalls["micro"], _ = precision_recall_curve(y_test.ravel(),
    y_score.ravel())
average_precision["micro"] = average_precision_score(y_test, y_score, average="micro")

plt.figure()
plt.step(recalls['micro'], precisions['micro'], where='post')

plt.xlabel('Odziv')
plt.ylabel('Preciznost')
plt.ylim([0.0, 1.05])
plt.xlim([0.0, 1.0])
plt.title(
    'Prosječna preciznost, mikro-prosječna u svim klasama: AP={0:0.2f}'
    .format(average_precision["micro"]))
plt.show()
```

```
#Srednja prosječna preciznost

mAP = np.mean(average_precision[i])

print('mAP:', mAP)

#Prikaz slika s pogrešnim predviđenim oznakama
row = 4
column = 4
fig, axes = plt.subplots(row, column, figsize=(22,12))
axes = axes.ravel()

misclassified_idx = np.where(y_pred != y_true)[0]
for i in np.arange(0, row*column):
    axes[i].imshow(x_test[misclassified_idx[i]])
    axes[i].set_title("Točno: %s \nPredviđeno: %s" % (labels[y_true[misclassified_idx[i]]],
                                                     labels[y_pred[misclassified_idx[i]]]))
    axes[i].axis('off')
    plt.subplots_adjust(wspace=1)

plt.show()
```