# Razvoj modela umjetne inteligencije za klasifikaciju slika

**Brković, Antonio**

**SVEUČILIŠTE U ZAGREBU**

**FAKULTET ORGANIZACIJE I INFORMATIKE**

**V A R A Ž D I N**

**Antonio Brković**

# Razvoj modela umjetne inteligencije za klasifikaciju slika

**ZAVRŠNI RAD**

**Varaždin, 2024.**

**SVEUČILIŠTE U ZAGREBU**

**FAKULTET ORGANIZACIJE I INFORMATIKE**

**V A R A Ž D I N**

**Antonio Brković**

**Matični broj: 0016147871**

**Studij: Informacijski i poslovni sustavi**

# Razvoj modela umjetne inteligencije za klasifikaciju slika

**ZAVRŠNI RAD**

**Mentor/Mentorica:**

Izv. prof. dr. sc. Dijana Oreški

**Varaždin, srpanj 2024.**

*Antonio Brković*

**Izjava o izvornosti**

Izjavljujem da je moj završni rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

*Autor/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi*

_____

**Sažetak**

U ovom radu se radi o primjeni umjetne inteligenciji za razvoj modela za klasifikaciju slika. Koristio sam model konvolucijskih neuronskih mreža te sam model trenirao i testirao na slikama prometnih znakova. Cilj modela je prepoznati koji je prometni znak prikazan na slici. Rad je podijeljen na teorijski i praktični dio. U teorijskom dijelu opisane su konvolucijske neuronske mreže i duboko učenje, dok je u praktičnom dijelu objašnjen razvoj modela klasifikacije slika. Na kraju je opisano kako se klasifikacija slika može praktični primijeniti.

**Ključne riječi:** konvolucijske neuronske mreže, klasifikacija slika, model za klasifikaciju, umjetna inteligencija, baza podataka.

# Sadržaj

# 1. Uvod

Mate Rimac 2023. godine najavljuje „Project 3 Mobility“ s kojim je obećao autonomna vozila na ulicama Zagreba. Autonomna vožnja ima različite faktore koji moraju biti ostvareni kako bi vožnje bile uspješne. Jedan od tih faktora je uspješno prepoznavanje prometnih znakova. Kamere na automobilima moraju uslikati prometne znakove i u stvarnom vremenu ih klasificirati, prepoznati, kako bi vozilo moglo reagirati na to što prometni znak zahtjeva. Uspješnost te klasifikacije može spasiti mnoge živote.

Ovaj rad će se baviti razvojem modela umjetne inteligencije za klasifikaciju slika, konkretno prometnih znakova. Koristit ću jednu metodu te ću se u radu baviti teorijskom i praktičnom primjenom navedenog modela. Nakon što model bude dovoljno precizan razvit ću i korisničko sučelje unutar kojeg ćemo moći ubaciti bilo koju sliku prometnog znaka te testirati sami je li model točno prepoznao prometni znak.

Model naravno ne mora biti isključivo za prepoznavanje prometnih znakova te se može razviti na drugim skupovima podataka, npr. prepoznavanje tumora, ali uspjeh našeg modela bi mogao biti koristan ne samo za autonomnu vožnju već i za različite aspekte unutar svakodnevne vožnje.

# 2. Strojno učenje

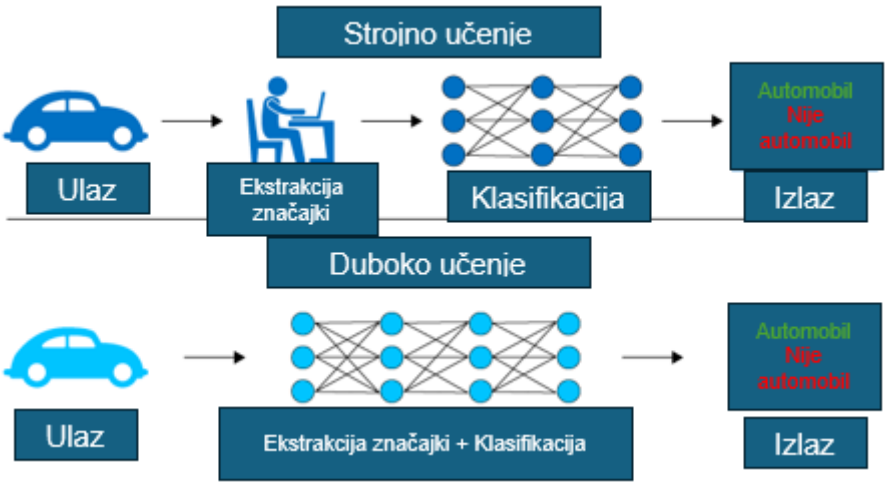
Strojno učenje je vrsta algoritma koja je dizajnirana da oponaša ljudsku inteligenciju tako što uči iz svoje okoline [1] .Uspješnost strojnog učenja ovisi o mogućnosti da model može naučiti iz iskustva, identificirati obrasce i skoro pa samostalno donositi odluke. Dijeli se na tri glavne kategorije, a one su nadzirano učenje, nenadzirano učenje i „reinforcement learning“.

Danas ima upotrebu u različitim granama kao što su financije, biomedicina, medicina, računalni vid i slično.

## 2.1. Duboko učenje

Duboko učenje je grana umjetne inteligencije koja uz pomoć velikih modela neuronskih mreža (NNM) donosi precizne odluke na bazi dobivenih podataka [2]. Modeli kreirani uz pomoć dubokog učenja su najuspješniji na velikim skupovima podataka.

Slika 1 prikazuje razlike između strojnog učenja i dubokog učenja na primjeru klasifikacije automobila. U slučaju strojnog učenja prvi korak je unos podataka, u ovom slučaju slike automobila. Zatim slijedi izdvajanje značajki, što je proces u kojem stručnjak definira koje značajke slike (npr. rubovi, boje, oblici) će se koristiti za daljnju analizu. Ove izdvojene značajke zatim ulaze u klasifikacijski algoritam, najčešće neuronsku mrežu, koja na temelju tih značajki donosi odluku o tome je li na slici automobil ili nije (output). Prvi korak je isti i za duboko učenje, ali nakon toga se javlja glavna razlika između strojnog i dubokog učenja. Dok unutar strojnog učenja moramo imati stručnjaka koji će definirati značajke, duboko učenje samo uči koje su značajke bitne za zadatak klasifikacije. Jedna od glavnih prednosti dubokog učenja je njegova sposobnost da automatski izvuče relevantne značajke iz sirovih podataka, što smanjuje potrebu za ručnim radom i stručnim znanjem. To također znači da duboko učenje može raditi s mnogo većim i složenijim skupovima podataka. S druge strane, duboke neuronske mreže zahtijevaju mnogo više računalnih resursa i vremena za treniranje.




Slika 1 Usporedba strojnog učenja i Dubokog učenja
(Sruthy, 2024)

Duboko učenje koristi slojevite strukture, gdje svaki sloj uči različite razine apstrakcije iz podataka. Na primjer, u prepoznavanju automobila, prvi sloj može naučiti prepoznavati rubove, drugi sloj može naučiti prepoznavati jednostavne oblike, a treći sloj može kombinirati ove oblike u prepoznavanje složenijih objekata [3], [4] . Zbog sposobnosti takvog načina učenja duboko učenje je izuzetno bitno za analizu složenih i nestrukturiranih podataka.

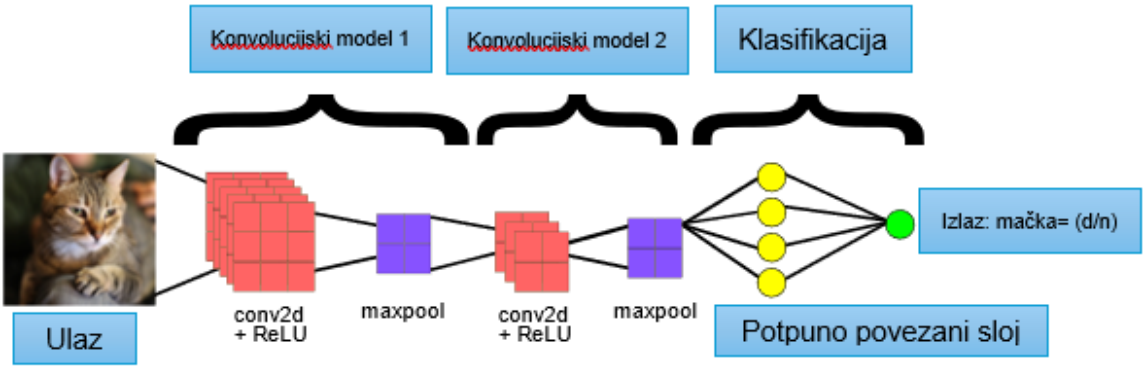
Iako je prvi model dubokog učenja kreiran već 1967. godine, Alexey Ivaknenko i Lapa, duboko učenje postaje popularno tek ranih 2000. godina kada je CNN model se koristio za prepoznavanje oko 10% čekova u SAD-u, a takozvana „Deep learning revolucija“ počinje početkom prošlog desetljeća [5]. Možemo reći da je dosadašnji vrhunac dostignut 2019. godine kada su Yoshua Bengio, Geoffrey Hinton i Yann LeCun osvojili Turingovu nagradu.

Kao što sam rekao danas mu je primjena široka, u medinici se koristi za dijagnozu različitih bolesti (npr. prepoznavanje tumora), u kontekstu autonomne vožnje koristi se kao takozvani „vid“ vozila (npr. prepoznavanje prometnih znakova). Postoje različiti modeli dubokog učenja, no nama je najbitniji CNN.

Kako se danas svijet razvija u smjeru svakodnevnog korištenja umjetne inteligencije, sa sigurnošću mogu reći da će duboko učenje biti veliki dio naše budućnosti pogotovo u navedenim industrijama.

## 2.2. Konvolucijske neuronske mreže (CNN)

Klasifikacija slika jedna je od najvažnijih primjena konvolucijskih neuronskih mreža (CNN) u području računalnog vida. CNN modeli omogućuju automatsko prepoznavanje i kategorizaciju objekata na slikama koristeći složene algoritme za ekstrakciju značajki. Ovaj članak istražuje različite aspekte CNN modela, uključujući tradicionalne CNN modele, dilatirane CNN modele i hibridne dilatirane CNN (HDC) modele, koristeći različite izvore za detaljnu analizu i prikaz rezultata.




Slika 2 Primjer CNN modela
(Govinda Dumane, 2020)

CNN modeli su posebno dizajnirani za obradu podataka u obliku rešetke, poput slika, kroz više slojeva konvolucijskih filtera. Svaki sloj CNN-a izvlači različite razine apstrakcije iz slike, od jednostavnih rubova i tekstura u početnim slojevima do složenih objekata u kasnijim slojevima. Najčešći pristup uključuje korištenje konvolucijskih slojeva, aktivacijskih funkcija poput ReLU, te slojeva za maksimalno izdvajanje (engl. max-pooling) kako bi se smanjila dimenzionalnost podataka zadržavajući najvažnije značajke.

Razvoj CNN-a može se pratiti unatrag do 1990-ih kada su LeCun [3] i suradnici razvili LeNet-5, koji je postavio temelje za moderne CNN-e. LeNet-5 sastoji se od dva konvolucijska sloja praćena slojevima za maksimalno izdvajanje i tri potpuno povezana sloja. Ovaj model pokazao je izvrsne rezultate na zadacima prepoznavanja rukom pisanih znamenki.

Kasnije, 2012. godine, Krizhevsky i suradnici predstavili su AlexNet[7], [8], dublji i širi model LeNet-5, koji je osvojio ImageNet natjecanje te godine i pokrenuo istraživački val u području CNN-a. AlexNet je uveo nekoliko ključnih inovacija, uključujući korištenje ReLU aktivacijske funkcije, metode normalizacije i dropout-a za sprječavanje prenaučenosti.

Godine 2014., istraživači s Oxford University razvili su VGGNet[8], koji je koristio manje 3x3 konvolucijske filtere umjesto većih filtera, što je rezultiralo boljom izvedbom mreže uz manji broj parametara. VGGNet je osvojio prvo mjesto u lokalizaciji i drugo mjesto u klasifikaciji na ILSVRC natjecanju.

Dilatirani konvolucijski slojevi uvode praznine (engl. holes) između vrijednosti u konvolucijskom kernelu, čime se povećava receptive field bez povećanja broja parametara[6]. Ovaj pristup omogućava modelu da "vidi" šire područje ulazne slike i tako poboljša performanse u zadacima prepoznavanja objekata i klasifikacije.

Dilatirani CNN modeli su posebno korisni u situacijama gdje je važno održavati visoku rezoluciju značajki kroz više slojeva mreže. Prema istraživanjima, dilatirani CNN modeli mogu smanjiti vrijeme treniranja i povećati točnost klasifikacije u usporedbi s tradicionalnim CNN modelima[9].

Primjena dilatiranih konvolucijskih slojeva nije ograničena samo na klasifikaciju slika. Ovi modeli pokazali su se korisnima i u drugim područjima poput segmentacije slika, prepoznavanja zvukova i super-rezolucije slika. Na primjer, Zhang i suradnici predložili su dilatirani CNN model za super-rezoluciju jedne slike, koji miješa tradicionalne i dilatirane konvolucijske filtere za postizanje bolje generalizacije.

Hibridni dilatirani CNN (HDC) modeli kombiniraju dilatirane konvolucijske slojeve s različitim stopama dilatacije kako bi se izbjegli problemi poput griding efekta, gdje se značajke mogu izgubiti zbog prevelikih praznina između vrijednosti u kernelu[10], [11]. HDC modeli omogućuju bolje prepoznavanje detalja na slici zadržavajući prednosti dilatiranih konvolucijskih slojeva.

Prema istraživanjima, HDC modeli pokazali su poboljšane rezultate u zadacima klasifikacije slika u odnosu na standardne dilatirane CNN modele, smanjujući vrijeme treniranja i povećavajući točnost treniranja i testiranja

U istraživanju "Classification of Image using Convolutional Neural Network (CNN)", autori su koristili CNN modele za automatsku klasifikaciju slika koristeći CIFAR-10 dataset, koji sadrži 60,000 slika podijeljenih u 10 klasa . Cilj je bio koristiti značajke iz višestrukih slojeva mreže za poboljšanje točnosti klasifikacije, umjesto oslanjanja samo na značajke iz posljednjeg sloja[12].

Istraživanje je pokazalo da kombinacija značajki iz višestrukih slojeva može značajno poboljšati performanse klasifikacije, jer različiti slojevi mreže uče različite razine apstrakcije i značajki. Ova metoda omogućuje modelu da bolje iskoristi informacije sadržane u ulaznim slikama, što rezultira točnijim i robusnijim klasifikacijama.

Konvolucijske neuronske mreže, posebno dilatirani i hibridni dilatirani modeli, pružaju moćne alate za klasifikaciju slika. Kombinacija različitih tehnika, kao što su dilatacija i fuzija značajki iz višestrukih slojeva, može značajno poboljšati performanse modela u zadacima prepoznavanja i klasifikacije slika. Daljnja istraživanja i razvoj u ovom području mogu donijeti još naprednije modele koji će biti sposobni za još precizniju i bržu analizu slika.

## 2.3. Metode i tehnike rada

Kako bih izradio model umjetne inteligencije za klasifikaciju slika odlučio sam cijeli projekt raditi u okruženju „Google Colab“.

Tijekom istraživanja shvatio sam da Google Colab ima veliku prednost nad drugim okruženjima zbog lake povezanosti s Google Driveom te zbog besplatnog pristupa GPU-ovima. Model ću trenirati nad skupom podataka koji sadrži preko 15000 slika, plus dodatni skup podataka za testiranje. Radi praktičnosti odlučio sam se podatke spremiti na Google Drive te okruženje spojiti s diskom umjesto da podatke lokalno spremam u okruženje. Također iz tehničkog razloga pristup besplatnim GPU-ovima mi je dosta pomogao tijekom treniranja i testiranja.

Naravno programski jezik koji sam koristio je Python te još jedna prednost koju sam imao s okruženjem jest što je većina biblioteka već bila instalirana. Koristio sam različite biblioteke kao što su numpy, pandas, os, različite tensorflow biblioteke i još mnoge druge kroz koje ćemo kasnije proći.

Sami skup podataka sam pronašao na stranici „Kaggle“, koja nudi širok raspon visokokvalitetnih skupova podataka. Glavni dio rada fokusirat ću se na razvoj CNN modela koji je vrsta dubokog učenja.

# 3. Opis problema

U dosadašnjim poglavljima prošli smo teorijski dio o dubokom učenju te smo objasnili metode I tehnike koje ćemo koristiti. U nastavku ćemo zapravo proći procese s kojim dolazimo do rješenja problemske domene. Ti procesi su: prikupljanje i priprema podataka, preprocesiranje podataka, izbor i dizajn modela, treniranje modela te evaluacija modela.

## 3.1. Prikupljanje i priprema podataka

Prikupljanje podataka je prvi i jedan od ključnih koraka u razvoju modela konvolucijskih neuronskih mreža. CNN modeli su temeljni za zadatke računalnog vida kao što su prepoznavanje objekata, klasifikacija slika i detekcija lica. Da bi ovi modeli mogli precizno i učinkovito raditi, prikupljanje podataka mora biti pažljivo planirano i provedeno. Kvaliteta ovih podataka direktno utječe na performanse CNN modela, podaci moraju biti relevantni, točni i visoke kvalitete kako ne bi došlo do loših performansi i netočnih predikcija.

Količina podataka također mora biti velika pošto više podataka omogućuje modelu da nauči složenije i suptilnije obrasce u slikama što onda direktno utječe na njegovu performansu. Također smanjuje mogućnost da dođe do pretreniranja, gdje model dobro radi na treniranim podacima, ali loše na novim podacima.

Slika 3 Primjer loše slike za treniranje


Slika 4 Primjer dobre slike za treniranje

Raznolikost podataka je također ključna. CNN modeli trebaju biti izloženi širokom spektru primjera kako bi mogli generalizirati na nove, neviđene podatke. To znači da prikupljeni podaci trebaju sadržavati različite scenarije, kutove, osvjetljenja i pozadine.

## 3.2. Preprocesiranje podataka

Preprocesiranje podataka je sljedeći ključni korak koji osigurava da su podaci u formatu pogodnom za treniranje modela. U ovoj fazi, slike se skaliraju na standardnu veličinu koju zahtijeva model (npr. 224x224 piksela za mnoge CNN modele). Skaliranje je važno jer većina modela očekuje slike jednake veličine kao ulaz.

Normalizacija je sljedeći korak, gdje se vrijednosti piksela slika prilagođavaju na raspon od 0 do 1 ili neki drugi standardizirani raspon. Normalizacija pomaže u ubrzavanju treniranja modela i može poboljšati performanse modela smanjenjem numeričkih problema. Augmentacija podataka također je važan dio preprocesiranja. Primjenjuju se tehnike kao što su rotacija, translacija, skaliranje, horizontalno/vertikalno zrcaljenje i druge transformacije na slikama kako bi se povećao broj treniranih primjera i poboljšala generalizacija modela. Augmentacija pomaže modelu da postane robustan na različite varijacije u slikama i smanjuje rizik od pretreniranja. Na kraju ove faze, podaci se obično dijele u tri skupa: trening, validacija i testiranje.

Trening skup koristi se za učenje modela, validacijski za podešavanje hiperparametara i izbjegavanje pretreniranja, a testni za konačnu evaluaciju modela. Ova podjela omogućuje procjenu performansi modela na neviđenim podacima.

## 3.3. Izbor i dizajn modela

Izbor i dizajn modela slijede nakon preprocesiranja podataka. U ovoj fazi odlučuje se koja će se arhitektura modela koristiti i kako će se model konstruirati. U kontekstu klasifikacije slika, najčešće se koriste konvolucijske neuronske mreže (CNN).

Arhitektura modela određuje kako će slojevi modela biti povezani. Popularne arhitekture za klasifikaciju slika uključuju LeNet, AlexNet, VGG, ResNet, Inception, itd. (Wiki) Svaka od ovih arhitektura ima svoje prednosti i slabosti, a izbor arhitekture ovisi o specifičnostima problema, dostupnim resursima i željenoj točnosti. Dizajn modela uključuje definiranje slojeva modela i njihovih hiperparametara.

U CNN modelima obično se koriste konvolucijski slojevi za ekstrakciju značajki, slojevi za pooling za smanjenje dimenzionalnosti, potpuno povezani slojevi za klasifikaciju i slojevi za normalizaciju i regularizaciju kao što je Dropout. Ovi slojevi rade zajedno kako bi naučili značajke iz slika i klasificirali ih s visokim stupnjem točnosti.

## 3.4. Treniranje modela

Treniranje modela je proces prilagodbe težina modela na temelju ulaznih podataka i njihovih pripadajućih oznaka. Prije treniranja potrebno je definirati hiperparametre, koji su vrijednosti postavljene prije početka treniranja i uključuju stopu učenja, broj epoha, veličinu mini-batcha, itd. Podešavanje hiperparametara može značajno utjecati na performanse modela.

Treniranje modela uključuje prolazak modela kroz trening podatke više puta, prilagođavanje težina kako bi se minimizirao gubitak. Optimizatori kao što su SGD, Adam ili RMSprop koriste se za ažuriranje težina. Tijekom treniranja, model se redovito procjenjuje na validacijskom skupu kako bi se pratila njegova izvedba i izbjeglo pretreniranje.

Na temelju performansi na validacijskom skupu mogu se podešavati hiperparametri. Validacija pomaže u održavanju ravnoteže između pretreniranja i podtreniranja, osiguravajući da model generalizira dobro na neviđene podatke.

## 3.5. Evaluacija performansi modela

Evaluacija modela ključno je područje u razvoju sustava strojnog učenja, koje omogućava procjenu učinkovitosti modela na stvarnim podacima. U praksi, evaluacija modela najčešće uključuje nekoliko standardnih postupaka i mjera koje pomažu istraživačima i

inženjerima da razumiju koliko je njihov model učinkovit. Dvije osnovne metode koje se često koriste su točnost (accuracy) i vizualizacija performansi modela kroz epohe treniranja.

Točnost je osnovna mjera performansi modela koja mjeri omjer ispravno predviđenih primjera u odnosu na ukupan broj primjera. Točnost se računa kao:

$$Točnost = \frac{Broj\ točno\ predviđenih\ primjera}{Ukupan\ broj\ primjera}$$

Ova mjera je jednostavna i intuitivna, te pruža osnovni uvid u to koliko je model dobar u klasifikaciji podataka. Međutim, točnost može biti varljiva u slučaju neuravnoteženih podataka gdje su neke klase značajno zastupljenije od drugih. U postupku evaluacije, model prvo predviđa klase za skup testnih podataka. Zatim se predikcije uspoređuju s točnim oznakama koristeći metričku funkciju accuracy_score iz knjižnice scikit-learn. Ova funkcija vraća postotak točno predviđenih primjera, što omogućuje jednostavnu interpretaciju performansi modela.

Uz samu točnost, važno je razumjeti kako se performanse modela mijenjaju tijekom procesa treniranja. Vizualizacija performansi modela kroz epohe pruža dubinski uvid u proces učenja i prilagodbe modela. Dvije ključne metrike koje se često vizualiziraju su točnost (accuracy) i gubitak (loss).

Nakon što je model treniran i evaluiran, slijedi faza implementacije i održavanja. Implementacija uključuje integraciju modela u stvarne aplikacije ili sustave. Ovo može uključivati postavljanje modela na servere, korištenje u mobilnim aplikacijama ili implementaciju u embedded sustave. Model se može implementirati koristeći različite frameworke i alate za produkciju, kao što su TensorFlow Serving, TensorRT, ONNX, itd.

Implementacija osigurava da model može biti korišten u stvarnom svijetu za donošenje odluka na temelju slika. Nakon implementacije, potrebno je kontinuirano praćenje performansi modela kako bi se osigurala njegova učinkovitost i pouzdanost. Ovo može uključivati praćenje metrike performansi, detekciju „drifta podataka“, i ponovno treniranje modela s novim podacima kako bi se poboljšale performanse ili prilagodba promjenama u podacima. Redovito održavanje modela uključuje ažuriranja, optimizacije i upravljanje verzijama modela kako bi se osiguralo da model ostaje relevantan i učinkovit u promjenjivim uvjetima.

# 4. Programsko rješenje

Numpy se najčešće koristi za rad s nizovima i za matematičke operacije nad velikim količinama podataka, ali konkretno u mom slučaju ja koristim mogućnost navedene biblioteke za pohranu i obradu slikovnih podataka u obliku nizova.

Pandas je također popularna biblioteka u Python-u koja se koristi za manipulaciju i analizu podataka u tabličnom obliku. U mom primjeru koristim ju za učitavanje CSV datoteke.

Tensorflow se koristi samo u slučaju da radimo s dubokim neuronskim mrežama (DNN), to jest za izgradnju i treniranje modela, u ovom kodu koristili smo ga za izgradnju CNN modela.

Os se koristi za interakciju s operativnim sustavom, a ja ga koristim za navigaciju kroz datotečni sustav i učitavanje slikovnih datoteka iz direktorija.

PIL (Pillow) je biblioteka za obradu slika u Pythonu, u kodu se koristi za učitavanje, skaliranje i pretvaranje slikovnih datoteka u numpy nizove.

Sklearn (scikit-learn), biblioteka za strojno učenje koja pruža alate za modeliranje, evaluaciju i razne druge zadatke u strojnom učenju. U ovom kodu, sklearn se koristi za podjelu podataka na trenirajući i testni skup te za izračun točnosti modela.

Tensorflow.keras.utils.to_categorical je funkcija koja se koristi za pretvaranje oznaka (labela) u one-hot enkodiranje, što je potrebno za kategorijsku klasifikaciju.

Tensorflow.keras.models.Sequential omogućuje izgradnju sekvencijalnog modela, što znači da se slojevi modela dodaju jedan za drugim.

Tensorflow.keras.layers.Conv2D je konvolucijski sloj koji se koristi za ekstrakciju značajki iz slika.

Tensorflow.keras.layers.MaxPool2D je sloj za smanjenje dimenzija koji smanjuje prostornu dimenziju izlaza iz konvolucijskih slojeva.

Tensorflow.keras.layers.Dense, potpuno povezani sloj koji se koristi za klasifikaciju nakon konvolucijskih i pooling slojeva.

Tensorflow.keras.layers.Flatten, sloj koji pretvara višedimenzionalne podatke u jednodimenzionalni vektor.

Tensorflow.keras.layers.Dropout, regularizacijski sloj koji nasumično isključuje određeni postotak neurona tijekom treniranja kako bi se spriječilo prekomjerno prilagođavanje (overfitting).[13]

Ipywidgets se koristi za kreiranje interaktivnih kontrola, ti jest korisničkih sučelja.

IPython.display je modul unutar IPython paketa koji pruža funkcionalnosti za prikaz različitih vrsta podataka.

## 4.1. Skup podataka

Skup „GTSRB - German Traffic Sign Recognition Benchmark“ autora MYKOLA sam preuzeo s web stranice Kaggle [14] koja je online platforma za dijeljenje skupova podataka, a kao takva je jedna od najvećih zajednica podatkovnih znanstvenika. Unutar datoteke se nalaze 3 CSV datoteke (Test, Train i Meta) te 3 datoteke s istim imenom koje sadrže slike prometnih znakova.

Autor je podatke već podijelio, a potreba za podjelom se javlja jer svaki od tih skupova se koristi za druge potrebe. Skup Train se prvi koristi i kako ime govori koristi se za treniranje modela. Tijekom treniranja model naravno radi greške te se prilagođava prema tim greškama. Naravno model će biti bolji što je veći broj podataka na kojim se može trenirati. Dok se skup Train koristi za treniranje, skup Test se naravno koristi za testiranje modela, to jest za evaluaciju modela. Ti podaci ne smiju biti isti kao oni na kojim je model treniran pošto bi onda došlo do prekomjerne prilagođenosti (overfitting). Meta podaci služe za validaciju modela tijekom treniranja, omogućuju nam podešavanje hiperparametara modela i pomažu nam pri sprečavanju overfitting-a. Naravno isto vrijedi i za CSV datoteke, jedina je razlika što njih koristimo zbog lakšeg pristupa podacima.

## 4.2. Priprema podataka

Nakon što smo izabrali skup podataka koji ćemo koristiti potrebno ga je pripremiti daljnju uporabu. Kako je već navedeno podaci su spremljeni u Google Drive, koji smo povezali na sljedeći način. Prije samog učitavanja podataka iz skupa stvorit ćemo par varijabli koje su bitne za kasniju upotrebu.

Sad dolazi dio gdje podatke učitavamo uz pomoć for petlje s kojom prolazimo kroz sve klase. Putanja koju koristim je zapravo putanja unutar osobnog Google Drive-a a koristeći os biblioteku dobijemo popis svih slika koje se nalaze u trenutnom direktoriju. Zatim slijedi još jedna for petlja koja se koristi kako bi prošli kroz podatke, slike, koje se nalaze u određenim klasama. Ta nam petlja ne služi samo za učitavanje slika, već ju koristimo odmah kako bi podatke skalirali na veličinu 30x30 piksela, pretvaranje tih slika u numpy niz te spremanje tih podataka u varijable koje smo pripremili. Na samom kraju imamo poruku koja će se ispisati u slučaju da dođe do neke greške.

## 4.3. Podjela podataka na skupove za treniranje i testiranje

Dolazimo do dijela gdje treba skup podijeliti na skup za treniranje i na testni skup. U našem slučaju, podjelu podataka na trenirajući i testni skup izvršili smo korištenjem funkcije train_test_split iz knjižnice scikit-learn. Ova funkcija uzima cijeli skup podataka i nasumično ga dijeli na dva dijela. Odlučili smo da 80% podataka koristimo za treniranje, dok će preostalih 20% biti rezervirano za testiranje[15]. Ovaj omjer odabran je kako bismo imali dovoljno podataka za treniranje modela, ali i dovoljnu količinu podataka za testiranje kako bismo mogli dobiti pouzdanu procjenu performansi modela.

Kada smo izvršili podjelu podataka, dobili smo četiri različita skupa: X_train, X_test, y_train i y_test. Skup X_train sadrži slike koje ćemo koristiti za treniranje modela, dok y_train sadrži pripadajuće oznake klasa za te slike. Slično tome, X_test sadrži slike koje ćemo koristiti za testiranje modela, a y_test pripadajuće oznake klasa za te slike.

```
print(data.shape, labels.shape)

X_train, X_test, y_train, y_test = train_test_split(data, labels, test_size=0.2, random_state=42)

print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)

y_train = to_categorical(y_train, 43)
y_test = to_categorical(y_test, 43)
```

```
(38950, 30, 30, 3) (38950,)
(31160, 30, 30, 3) (7790, 30, 30, 3) (31160,) (7790,)
```

Slika 5 Podjela podataka
(vlastita izrada)

Ova podjela omogućuje nam treniranje modela na velikom broju slika, dok istovremeno zadržavamo zaseban skup slika koje ćemo koristiti isključivo za procjenu njegove točnosti. Na taj način osiguravamo da naš model ne samo da dobro funkcionira na podacima na kojima je treniran, već i na potpuno novim podacima, što je ključno za njegovu primjenu u stvarnom svijetu (Slika 5).

## 4.4. Izgradnja modela

Model je izgrađen korištenjem Sequential klase iz Keras-a, što nam omogućuje dodavanje slojeva jedan po jedan, u linearnoj sekvenci. Prvi sloj koji dodajemo je konvolucijski sloj (Conv2D). Ovaj sloj je ključan za ekstrakciju značajki iz slikovnih podataka. Konvolucijski sloj koristi filtere koji prelaze preko slike i otkrivaju različite značajke, kao što su rubovi, teksture i oblici. U našem modelu, prvi konvolucijski sloj koristi 32 filtera veličine 5x5, a aktivacijska funkcija koju koristimo je relu (rectified linear unit), koja pomaže u uvođenju nelinearnosti u model.

Nakon prvog konvolucijskog sloja, dodajemo još jedan konvolucijski sloj s istim brojem filtera i veličinom filtera[16], [17]. Dodavanjem više konvolucijskih slojeva, model može naučiti složenije i apstraktnije značajke iz slika. Slijedi sloj za smanjenje dimenzija (MaxPool2D), koji smanjuje prostornu dimenziju izlaza iz konvolucijskih slojeva, što smanjuje broj parametara i računsku složenost, te pomaže u sprječavanju prekomjernog prilagođavanja. U našem modelu koristimo pooling sloj s veličinom prozora 2x2.

Kako bismo dodatno smanjili rizik od prekomjernog prilagođavanja, dodajemo Dropout sloj. Ovaj sloj nasumično isključuje određeni postotak neurona tijekom treniranja. U našem modelu, koristimo dropout s stopom od 25%, što znači da se 25% neurona isključuje tijekom svakog koraka treniranja.

Sljedeći set slojeva uključuje dva konvolucijska sloja s 64 filtera veličine 3x3, ponovno koristeći relu aktivacijsku funkciju. Nakon njih, dodajemo još jedan pooling sloj s istom konfiguracijom kao i prethodni, te još jedan dropout sloj s istom stopom isključivanja.

Nakon konvolucijskih i pooling slojeva, izlaz se spljošti korištenjem Flatten sloja, koji pretvara višedimenzionalni izlaz u jednodimenzionalni vektor. Ovaj vektor se zatim prosljeđuje potpuno povezanom (Dense) sloju s 256 neurona i relu aktivacijskom funkcijom, što omogućuje modelu da uči složene kombinacije značajki.

Na kraju, dodajemo izlazni sloj s 43 neurona (koliko imamo klasa prometnih znakova) i softmax aktivacijskom funkcijom. Softmax funkcija pretvara izlazne vrijednosti u vjerojatnosti za svaku klasu, osiguravajući da suma svih vjerojatnosti iznosi 1.

Nakon definiranja arhitekture, kompajliramo model koristeći categorical_crossentropy kao funkciju gubitka, adam kao optimizer i accuracy kao metriku. Categorical_crossentropy je standardna funkcija gubitka za probleme više-klasne klasifikacije, dok adam optimizer kombinira prednosti dvaju drugih optimizera (AdaGrad i RMSProp) za brže i učinkovitije treniranje.

```
model = Sequential()
model.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu', input_shape=X_train.shape[1:]))
model.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.25))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.25))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(rate=0.5))
model.add(Dense(43, activation='softmax'))

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

model.summary()
```

Slika 6 Izgradnja modela
(vlastita izrada)

Ovim pristupom, model je spreman za treniranje na trenirajućim podacima, gdje će učiti prepoznavati prometne znakove na temelju značajki ekstrahiranih iz slika. Evaluacija na testnim podacima omogućit će nam da provjerimo koliko je model uspješan u generalizaciji naučenih znanja na nove, neviđene podatke (Slika 6).

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        2432

 conv2d_1 (Conv2D)           (None, 22, 22, 32)        25632

 max_pooling2d (MaxPooling2  (None, 11, 11, 32)        0
 D)

 dropout (Dropout)           (None, 11, 11, 32)        0

 conv2d_2 (Conv2D)           (None, 9, 9, 64)          18496

 conv2d_3 (Conv2D)           (None, 7, 7, 64)          36928

 max_pooling2d_1 (MaxPoolin  (None, 3, 3, 64)          0
 g2D)

 dropout_1 (Dropout)         (None, 3, 3, 64)          0

 flatten (Flatten)           (None, 576)               0

 dense (Dense)               (None, 256)               147712

 dropout_2 (Dropout)         (None, 256)               0

 dense_1 (Dense)             (None, 43)                11051

=================================================================
Total params: 242251 (946.29 KB)
Trainable params: 242251 (946.29 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Slika 7 Izgled modela
(vlastita izrada)

Nakon definiranja i kompilacije konvolucijske neuronske mreže (CNN) za prepoznavanje prometnih znakova, izuzetno je korisno pregledati strukturu modela i razumjeti

pojedinosti o svakom sloju. Na slici je prikazana struktura modela zajedno s informacijama o obliku izlaza (Output Shape) i broju parametara (Param #) za svaki sloj. Ova analiza omogućuje nam bolje razumijevanje unutarnjih mehanizama modela i složenosti njegove arhitekture (Slika 7).

## 4.5. Treniranje modela

Treniranje modela dubokog učenja ključan je korak u razvoju sustava za prepoznavanje prometnih znakova. Nakon što smo definirali arhitekturu konvolucijske neuronske mreže (CNN) i pripremili podatke, pristupili smo procesu treniranja kako bismo omogućili modelu da nauči prepoznavati različite klase prometnih znakova.

Proces treniranja započeli smo definiranjem nekoliko ključnih parametara. Odlučili smo da će model trenirati kroz 15 epoha. Jedna epoha predstavlja jedan puni prolazak kroz cijeli trenirajući skup podataka. Veći broj epoha omogućuje modelu da bolje nauči obrasce iz podataka, ali također povećava vrijeme treniranja. Nadalje, postavili smo veličinu skupa (batch size) na 32. To znači da će model ažurirati svoje težine nakon svakih 32 primjera. Manje veličine skupa omogućuju bržu prilagodbu težina, ali mogu biti manje stabilne, dok veće veličine skupa daju stabilnija ažuriranja, ali zahtijevaju više memorije.

Kako bismo maksimalno iskoristili dostupne resurse, treniranje modela izvodili smo na GPU-u. GPU-ovi su izuzetno učinkoviti za izvođenje matriksnih operacija koje su temeljne za treniranje neuronskih mreža. Upotrebom GPU-a značajno smo ubrzali proces treniranja, što je posebno korisno kada se radi o složenim modelima i velikim skupovima podataka.

Sam proces treniranja modela izveli smo korištenjem funkcije fit iz Keras-a. Ova funkcija pokreće proces treniranja modela s definiranim parametrima. Funkciji smo proslijedili trenirajući skup podataka (X_train, y_train), veličinu skupa, broj epoha i validacijski skup podataka (X_test, y_test). Validacijski skup podataka koristi se za praćenje performansi modela tijekom treniranja. Na taj način možemo vidjeti kako model napreduje i prilagođava se podacima kroz vrijeme.

Tijekom treniranja, model prolazi kroz trenirajući skup podataka više puta, svaki put prilagođavajući svoje težine kako bi smanjio gubitak (loss). Gubitak se mjeri korištenjem funkcije gubitka categorical_crossentropy, koja je prikladna za probleme višeklasne klasifikacije. Uz gubitak, pratili smo i točnost (accuracy) modela kako na trenirajućem, tako i na validacijskom skupu podataka.

```
with tf.device('/GPU:0'):
    epochs = 15
    history = model.fit(X_train, y_train, batch_size=32, epochs=epochs, validation_data=(X_test, y_test))
```

```
Epoch 1/15
974/974 [==============================] - 157s 159ms/step - loss: 1.5572 - accuracy: 0.5987 - val_loss: 0.3927 - val_accuracy: 0.9005
Epoch 2/15
974/974 [==============================] - 153s 158ms/step - loss: 0.4537 - accuracy: 0.8713 - val_loss: 0.1499 - val_accuracy: 0.9616
Epoch 3/15
974/974 [==============================] - 148s 152ms/step - loss: 0.3256 - accuracy: 0.9104 - val_loss: 0.1263 - val_accuracy: 0.9692
Epoch 4/15
974/974 [==============================] - 149s 153ms/step - loss: 0.2582 - accuracy: 0.9289 - val_loss: 0.1185 - val_accuracy: 0.9712
Epoch 5/15
974/974 [==============================] - 149s 153ms/step - loss: 0.2569 - accuracy: 0.9316 - val_loss: 0.0546 - val_accuracy: 0.9872
Epoch 6/15
974/974 [==============================] - 153s 157ms/step - loss: 0.2312 - accuracy: 0.9391 - val_loss: 0.0662 - val_accuracy: 0.9819
Epoch 7/15
974/974 [==============================] - 150s 154ms/step - loss: 0.2207 - accuracy: 0.9422 - val_loss: 0.0553 - val_accuracy: 0.9860
Epoch 8/15
974/974 [==============================] - 150s 154ms/step - loss: 0.2304 - accuracy: 0.9418 - val_loss: 0.0545 - val_accuracy: 0.9856
Epoch 9/15
974/974 [==============================] - 150s 154ms/step - loss: 0.2194 - accuracy: 0.9467 - val_loss: 0.0663 - val_accuracy: 0.9816
Epoch 10/15
974/974 [==============================] - 154s 158ms/step - loss: 0.2526 - accuracy: 0.9375 - val_loss: 0.0950 - val_accuracy: 0.9739
Epoch 11/15
974/974 [==============================] - 150s 154ms/step - loss: 0.2246 - accuracy: 0.9470 - val_loss: 0.0459 - val_accuracy: 0.9891
Epoch 12/15
974/974 [==============================] - 149s 153ms/step - loss: 0.2104 - accuracy: 0.9494 - val_loss: 0.0575 - val_accuracy: 0.9866
Epoch 13/15
974/974 [==============================] - 150s 154ms/step - loss: 0.2336 - accuracy: 0.9452 - val_loss: 0.0476 - val_accuracy: 0.9893
Epoch 14/15
974/974 [==============================] - 149s 153ms/step - loss: 0.2329 - accuracy: 0.9446 - val_loss: 0.0820 - val_accuracy: 0.9800
Epoch 15/15
974/974 [==============================] - 149s 153ms/step - loss: 0.2439 - accuracy: 0.9452 - val_loss: 0.0499 - val_accuracy: 0.9858
```

Slika 8 Treniranje modela

(vlastita izrada)

Rezultati treniranja pohranjeni su u objektu history, koji sadrži podatke o gubitku i točnosti modela za svaku epohu treniranja. Ovi podaci omogućuju nam kasniju analizu performansi modela kroz vrijeme. Kako bismo vizualizirali performanse modela, koristili smo matplotlib za crtanje grafova koji prikazuju promjene točnosti i gubitka kroz epohe treniranja. Grafovi nam pomažu vizualizirati kako se model poboljšavao tijekom vremena i mogu nam dati uvid u to je li model postigao konvergenciju ili je li pretrpio prekomjerno prilagođavanje (Slika 8).

## 4.6. Vizualizacija performansi i evaluacija modela

Nakon uspješnog treniranja modela konvolucijske neuronske mreže (CNN) za prepoznavanje prometnih znakova, sljedeći ključan korak bio je vizualizacija njegovih performansi. Vizualizacija nam omogućuje dublji uvid u to kako se model ponašao tijekom procesa treniranja te kako je napredovao kroz epohe. Korištenjem grafova možemo identificirati trendove, prepoznati potencijalne probleme poput prekomjernog prilagođavanja i donijeti odluke o daljnjim poboljšanjima modela.

Tijekom treniranja, pohranili smo podatke o točnosti i gubitku modela u objekt history. Ovaj objekt sadrži povijest treniranja modela, uključujući informacije o točnosti i gubitku na trenirajućem i validacijskom skupu podataka za svaku epohu. Korištenjem tih podataka, generirali smo grafove koji prikazuju promjene u performansama modela kroz vrijeme.

Za vizualizaciju smo koristili knjižnicu matplotlib, koja je standardna alatka za crtanje grafova u Pythonu. Proces vizualizacije započeli smo definiranjem figura za prikaz grafova. Prva figura prikazuje promjene u točnosti modela, dok druga figura prikazuje promjene u gubitku modela.

U prvoj figuri prikazali smo točnost modela na trenirajućem i validacijskom skupu podataka tijekom 15 epoha. Točnost mjeri koliko je model uspješno klasificirao prometne znakove. Korištenjem plt.plot, nacrtali smo dvije linije: jednu za točnost na trenirajućem skupu (history.history['accuracy']), a drugu za točnost na validacijskom skupu (history.history['val_accuracy']). Dodali smo naslov grafu (''Accuracy''), oznake osi (''epochs'' za x-os i ''accuracy'' za y-os) te legendu koja označava linije. Na kraju, koristili smo plt.show za prikaz grafova.

Ovaj graf nam omogućuje da vidimo kako se točnost modela mijenjala kroz epohe i kako su performanse na trenirajućem skupu usporedive s onima na validacijskom skupu. Ako linije za trenirajuću i validacijsku točnost prate sličan trend, to sugerira da model dobro generalizira. Ako postoji velika razlika između njih, to može ukazivati na prekomjerno prilagođavanje.

U drugoj figuri prikazali smo promjene u gubitku modela na trenirajućem i validacijskom skupu podataka. Gubitak mjeri koliko su predikcije modela odstupale od stvarnih oznaka. Korištenjem plt.plot, nacrtali smo dvije linije: jednu za gubitak na trenirajućem skupu (history.history['loss']), a drugu za gubitak na validacijskom skupu (history.history['val_loss']).

Dodali smo naslov grafu (''Loss''), oznake osi (''epochs'' za x-os i ''loss'' za y-os) te legendu koja označava linije. Na kraju, koristili smo plt.show za prikaz grafova.

Ovaj graf nam omogućuje da vidimo kako se gubitak modela mijenjao kroz epohe i kako su performanse na trenirajućem skupu usporedive s onima na validacijskom skupu. Slično kao i kod točnosti, sličan trend između linija za trenirajući i validacijski gubitak sugerira da model dobro generalizira. Velika razlika između njih može ukazivati na prekomjerno prilagođavanje ili podprilagođavanje.

Vizualizacija performansi modela ključna je za razumijevanje ponašanja modela tijekom treniranja. Grafovi točnosti i gubitka omogućuju nam da vidimo kako se model prilagođavao podacima kroz epohe i da prepoznamo potencijalne probleme. Ovi uvidi su neprocjenjivi za daljnje poboljšanje modela i osiguranje njegove učinkovitosti u stvarnom svijetu. Kroz ove grafove, možemo donijeti informirane odluke o promjenama u arhitekturi modela, parametrima treniranja i strategijama za sprječavanje prekomjernog prilagođavanja.

Vizualizacija točnosti modela omogućila nam je da jasno vidimo koliko je model bio uspješan u klasifikaciji prometnih znakova tijekom vremena. Graf točnosti prikazuje usporedne

linije za trenirajući i validacijski skup podataka, omogućujući nam da vidimo kako su se obje točnosti mijenjale tijekom 15 epoha. S druge strane, graf gubitka dao nam je uvid u to koliko su predikcije modela odstupale od stvarnih oznaka kroz epohe. Promatrajući ove grafove, mogli smo zaključiti koliko se model dobro prilagodio trenirajućim podacima i koliko je bio sposoban generalizirati na validacijski skup.

```
plt.figure(0)
plt.plot(history.history['accuracy'], label='training accuracy')
plt.plot(history.history['val_accuracy'], label='val accuracy')
plt.title('Accuracy')
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.legend()
plt.show()

plt.figure(1)
plt.plot(history.history['loss'], label='training loss')
plt.plot(history.history['val_loss'], label='val loss')
plt.title('Loss')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.legend()
plt.show()
```

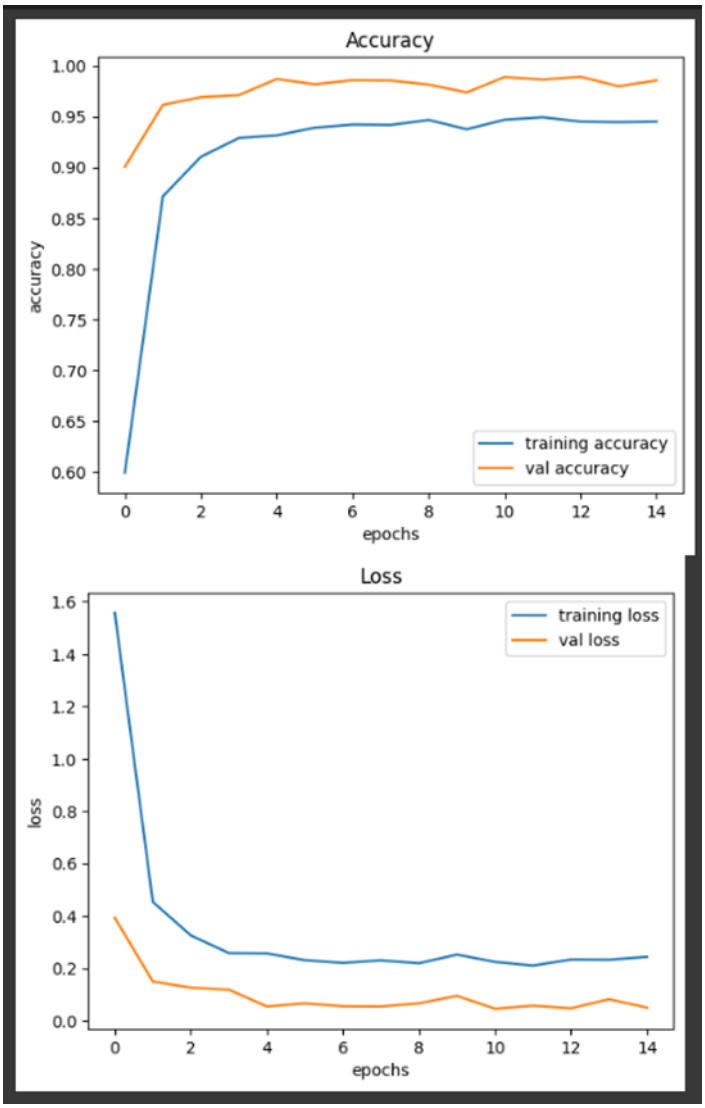
Slika 9 Vizualizacija efikasnosti
(vlastita izrada)

Gornji grafik prikazuje promjene u točnosti modela na trenirajućem i validacijskom skupu podataka tijekom 15 epoha. Plava linija predstavlja točnost na trenirajućem skupu podataka, dok narančasta linija prikazuje točnost na validacijskom skupu podataka.

Iz grafa možemo primijetiti da je točnost na trenirajućem skupu podataka rapidno porasla tijekom prvih nekoliko epoha, dosegnuvši više od 90% već nakon treće epohe. To pokazuje da je model brzo naučio prepoznavati obrasce u trenirajućim podacima. Točnost na validacijskom skupu također je pokazala značajan rast u početnim epohama, dosegnuvši oko 95% nakon treće epohe. Ovaj brzi porast točnosti sugerira da model dobro generalizira na neviđene podatke već od ranih faza treniranja.

Daljnji tijek grafa pokazuje relativno stabilne vrijednosti točnosti za oba skupa podataka. Točnost na validacijskom skupu čak je neznatno veća od točnosti na trenirajućem skupu, što može ukazivati na vrlo dobro prilagođen model. Stabilnost linija u kasnijim epohama

sugerira da model nije pretrpio prekomjerno prilagođavanje, jer nema značajnog pada točnosti na validacijskom skupu.




Slika 10 Grafovi preciznosti i gubitka

(vlastita izrada)

Donji grafik prikazuje promjene u gubitku modela na trenirajućem i validacijskom skupu podataka tijekom 15 epoha. Plava linija predstavlja gubitak na trenirajućem skupu podataka, dok narančasta linija prikazuje gubitak na validacijskom skupu podataka.

Grafik gubitka pokazuje značajan pad gubitka na oba skupa podataka tijekom prvih nekoliko epoha, što je očekivano jer model uči i prilagođava svoje težine kako bi minimizirao gubitak. Gubitak na trenirajućem skupu brzo se smanjio, dosegnuvši vrijednost ispod 0.4 nakon treće epohe. Gubitak na validacijskom skupu također je brzo opao, dosegnuvši vrijednost ispod 0.2 nakon treće epohe. U kasnijim epohama, grafici gubitka pokazuju relativno stabilne vrijednosti, što sugerira da je model dosegnuo fazu konvergencije gdje dodatne epohe

treniranja ne donose značajno smanjenje gubitka. Stabilnost gubitka na validacijskom skupu također potvrđuje da model nije pretrpio prekomjerno prilagođavanje, jer nema značajnog porasta gubitka.

Kada pričamo o točnosti modela standard točnosti je barem 85%. Taj standard je uveden još 1999. godine[18] te ga se držimo i dan danas, naravno ovisno o bitnosti modela taj minimum se može povećati. Bitnost točnog prepoznavanja prometnog znaka može spasiti život te sam taj minimum od 85% povećao na 95% koji sam na kraju i uspio ostvariti. Naravno ne možemo znati gdje nedostaje ovih 5% točnosti da model bude potpuno točan, no tijekom testiranja primijetio sam da model ne griješi na znakovima koji su dosta različiti, nego na znakovima ograničenja. Ako modelu pošaljete sliku znaka ograničenja od 50 km/h u nekim slučajevima ga prepoznaje kao ograničenje od 30 km/h i obratno. Moja pretpostavka je da model zbog sličnosti brojeva 3 i 5 ne može uvijek ih razlikovati, ako je slika niske kvalitete ili iz nekog čudnog kuta. Naravno model je treniran na 42 vrste prometnih znakova, većina vrlo različita, koji se nalaze unutar skupa podataka te možda i to utječe na tako veliku preciznost.

## 4.7. Testiranje modela

Nakon uspješnog treniranja modela konvolucijske neuronske mreže (CNN) za prepoznavanje prometnih znakova, ključni korak bio je testiranje modela na neviđenim podacima kako bismo procijenili njegovu sposobnost generalizacije. Proces testiranja modela omogućuje nam da provjerimo koliko je model točan i učinkovit u prepoznavanju prometnih znakova kada je suočen s novim, neviđenim slikama.

Testiranje modela započeli smo učitavanjem testnih podataka iz CSV datoteke. CSV datoteka sadrži putanje do testnih slika i pripadajuće oznake klasa. Koristili smo knjižnicu pandas za učitavanje CSV datoteke, što nam je omogućilo da na strukturiran način pristupimo putanjama do slika i pripadajućim klasama. Ovaj pristup omogućuje učinkovitije upravljanje podacima jer imamo centraliziran popis svih slika i njihovih oznaka.

Nakon što smo učitali testne podatke, pristupili smo učitavanju samih slika. Koristili smo knjižnicu PIL za otvaranje i skaliranje slika na odgovarajuću veličinu od 30x30 piksela. Svaka slika je zatim pretvorena u numpy niz i pohranjena u listu data. Ova lista je kasnije pretvorena u numpy niz X_test, koji predstavlja skup podataka za testiranje.

```
data=[]

with tf.device('/GPU:0'):
    for img in imgs:
        image = Image.open('/content/drive/MyDrive/Prometniznakovi/'+img)
        image = image.resize([30, 30])
        data.append(np.array(image))

X_test=np.array(data)
```

Slika 11 Skaliranje podataka na potrebnu veličinu

(vlastita izrada)

Nakon pripreme testnih podataka, koristili smo trenirani model za predikciju klasa testnih slika. Model je prošao kroz sve slike u X_test skupu i generirao predikcije u obliku vjerojatnosti za svaku klasu. Kako bismo dobili konačne predikcije, koristili smo funkciju np.argmax koja pronalazi klasu s najvećom vjerojatnošću za svaku sliku.

```
with tf.device('/GPU:0'):
    pred = np.argmax(model.predict(X_test), axis=-1)

from sklearn.metrics import accuracy_score
print(accuracy_score(labels, pred))
```

```
395/395 [==============================] - 16s 39ms/step
0.9593032462391132
```

Slika 12 Rezultati testiranja

(vlastita izrada)

Na kraju, evaluirali smo točnost modela korištenjem stvarnih oznaka (labels) i predikcija (pred). Koristili smo funkciju accuracy_score iz knjižnice sklearn kako bismo izračunali točnost modela na testnom skupu. Ova funkcija uspoređuje stvarne oznake i predikcije te izračunava postotak točno klasificiranih slika.

Rezultati testiranja pokazali su visoku točnost modela, što ukazuje na njegovu sposobnost da točno prepozna prometne znakove na neviđenim podacima. Visoka točnost na testnom skupu potvrđuje da je model dobro generalizirao i da može biti pouzdano korišten u stvarnim aplikacijama za prepoznavanje prometnih znakova.

## 4.8. Korisničko sučelje

Nakon pripreme okruženja, učitali smo prethodno trenirani model za prepoznavanje prometnih znakova. Ovaj model je treniran na skupu podataka koji sadrži slike različitih prometnih znakova.


Slika 13 Izgled korisničkog sučelja
(vlastita izrada)

Koristeći ipywidgets biblioteku, kreirali smo interaktivno korisničko sučelje. Glavni element sučelja je widget za učitavanje datoteka (FileUpload). Ovaj widget omogućava korisnicima da učitaju slike prometnih znakova iz svog lokalnog računala.[19]

Zatim smo definirali funkciju koja će se pozvati kada korisnik učita datoteku. Ova funkcija učitava sliku, prikazuje je korisniku i koristi model za predikciju klase prometnog znaka.

```
import ipywidgets as widgets
from IPython.display import display
from PIL import Image
import numpy as np
import tensorflow as tf
import io

model = tf.keras.models.load_model('/content/drive/MyDrive/Prometniznakovi/traffic_sign_model.h5')

class_names = [
    "Ograničenje brzine (20 km/h)", "Ograničenje brzine (30 km/h)", "Ograničenje brzine (50 km/h)",
    "Ograničenje brzine (60 km/h)", "Ograničenje brzine (70 km/h)", "Ograničenje brzine (100 km/h)",
    "Kraj ograničenja brzine (80 km/h)", "Ograničenje brzine (100 km/h)", "Ograničenje brzine (120 km/h)",
    "Zabrana preticanja", "Zabrana preticanja za kamione", "Opasnost (križanje sa sporednom cestom)",
    "Glavna cesta", "Dajte prednost", "STOP", "Zabrana prolaska u oba smjera", "Zabrana prolaska za kamione",
    "Jednosmjerna ulica", "Opasnost", "Opasnost (oštar zavoj lijevo)", "Opasnost (oštar zavoj desno)",
    "Opasnost (zavoji)", "Opasnost (neravnine)", "Sklizak kolnik", "Opasnost (suzavanje ceste)",
    "Radovi na cesti", "Semafor", "Opasnost (pješaci)", "Opasnost (djeca)", "Opasnost (biciklisti)",
    "Opasnost (poledica ili snijeg)", "Opasnost (divljač)", "Kraj svih ograničenja i zabrana",
    "Obvezan smjer skretanja desno", "Obvezan smjer skretanja lijevo", "Obvezan smjer ravno",
    "Obvezan smjer desno ili ravno", "Obvezan smjer lijevo ili ravno", "Obvezan smjer udesno",
    "Obvezan smjer ulijevo", "Kružni tok", "Kraj zabrane preticanja", "Kraj zabrane preticanja za kamione"
]
```

Slika 14 Kreiranje klase s imenima prometnih znakova
(vlastita izrada)

Kako bi predikcija bila razumljiva korisnicima, definirali smo popis klasa koje odgovaraju različitim prometnim znakovima. Svaka klasa ima svoj jedinstveni naziv, primjerice "Ograničenje brzine (20 km/h)" ili "Zabrana preticanja".

```
def classify_image(image):
    image = image.resize((30, 30))
    image = np.array(image)
    image = np.expand_dims(image, axis=0)

    pred = model.predict(image)
    class_id = np.argmax(pred, axis=1)[0]

    return class_names[class_id]

file_upload = widgets.FileUpload(accept='image/*', multiple=False)

def on_upload_change(change):
    for filename, file_info in file_upload.value.items():
        image = Image.open(io.BytesIO(file_info['content']))
        u
        display(image)

        class_name = classify_image(image)
        print(f"Predikcija: {class_name}")

file_upload.observe(on_upload_change, names='value')

display(file_upload)
```

Slika 15 Kreiranje korisničkog sučelja

(vlastita izrada)

Funkcija classify_image uzima učitanu sliku, prilagođava je veličini koju model očekuje (30x30 piksela), te koristi model za predikciju klase prometnog znaka. Rezultat predikcije je naziv klase iz popisa class_names.

# 5. Praktična primjena razvijenog rješenja

Razvoj modela za prepoznavanje prometnih znakova predstavlja značajan korak prema poboljšanju sigurnosti i učinkovitosti cestovnog prometa. Model temeljen na konvolucijskoj neuronskoj mreži (CNN) pokazao se izuzetno učinkovitim u prepoznavanju različitih prometnih znakova, a njegove primjene u stvarnom svijetu su brojne i raznolike. U ovom eseju razmotrit ćemo nekoliko ključnih područja gdje se ovaj model može uspješno primijeniti, donoseći brojne prednosti i unapređenja.

Jedna od najvažnijih primjena modela za prepoznavanje prometnih znakova je integracija u napredne sustave za pomoć vozačima (ADAS). Ovi sustavi koriste razne senzore i kamere za praćenje okoline vozila te pružaju vozaču informacije i upozorenja u stvarnom vremenu. Prepoznavanje prometnih znakova može pomoći vozačima u održavanju propisane brzine, upozoravanju na nadolazeće opasnosti ili promjenu ograničenja brzine, čime se značajno povećava sigurnost na cestama. Integracija ovog modela u ADAS može smanjiti broj nesreća uzrokovanih nepažnjom ili nepoznavanjem prometnih znakova.

Autonomna vozila predstavljaju budućnost cestovnog prometa, a prepoznavanje prometnih znakova ključan je aspekt njihovog funkcioniranja. Autonomna vozila moraju biti sposobna prepoznati i reagirati na sve prometne znakove kako bi mogla sigurno i pravilno upravljati vozilom bez ljudske intervencije. Model za prepoznavanje prometnih znakova može se integrirati u sustav autonomnih vozila, omogućujući im da točno identificiraju znakove poput ograničenja brzine, zabrane skretanja ili obaveznih zaustavljanja, čime se osigurava sigurna i učinkovita vožnja.

U kontekstu pametnih gradova, prepoznavanje prometnih znakova može igrati značajnu ulogu u poboljšanju upravljanja prometom i infrastrukturom. Kamere postavljene na strateškim lokacijama mogu koristiti model za prepoznavanje prometnih znakova kako bi pratile i analizirale prometne uvjete u stvarnom vremenu. Ove informacije mogu se koristiti za optimizaciju prometnih tokova, smanjenje gužvi i poboljšanje sigurnosti na cestama. Također, prikupljeni podaci mogu pomoći u planiranju i održavanju prometne infrastrukture, identificirajući područja koja zahtijevaju intervenciju ili poboljšanja.

Moderni navigacijski sustavi mogu značajno koristiti prepoznavanje prometnih znakova kako bi pružili preciznije i sigurnije rute vozačima. Integracija modela za prepoznavanje prometnih znakova u navigacijske aplikacije omogućuje im da vozačima pružaju ažurirane informacije o prometnim ograničenjima i pravilima na njihovoj ruti. Na primjer, navigacijski sustav može upozoriti vozača na promjenu ograničenja brzine ili približavanje školskih zona,

čime se povećava svijest vozača o trenutnim prometnim uvjetima i smanjuje rizik od prekršaja ili nesreća.

Prepoznavanje prometnih znakova može značajno unaprijediti prometni nadzor i provedbu zakona. Policijske i druge nadzorne agencije mogu koristiti kamere opremljene modelima za prepoznavanje prometnih znakova kako bi automatski identificirale vozače koji krše prometne propise, poput prekoračenja brzine ili ignoriranja znakova stop. Ova tehnologija može pomoći u bržem i učinkovitijem provođenju zakona, smanjujući potrebu za ručnim nadzorom i omogućujući agencijama da se usredotoče na ozbiljnije prekršaje.

# 6. Zaključak

Rad se bavio razvojem i evaluacijom modela za prepoznavanje prometnih znakova koristeći konvolucijsku neuronsku mrežu (CNN). Podaci su prikupljeni iz skupa slika prometnih znakova, a model je implementiran korištenjem Pythona 3 i TensorFlow knjižnice. Implementacija modela zahtijevala je osnovno razumijevanje strojnog učenja i programiranja, a zajednica za podršku pružila je brojne resurse za pomoć.

Evaluacija modela provedena je pomoću mjere točnosti (accuracy) i vizualizacije performansi tijekom treniranja. Grafički prikazi točnosti i gubitka omogućili su uvid u proces učenja modela i identificirali potencijalne probleme poput prekomjernog prilagođavanja. Model je postigao visoku točnost u prepoznavanju prometnih znakova na testnom skupu podataka, što ukazuje na njegovu sposobnost generalizacije.

Rezultati istraživanja pokazuju da model može s visokom točnošću klasificirati prometne znakove, no važno je napomenuti da je moguće daljnje optimiziranje arhitekture mreže za još bolje performanse. Model pokazuje potencijal za primjenu u stvarnom okruženju, ali dodatna ispitivanja i proširenje skupa podataka su potrebni kako bi se osigurala robusnost i pouzdanost modela u različitim scenarijima.

Zaključno, razvijeni model za prepoznavanje prometnih znakova demonstrira značajan potencijal za unapređenje sigurnosti u cestovnom prometu, posebno kroz integraciju u sustave za pomoć vozačima i autonomna vozila. Daljnji razvoj i testiranje doprinijet će njegovoj učinkovitosti i pouzdanosti u stvarnim uvjetima.

# Popis literature


[1] I. El Naqa and M. J. Murphy, “What Is Machine Learning?,” in *Machine Learning in Radiation Oncology: Theory and Applications*, I. El Naqa, R. Li, and M. J. Murphy, Eds., Cham: Springer International Publishing, 2015, pp. 3–11. doi: 10.1007/978-3-319-18305-3_1.

[2] Y. Bengio, Y. Lecun, and G. Hinton, “Deep learning for AI,” *Commun ACM*, vol. 64, no. 7, pp. 58–65, Jun. 2021, doi: 10.1145/3448250.

[3] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, doi: 10.1038/nature14539.

[4] *Deep Learning | What is Deep Learning? | Deep Learning Tutorial For Beginners | 2023 | Simplilearn*, (2019). Accessed: Jul. 02, 2024. [Online Video]. Available: https://www.youtube.com/watch?v=6M5VXKLf4D4

[5] “Deep learning,” *Wikipedia*. Jun. 26, 2024. Accessed: Jul. 02, 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Deep_learning&oldid=1231081009

[6] X. Lei, H. Pan, and X. Huang, “A Dilated CNN Model for Image Classification,” *IEEE Access*, vol. 7, pp. 124087–124095, 2019, doi: 10.1109/ACCESS.2019.2927169.

[7] S. Saxena, “Introduction to The Architecture of Alexnet,” Analytics Vidhya. Accessed: Jul. 02, 2024. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/03/introduction-to-the-architecture-of-alexnet/

[8] V. Verdhan, “VGGNet and AlexNet Networks,” in *Computer Vision Using Deep Learning: Neural Network Architectures with Python and Keras*, V. Verdhan, Ed., Berkeley, CA: Apress, 2021, pp. 103–139. doi: 10.1007/978-1-4842-6616-8_4.

[9] “A Dilated CNN Model for Image Classification | IEEE Journals & Magazine | IEEE Xplore.” Accessed: Jun. 28, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8756165

[10] M. S. Islam, M. N. Islam, N. Hashim, M. Rashid, B. S. Bari, and F. A. Farid, “New Hybrid Deep Learning Approach Using BiGRU-BiLSTM and Multilayered Dilated CNN to Detect Arrhythmia,” *IEEE Access*, vol. 10, pp. 58081–58096, 2022, doi: 10.1109/ACCESS.2022.3178710.

[11] Y. Wang, Y. Zhao, L. Wu, Y. Zhang, and J. Hu, “Hybrid dilated convolutional neural network for solving electromagnetic inverse scattering problems,” *Int. J. RF Microw. Comput.-Aided Eng.*, vol. 32, no. 3, p. e23023, 2022, doi: 10.1002/mmce.23023.

[12] A. Hossain, “Classification of Image using Convolutional Neural Network (CNN).” May 2019.

[13] “Convolutional Neural Network (CNN) | TensorFlow Core,” TensorFlow. Accessed: Jun. 28, 2024. [Online]. Available: https://www.tensorflow.org/tutorials/images/cnn

[14] “GTSRB - German Traffic Sign Recognition Benchmark.” Accessed: Jul. 02, 2024. [Online]. Available: https://www.kaggle.com/datasets/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign

[15] “Python Machine Learning Train/Test.” Accessed: Jul. 02, 2024. [Online]. Available: https://www.w3schools.com/python/python_ml_train_test.asp

[16] *Build a Deep CNN Image Classifier with ANY Images*, (2022). Accessed: Jul. 02, 2024. [Online Video]. Available: https://www.youtube.com/watch?v=jztwpsIzEGc

[17] *Image Classification using CNN Keras | Full implementation*, (2021). Accessed: Jul. 02, 2024. [Online Video]. Available: https://www.youtube.com/watch?v=J1jhfAw5Uvo

[18] G. M. Foody, “Harshness in image classification accuracy assessment,” *Int. J. Remote Sens.*, vol. 29, no. 11, pp. 3137–3158, Jun. 2008, doi: 10.1080/01431160701442120.

[19] “ChatGPT.” Accessed: Jun. 28, 2024. [Online]. Available: https://chatgpt.com

# Popis slika