SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Raspoznavanje uzoraka i strojno učenje

Klasifikacija marki automobila na osnovu fotografije logotipa

Nikola Barbarić

Diplomski studij računarstva – modul DRD

Sadržaj

1. UVOD

Projektni zadatak obuhvaća predobradu i klasifikaciju fotografija logotipa marki vozila upotrebom strojnog učenja i korištenjem pretrenirane konvolucijske neuronske mreže nad setom podataka Car Logo dataset. [1] Korišten je programski jezik Python u razvojnom okruženju Anaconda. Cilj projektnog zadatka je istrenirati model nad dostupnim datasetom, te izvršiti njegovo testiranje sa fotografijama iz stvarnog života. Većina ljudi je upoznata sa logotipima automobila, međutim primjena je moguća i u mnogim drugim aspektima života, kao što su recimo upotreba u pametnim automobilima, kamerama na granicama, upotreba u policijske svrhe itd.

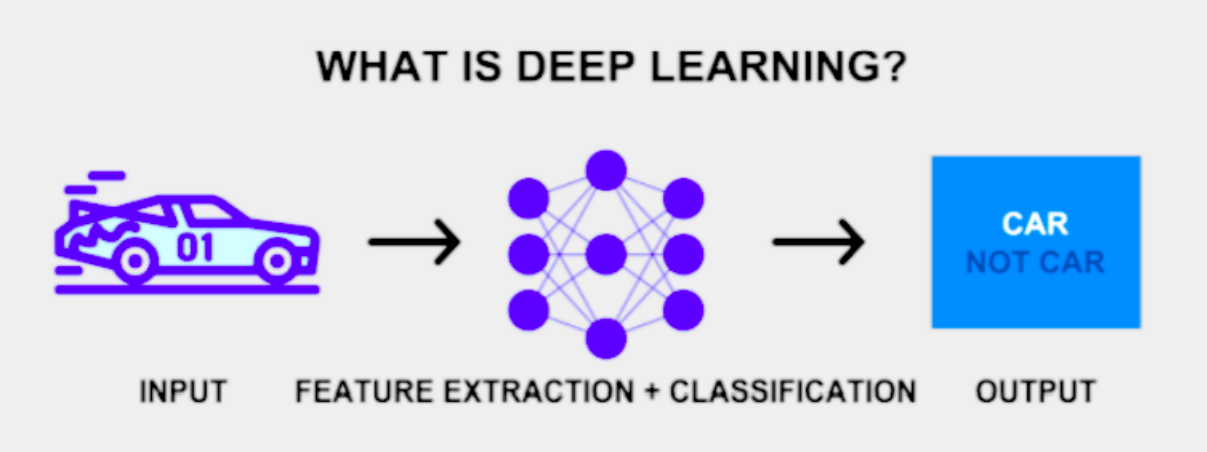
Car logo dataset sadrži 20 800 fotografija logotipa. Prikupljanje podataka temelji se na podacima flicr, google i yandex fotografija. Fotografije su podijeljene u 40 klasa; audi, bmw, mercedes, tesla... Za svaku klasu postoji oko 550 fotografija. Sve fotografije su dimenzija 50 x 50 piksela, te su razvrstane unutar 40 mapa. Ukupna veličina dataseta je oko 100 MB.

Upotreba pretreniranih modela spada u domenu tzv. „Transfer“ učenja. Keras je biblioteka neuronskih mreža otvorenog koda napisana u Python programskom jeziku. Keras se može izvoditi nad TensorFlow, Theano ili MCT-u (Microsoft Cognitive Toolkit). Dizajniran je kako bi omogućio brzo eksperimentiranje s dubokim neuronskim mrežama, jednostavan je za korištenje, modularan i proširiv. Sadrži brojne implementacije često korištenih blokova neuronskih mreža poput slojeva, aktivacijskih funkcija, optimizatora i mnoštvo drugih alata koji olakšavaju rad s slikovnim i tekstualnim podacima. Nadalje omogućava korisnicima distribuirano učenje modela dubokih neuronskih mreža na klasterima grafičkih procesora, te korištenje modela na iOS, Android i JVM (Java Virtual Machine) platformama.

U jednom od poglavlja ovog seminarskog rada, posebno je pokriveno testiranje, odnosno uspješnost ovog projektnog rada unutar lošijih uvjeta. Tu će biti vidljivi rezultati izvršeni nad 200-tinjak fotografija iz stvarnog života. Unutar tih 200-tinjak fotografija, bit će pokriveni slučajevi koji nisu baš idealni, poput lošeg osvjetljenja, krivog ugla fotografiranja itd.

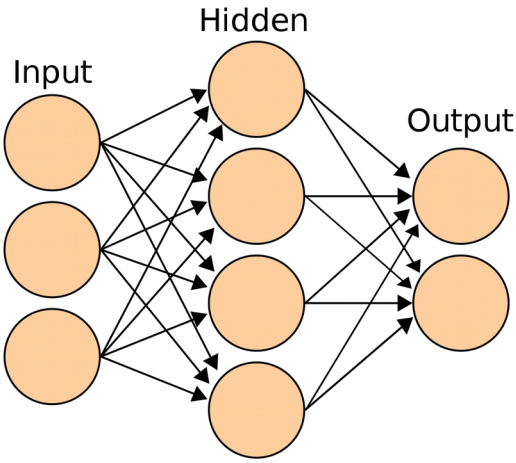
2. PREGLED PODRUČJA I PROBLEMATIKE

Duboko učenje je grana strojnog učenja posebno prikladna za rješavanje problema s područja umjetne inteligencije, poput razumijevanja govora, obrade fotografije i računalnog vida, obrade prirodnih i govornih jezika, i sl. Temelji se na predstavljanju podataka složenim reprezentacijama do kojih se dolazi slijedom naučenih nelinearnih transformacija. Pridjev „duboko“ u izrazu „duboko učenje“ označava slojevitost mreža koje rade na taj način, pri čemu ulazni podatci prolaze kroz svaki od slojeva i transformiraju se, nakon svakog sloja u složenije i apstraktnije reprezentacije. Za uspješnu primjenu dubokog učenja potreban je jako veliki skup podataka za učenje, te su u tu svrhu na internetu dostupni mnogi servisi koji pružaju različite vrste velikih skupova podataka, poput Kagglea, MNIST-a i drugih. Slika 2.1. vizualizira duboko učenje kao postupak kojim se za ulazne podatke izdvajaju značajke i obavlja klasifikacija, čime se na izlazu dobivaju korisni i strukturirani podatci.



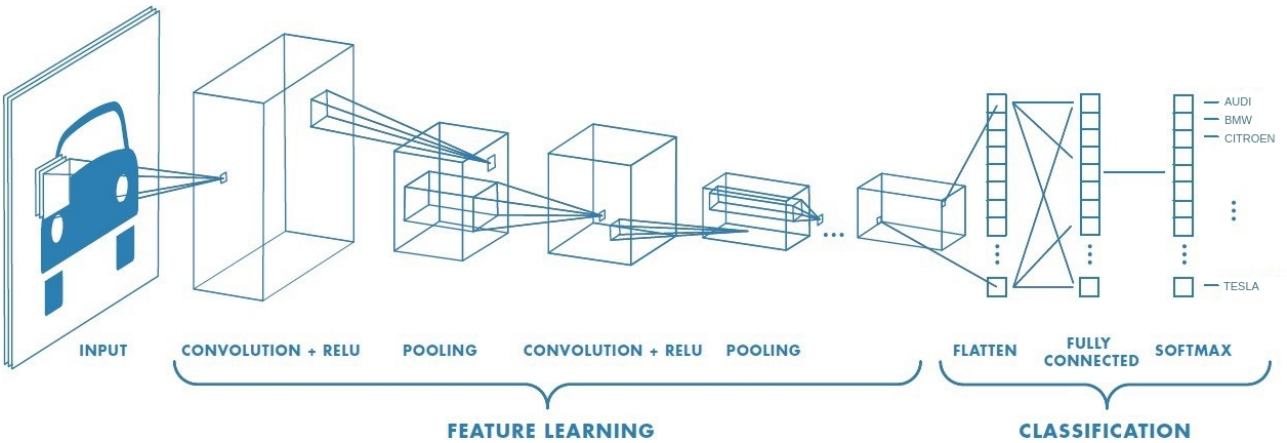
Sl. 2.1. Vizualizacija dubokog učenja

Umjetne neuronske mreže su računalni modeli obrade informacija čiji je koncept zasnovan na promatranju ljudskog mozga, u kojem biološki neuron prima neki ulazni signal preko dendrida, nakon čega se signal obrađuje u somi. U sljedećem koraku biološki neuron putem aksona pretvara obrađeni ulaz u izlaz i potom šalje informacije putem sinapsi do svih drugih neurona s kojima je promatrani neuron povezan. Analogno biološkom, umjetni neuron prima ulaze određene nekim težinskim koeficijentima, obrađuje ih, pretvara u izlaz pomoću prijenosne funkcije i u konačnici šalje informaciju prema izlazu i sljedećim neuronima. Svaka umjetna neuronska mreža sastoji se od triju slojeva: ulaznog, skrivenog i izlaznog. Pojednostavljeni prikaz dan je slikom 2.2



Slika 2.2. Pojednostavljeni prikaz umjetne neuronske mreže

Konvolucijske neuronske mreže trenutačno su najkorišteniji oblik dubokih neuronskih mreža u računalnom vidu. Sastoje se od niza slojeva kojeg obično čine konvolucijski sloj, sloj sažimanja i potpuno povezani sloj. Prvi sloj u konvolucijskoj mreži uvijek je konvolucijski sloj, koji koristi filter prolazeći kroz podatak s ulaza i spremajući zbroj konvolucija u mapu značajki, pri čemu se smanjuje broj parametara koji se trebaju naučiti. Nakon konvolucijskog sloja obično slijedi sloj sažimanja u kojem se, pomoću funkcije sažimanja, smanjuje količina ulaznih podataka. Pritom je funkcija sažimanja uglavnom u obliku maksimalnog ili prosječnog sažimanja. Potpuno povezani sloj najčešće je zadnji sloj konvolucijske neuronske mreže, a njegova je zadaća obaviti klasifikaciju temeljem značajki izvučenih iz prethodnih slojeva. Shematski prikaz arhitekture konvolucijske neuronske mreže nalazi se na slici 2.3.

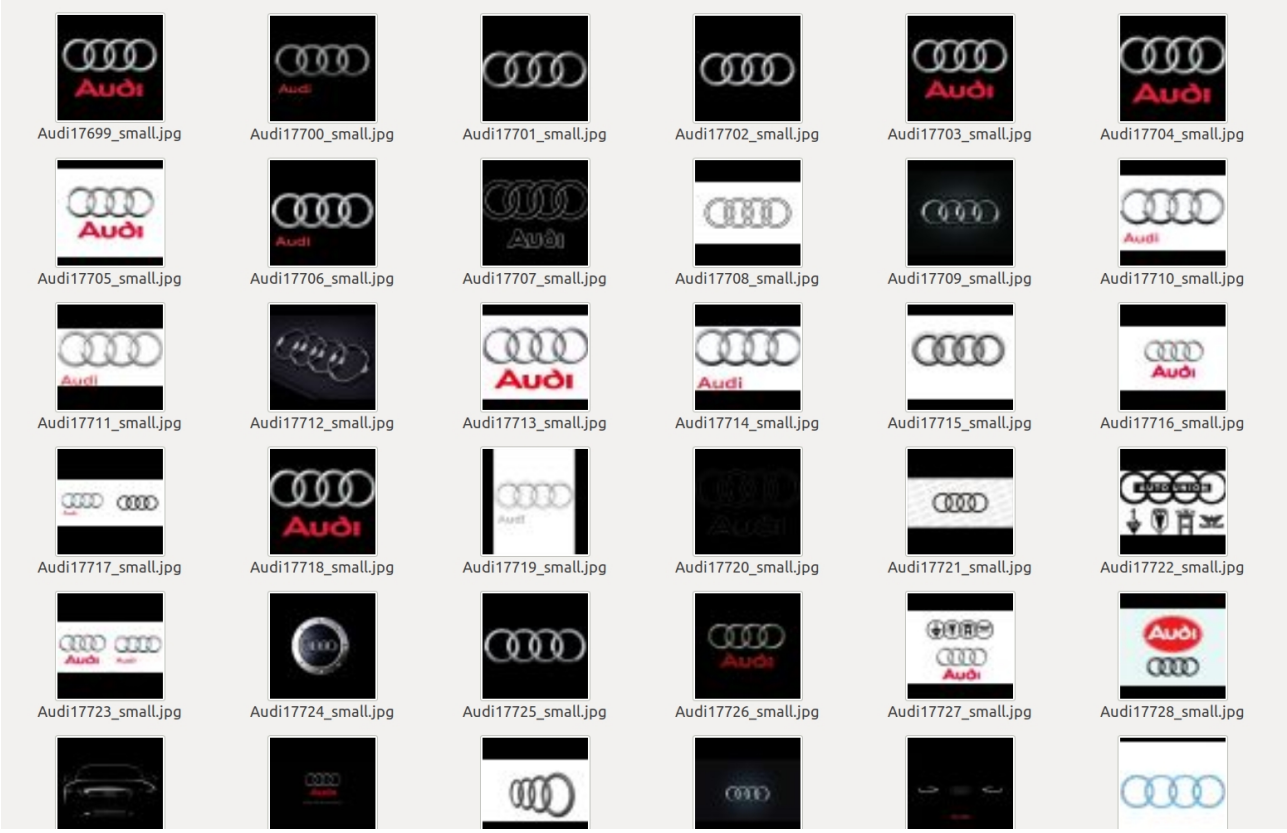


Slika 2.3. Shematski prikaz arhitekture konvolucijske neuronske mreže

3. OPIS ZADATKA

3.1. Podatkovni skup

Za potrebe realizacije zadatka, odnosno za treniranje, validaciju i testiranje modela, koristi se podatkovni skup Car Logo Dataset, koji je dostupan na Kaggleu. Sastoji se od ukupno 20.800 fotografija logotipa proizvođača automobila. Datoteke nisu odmah podijeljene u train i test mapu, nego se u nastavku dijele u omjeru 75% i 25%. Fotografije su raspoređene u 40 klasa, pri čemu svaka klasa predstavlja jednog proizvođača automobila, odnosno sadrži samo fotografije tog proizvođača. Dio sadržaja jedne od klasa prikazan je na sljedećoj slici.



Slika 3.1. Dio sadržaja klase Audi

3.2. Učenje mreže

U prethodnom poglavlju kao jedna od prednosti Pythona navedena je činjenica postojanja biblioteka koje sadrže vrlo jake alate pogodne za različite operacije. Kako bi se ti alati mogli koristiti u nekom programu, u njega je potrebno uključiti biblioteke koje sadrže tražene alate, a to se postiže navođenjem ključne riječi import iza koje slijedi modul kojeg se uključuje. Moguće je uključivanje više modula, a ovdje se koriste OS, NumPy, Pandas, Matplotlib, PIL, Tensorflow, Scikit-learn i

Keras. OS je biblioteka koja pruža funkcije za interakciju s operacijskim te datotečnim sustavom, što je važno kod rada s datotekama (npr. putanje) s obzirom da se neke značajke razlikuju od sustava do sustava. NumPy je jedan od osnovnih modula za Python, a namijenjen je prije svega radu s nizovima, vektorima i matricama, što je važno pri radu sa fotografijama koje nisu ništa drugo nego matrice. Pandas je open-source biblioteka koja omogućuje rad sa strukturiranim podatcima te alate za analizu podataka. Matplotlib je glavna Pythonova biblioteka za crtanje grafova. PIL je skraćenica od Python Imaging Library, a odnosi se na biblioteku s podrškom za otvaranje, rukovanje i spremanje različitih formata fotografija. Tensorflow je open-source biblioteka koju su razvili Googleovi stručnjaci za potrebe suočavanja s izazovima na području strojnog učenja i umjetne inteligencije, a koja se, osim u Pythonu, može koristiti i u C-u, Javi i Go-u. Scikit-learn je najkorisniji modul u Pythonu za područje strojnog učenja, a sadrži razne algoritme za klasifikaciju, regresiju i klasteriranje. Keras je biblioteka visoke razine namijenjena dizajniranju i treniranju dubokih neuronskih mreža, a može koristiti Tensorflow kao pozadinski kod. Uključivanje, odnosno učitavanje svih navedenih modula, ili barem njihovih manjih dijelova, u program, prikazano je slikom 3.4.

```
import matplotlib
matplotlib.use("Agg")
# import the necessary packages
from keras.preprocessing.image import ImageDataGenerator
from keras.optimizers import Adam
from sklearn.model_selection import train_test_split
from keras.preprocessing.image import img_to_array
from keras.utils import to_categorical
from lenet import LeNet
from imutils import paths
import matplotlib.pyplot as plt
import numpy as np
import argparse
import random
from cv2 import os
import cv2
```

Slika 3.2. Učitavanje modula u program

Za izradu aplikacije za prepoznavanje logotipa automobila na slici potrebno je prvo izraditi model koji raspoznaje logotipe. Koristeći Keras i Python napravljen je model konvolucijskom neuronskom mrežom. Fotografije se sastoje od velikog broja piksela i 3 kanal boja. Iako su slike male i dalje predstavljaju problem velike vremenske i prostorne složenosti. Upravo je to razlog upotrebe konvolucijske neuronske mreže. Što više fotografija za učenje model ima, to će njegovi rezultati biti točniji.

Python skripta koja je napravljena za treniranje modela kao argumente prima putanju do mape sa slikama dataseta, putanju do mjesta gdje želimo spremiti model, te putanju do lokacije na koju se želi spremiti graf koji pokazuje točnost modela. Navedeno je vidljivo na sljedećoj slici.

```
# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-d", "--dataset", required=True,
        help="path to input dataset")
ap.add_argument("-m", "--model", required=True,
        help="path to output model")
ap.add_argument("-p", "--plot", type=str, default="plot.png",
        help="path to output accuracy/loss plot")
args = vars(ap.parse_args())
```

Slika 3.3. Argumenti skripte (train_network.py)

Prije nego je model istreniran potrebno je odrediti broj epoha (engl. Epoch) i batch size. Batch_size je parametar koji nam definira broj uzoraka koji će propagirati kroz mrežu. Prednosti manjeg batcha su manji zahtjevi za memorijom i veća brzina učenja. Međutim ukoliko je batch_size malen, veća je pogreška preciznosti procjene gradijenta. Koristi se batch_size 32. Broj epoha je broj prolazaka kroz čitav dataset za treniranje, te može biti između 1 i beskonačno, dok batch_size mora biti između 1 i broja uzoraka za testiranje. Treniranje se može zaustaviti iako nisu sve epohe odrađene, iako sa svakom novom epohom se dobija na točnosti i pouzdanosti modela.

Broj epoha i vrijednost batch_size nisu određene, nego se testiraju, te se njihov broj povećava dok se ne zadovolji određena točnost. U ovom primjeru za broj epoha je odabaran broj 25. U nastavku skripte, sve se fotografije učitavaju, te se miješaju nasumično. Radi lakše obrade svaka slika je snižena na veličinu od 28x28 piksela. Svaka fotografija se sprema u jedno polje, dok se naziv njenog direktorija sprema u polje, gdje svaki element ima vrijedsosti između 0 i 39, ovisno koji je logotip u pitanju. Sljedeći korak obuhvaća smanjivanje piksela slika iz raspona 0-255 na raspon 0-1, te se podatci dijele na dva dijela. Jedan za treniranje i drugi za testiranje. Omjer između njih je 75-25%. Za treniranje je korištena konvolucijska mreža pod nazivom LeNet koja za parametre ima veličinu slike, te broj klasa. Nadalje, model se trenira, te se sprema na računalo.

```
# scale the raw pixel intensities to the range [0, 1]
data = np.array(data, dtype="float") / 255.0
labels = np.array(labels)
# partition the data into training and testing splits using 75% of
# the data for training and the remaining 25% for testing
(trainX, testX, trainY, testY) = train_test_split(data,
        labels, test_size=0.25, random_state=42)
# convert the labels from integers to vectors
trainY = to_categorical(trainY, num_classes=40)
testY = to_categorical(testY, num_classes=40)

# construct the image generator for data augmentation
aug = ImageDataGenerator(rotation_range=30, width_shift_range=0.1,
        height_shift_range=0.1, shear_range=0.2, zoom_range=0.2,
        horizontal_flip=True, fill_mode="nearest")

# initialize the model
print("[INFO] compiling model...")
model = LeNet.build(width=28, height=28, depth=3, classes=40)
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model.compile(loss="categorical_crossentropy", optimizer=opt,
        metrics=["accuracy"])
# train the network
print("[INFO] training network...")
H = model.fit_generator(aug.flow(trainX, trainY, batch_size=BS),
        validation_data=(testX, testY), steps_per_epoch=len(trainX) // BS,
        epochs=EPOCHS, verbose=1)
# save the model to disk
print("[INFO] serializing network...")
model.save(args["model"])
```

Slika 3.4. Kod train_network.py

Nakon što je model spremljen, potrebno ga je testirati sa podatcima koje do sada nije vidio. Ovisno o projektu, rezultati koje model daje u ovom koraku određuju da li je spreman za korištenje.

```
# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-m", "--model", required=True,
        help="path to trained model model")
ap.add_argument("-i", "--image", required=True,
        help="path to input image")
args = vars(ap.parse_args())
```

Slika 3.5. Dio koda test_network.py

Na slici iznad je vidljiv dio koda koji pokazuje što se predaje skripti kao argument, a to su putanja do modela, te putanja do slike za koju se vrši testiranje. Pokretanjem skripte, slika se učitava te se obrađuje da bi ju model mogao pročitati. Smanjuje se na veličinu 28x28 piksela, te se svi pikseli dovode u raspon 0-1. Sljedeći korak je učitavanje modela koji određuje kojoj klasi slika pripada. Za svaku klasu model vraća broj u rasponu između 0-1 koji govori koliko je siguran da je riječ upravo

o toj klasi. Što je taj broj veći to je model sigurniji. Cilj je postići što veću točnost modela, međutim moguće su razne smetnje na slici koja se testira, od lošeg osvjetljenja, preko nepravilne veličine, do pogrešnog kuta slikanja. Sve navedeno utječe na rezultat. Za izradu mobilne verzije aplikacije, korištena je skripta converter.py, koja pretvara računalni model u lite model za mobilne uređaje.

```
# load the image
image = cv2.imread(args["image"])
orig = image.copy()
# pre-process the image for classification
image = cv2.resize(image, (28, 28))
image = image.astype("float") / 255.0
image = img_to_array(image)
image = np.expand_dims(image, axis=0)
# load the trained convolutional neural network
print("[INFO] loading network...")
model = load_model(args["model"])
# classify the input image
(alfa, audi, bmw, chevrolet, citroen, dacia, daewoo, dodge, ferrari, fiat, ford, honda,
hyundai, jaguar, jeep, kia, lada, lancia, landrover, lexus, masserati, mazda, mercedes,
mitshubisi, nissan, opel, pegueot, porsche, renault, rover, saab, seat, skoda, subaru, suzuki,
tata, tesla, toyota, volkswagen, volvo) = model.predict(image)[0]
```

Slika 3.6. Dio koda test_network.py

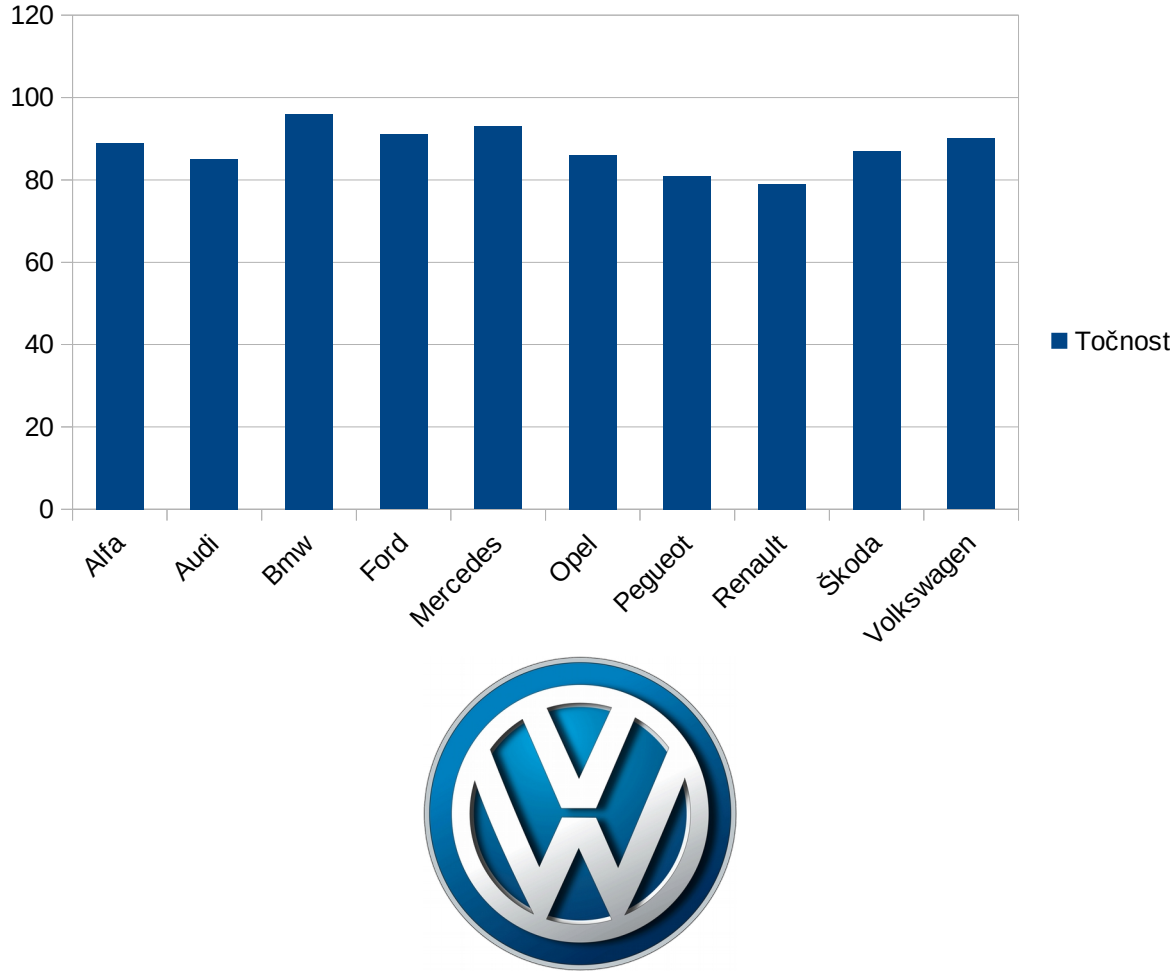
4. TESTIRANJE FOTOGRAFIJA IZ STVARNOG ŽIVOTA

Budući da je model treniran nad datasetom unutar kojega je većina fotografija logotipa računalno napravljena, u ovom dijelu seminara bit će prikazani rezultati primjene modela nad fotografijama iz stvarnog života. Budući da se dataset sastoji od 40 različitih proizvođača te njihovih logotipa, od kojih neke je rijetkost sresti na hrvatskim cestama, prilikom testiranja bit će uzeto 10 najpoznatijih brandova. Zbog toga što je modelu najbitniji dio logotip, ostatak auta će biti izrezan sa fotografija, osim u jednom slučaju kada će se provjeriti kolika je točnost u tom slučaju. Neki od primjera kada se ne očekuje velika točnost modela su: slabije osvjetljenje, loš ugao snimanja...

Testiranje je vršeno na sljedeći način. Po parkinzima je pronađeno po 5 automobila za svakog proizvođača koji se testira, te za svakog proizvođača napravljene po 4 fotografije od kojih je jedna onakva kakvu model očekuje, jedna sa lošim osvjetljenjem, jedna fotografija čitavog automobila i jedna uslikana pod pogrešnim uglom. Znači za svakog proizvođača koji je testiran je fotografirano po 20 fotografija, ukupno 200 njih, nad kojima je model testiran. Rezultati su dani u nastavku, a za poboljšanje rezultata, bilo bi korisno uvesti povratne informacije od korisnika, te da se fotografije koje su pogrešne spremaju i iskoriste kao dodatak datasetu za treniranje mreže.

Analizom rezultata je utvrđeno da BMW daje najbolje rezultate, zbog specifičnosti svog logotipa koji nema praznina poput ostalih, ali utjecaj igraju i fotografije u datasetu kojih ima malo više u odnosu na ostale proizvođače, te su kvalitetnije.

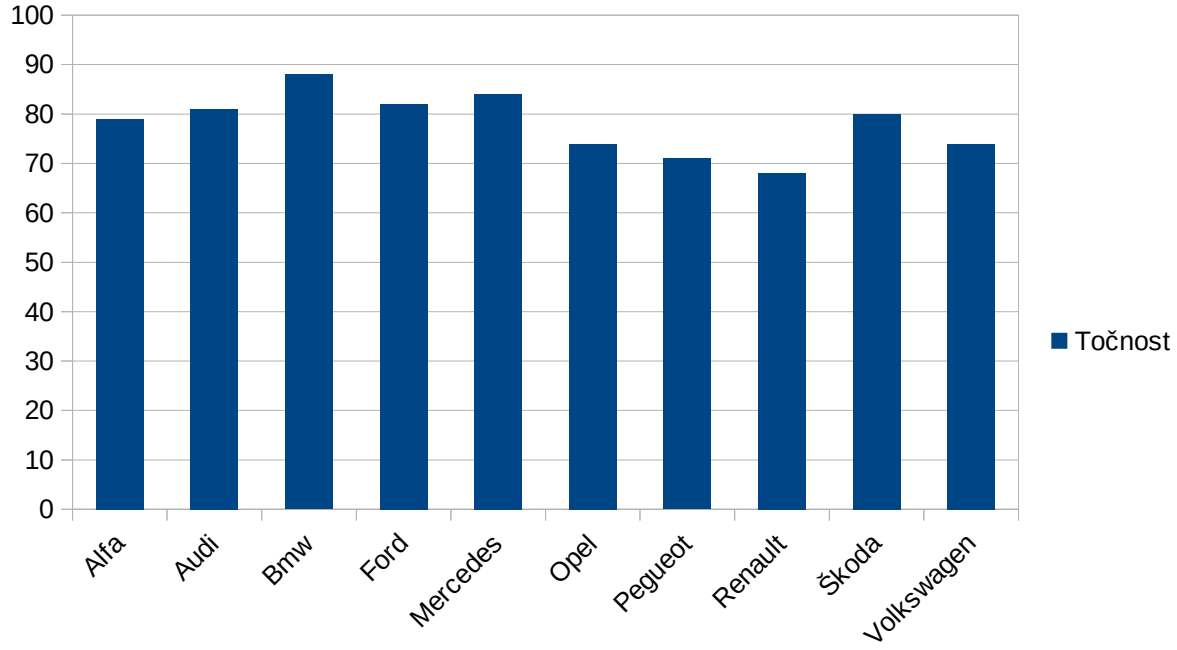
4.1. Idealne fotografije



Slika 4.1.1. Primjer idealne slike

U ovom testu su korištene idealne slike preuzete sa interneta, te je vidljivo da je točnost modela iznad 79%, što je već dovoljno, iako je prosječna točnost 87,7 %. Ovaj slučaj je idealan, ali nije baš najmjerodavniji jer nikad u stvarnom životu nije moguće dobiti ovako lijepe slike.

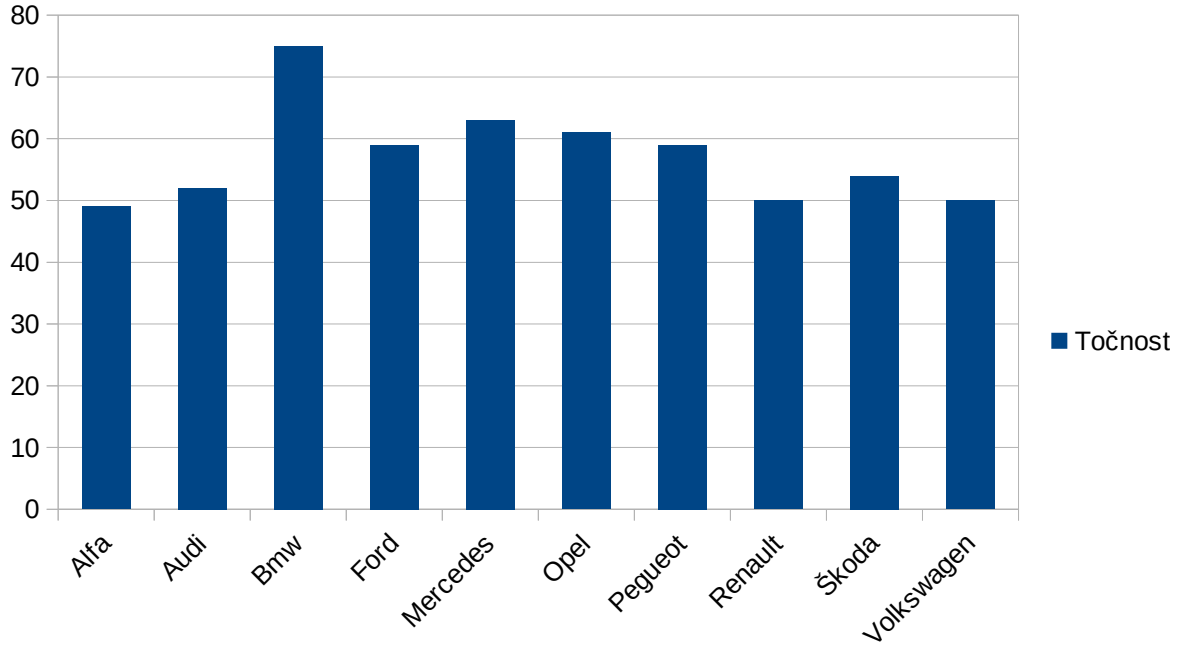
4.2. Fotografije sa lošijim uglom



U ovom testu su korištene fotografije koje su fotografirane pod pogrešnim uglom, ali je izrezan samo logo sa fotografija, te je model i dalje prilično točan. Model ipak daje prilično dobre rezultate, jer u datasetu ima i određeni broj fotografija koje su slikane pod različitim uglovima. Najmanju točnost ovdje ima Renault sa točnosti od 68%, dok je prosječna točnost i dalje iznad toga, pa model još uvijek možemo smatrati korisnim. U ovom dijelu se računalo i čovjek razlikuju, jer čovjek puno lakše prepoznaje i pod lošijim uvjetima.

Slika 4.2.1. Primjer fotografije pod lošim uglom

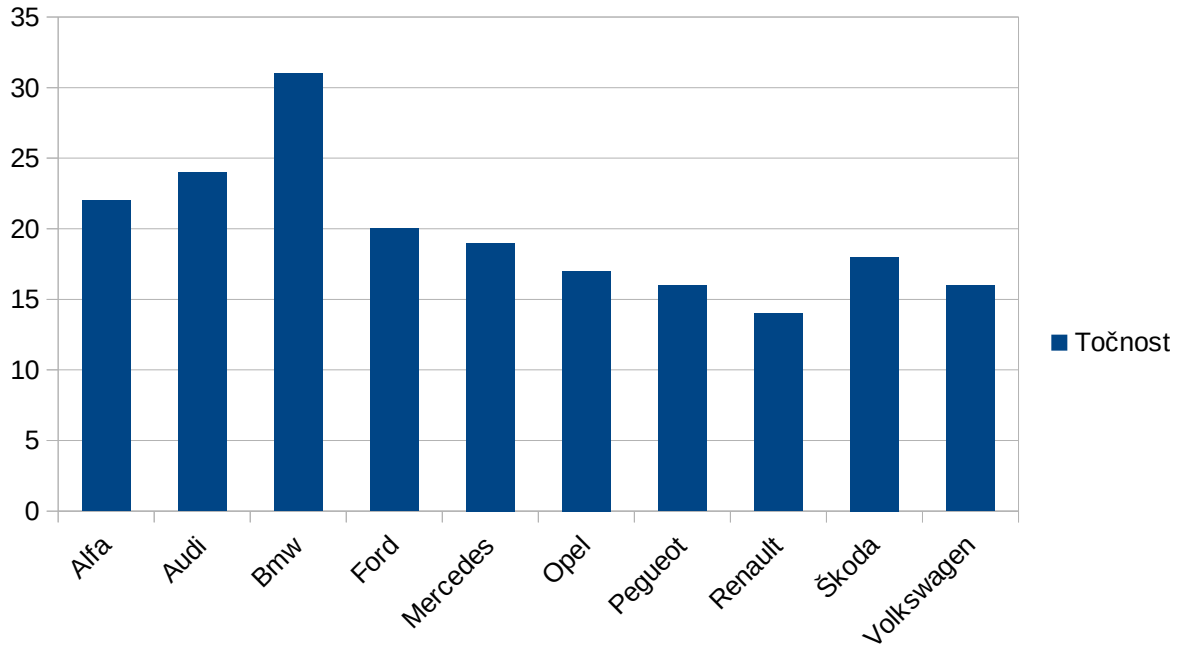
4.3. Fotografije napravljene u mračnijim uvjetima



Unutar ovog testa korištene su fotografije napravljene u mračnijim uvjetima, te se vidi pad u točnosti modela, a glavni razlog je taj što većina fotografija u datasetu je napravljena pri odličnom osvjetljenju. Iako ima točnost uglavnom preko 50%, ovaj test pokazuje da je bitno prilikom fotografiranja omogućiti osvjetljenje, jer se gubi 20% točnosti bez osvjetljenja.

Slika 4.3.1. Primjer fotografije pod mračnijim osvjetljenjem

4.4. Korištenje fotografija na kojima nije izrezan logo



U ovom testu su korištene fotografije koje nisu izrezane, te je vidljivo da je model loš za takvo testiranje, jer kao prvo u datasetu nema takvih fotografija, a i cijelu fotografiju promatra kao logotip, jer je to ono što on očekuje. Vrijednosti koje se dobivaju ovdje već nisu pouzdane za korištenje modela. Da bi se koristile ovakve fotografije, potreban bi bio drugačiji dataset. Rješenje za korisničku aplikaciju bi moglo biti i predkorak u kojemu bi korisnik morao označiti logo na slici.

Slika 4.4.1. Primjer fotografije cijelog automobila

5. ZAKLJUČAK

Koristeći programski jezik Python i biblioteku Keras u ovom projektnom zadatku napravljen je model koji omogućuje klasifikaciju proizvođača automobila na osnovu logotipa. Model je naučen da prepoznaje logotipe koristeći konvolucijsku neuronsku mrežu. Model je istreniran da prepoznaje 40 različitih proizvođača. Korišten je dataset sa 20800 fotografija, koje su razvrstane po mapama, odnosno proizvođačima. Od toga broja 75% je odvojeno za treniranje mreže, a 25% za testiranje. Nakon što je mreža istrenirana i testirana, sa prosječnom točnosti od 94%, izvršeno je pretvaranje u laganije model .tflite, koji je ubačen u android aplikaciju da bi se omogućila brza upotreba. Iako je model pokazao točnost od 94% prilikom testiranja nad odvojenim podatcima, izvršeno je testiranje modela u stvarnom životu nad 10 poznatijih proizvođača automobila koji se mogu sresti na hrvatskim prometnicama. Tu je točnost bila 87,7% što je i dalje vrlo dobro i zadovoljavajuće.Nadalje je model testiran sa malo lošijim slikama, gdje i ako je slika pod lošijim kutem daje prihvatljivu točnost, međutim ako se radi o slikama sa lošijim osvjetljenjem model je na granici točnosti. Za fotografije čitavih automobila model je imao lošu točnost i nije upotrebljiv. Sam projektni zadatak je odličan način za upoznavanje sa strojnim učenjem, te je projekt zanimljiv za početnike. Prostor za napredak se ostavlja u omogućavanju proširivanja dataseta na osnovu korisničke povratne informacije, te ponovnog treniranja mreže. U radu je korištena malo modificirana verzija LeNet mreže, programski jezik Python, Keras, TensorFlow, te ostale biblioteke za rad sa slikama.

6. LITERATURA

1. https://www.kaggle.com/ritikgoyal1710/54698752
2. https://keras.io/why_keras/
3. https://towardsdatascience.com/machine-learning-classifiers-a5cc4e1b0623