Univerzitet u Sarajevu

Elektrotehnički fakultet
Odsjek za računarstvo i informatiku

Projekat

Klasifikacija motornih vozila iz zračnih fotografija

Begović Amar, Bundavica Aldina, Omerbegović Armin
Predmet: Prepoznavanje oblika i obrada slike
Nastavnik: Van.prof.dr Samir Omanović, dipl.ing.el
Akademska godina: 2018/2019

Sarajevo, novembar 2018.

Sadržaj

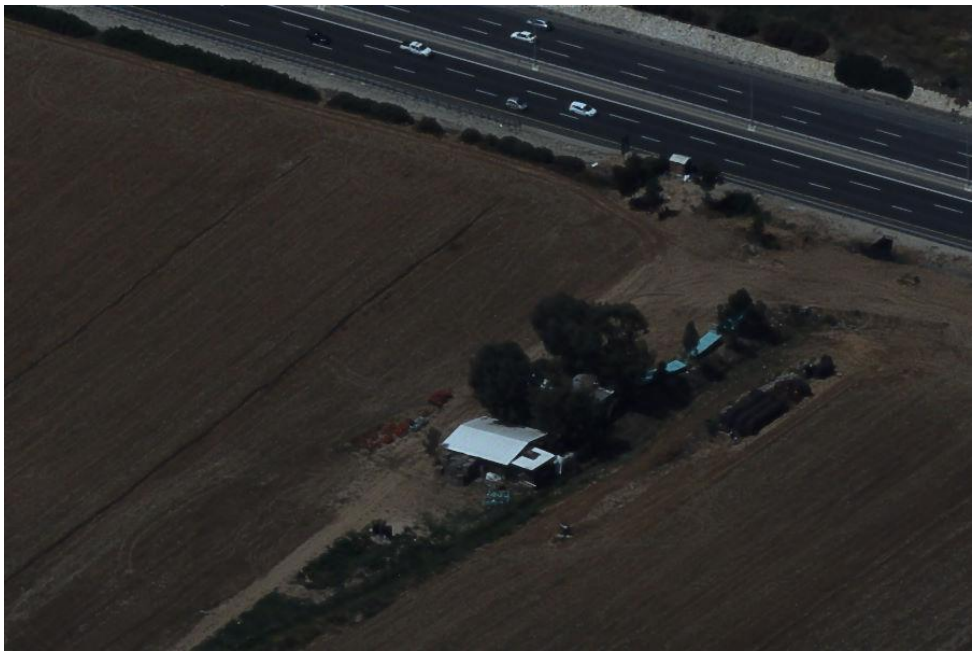
1 Projektni zadatak I

1.1 Skup podataka

Klasifikacija objekata predstavlja važan aspekt u obradi slike. Detekcija objekata se može tretirati kao klasifikacija ukoliko jednu klasu predstavimo kao region od interesa, a drugu klasu ostatak slike. Sljedeće slike su reprezentativan primjer našeg skupa podataka, nad kojim će se vršiti daljnja obrada.

Slika 1.1: Slika iz skupa podataka - 10082

Slika 1.2: Slika iz skupa podataka - 10173



Slika 1.3: Slika iz skupa podataka - 10182

U konkretnom projektu vršit će se klasifikacija motornih vozila na velika i mala. Mala vozila podrazumijevaju automobile, kombi vozila, kamp kućice, dok velika podrazumijevaju kamione, autobuse, te vozila posebne namjene, kao što su građevinska vozila i vozila za obrađivanje poljoprivrednih dobara.

Skup podataka je podijeljen tako da je broj slika u svakom:

- Skup podataka za treniranje: 1330
- Skup podataka za testiranje: 167
- Skup podataka za validaciju: 166

Ukupni broj malih motornih vozila po skupovima:

- Skup podataka za treniranje: 6406
- Skup podataka za testiranje: 903
- Skup podataka za validaciju: 860

Pošto je skup podataka već preuzet u podijeljenom obliku na skup za treniranje i testiranje, a skup za validaciju je kreiran iz skupa za testiranje, nije bilo moguće da broj svih klasa bude jednak u skupu za validaciju. Također, nije prirodno da broj velikih vozila bude jednak broju malih vozila, jer i u stvarnom svijetu nije taj broj jednak. Tako da je skup za validaciju kreiran na bazi slučajnog odabira i skripta koja to omogućava je data ispod.

klasa	train	test	val
hatchback	1898	243	111
jeep	543	68	50
minivan	330	65	9
pickup	203	28	16
sedan	3291	476	169
van	141	23	12

Tablica 1: Ukupni broj malih motornih vozila po skupovima

```
import split_folders


src_dir = 'dataset/input'
dst_dir = 'dataset/output'

split_folders.ratio(src_dir, dst_dir, seed=1337, ratio=(.835, .165))
```

1.2 Pretprocesiranje podataka

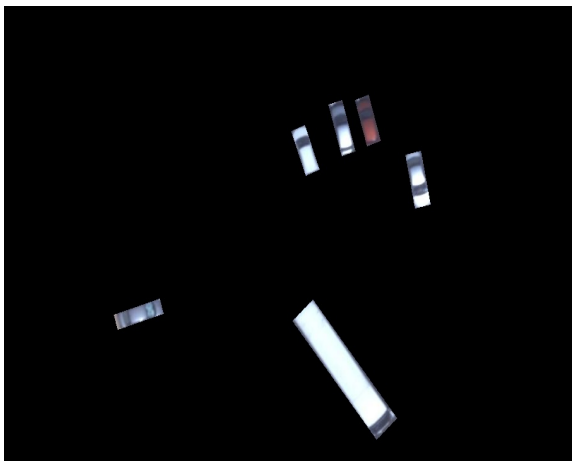
1.2.1 Regioni od interesa

Regioni od interesa predstavljaju dijelove slika koje sadrže informacije bitne za dalje procesiranje, odnosno predstavljaju dio slike koji se želi detektovati i/ili klasificirati.

Kreiranje maski uklanja sve dijelove slike (boji ih u crno) osim potrebnih objekata na slici, odnosno vozila. Funkcija radi tako što joj se proslijedi slika i naziv te slike, nakon čega se dobavljaju koordinate vozila iz CSV file-a i maskiraju se nepotrebni dijelovi.

```
import numpy as np
import cv2
import pandas as pd
import glob
import os


def create_mask(img, image_name, csv):
    image_id_new = int(image_name)
    selection = csv[csv.image_id == image_id_new]
    objects_vehicles = []
    for idx, row in selection.iterrows():
        point1 = [row.p1_x, row.p1_y]
        point2 = [row.p2_x, row.p2_y]
        point3 = [row.p3_x, row.p3_y]
        point4 = [row.p4_x, row.p4_y]
        objects_points = [[point1, point2, point3, point4]]
        objects_vehicles += objects_points
    mask = np.zeros(img.shape)
    for x in objects_vehicles:
        vrx = np.array(x, np.int32)
        vrx = vrx.reshape((-1, 1, 2))
        mask = cv2.fillPoly(mask, [vrx], (255, 255, 255))
    masked = np.where(mask > 1, img, np.zeros(img.shape))
    return masked


print("Working on it, please have patience")
src_dir = 'dataset/training_imagery'
dst_dir = 'dataset/masked_training/'
df = pd.read_csv('dataset/train.csv', sep=',')
for filename in glob.glob(os.path.join(src_dir, '*')):
    im = cv2.imread(filename)
    name = filename.replace(src_dir, '')
    img_name = name.replace('.jpg', '')
    img_name = img_name.replace('\\', '')
    masked_img = create_mask(im, img_name, df)
    cv2.imwrite(dst_dir + name, masked_img)
print("Done")
```

Slika 1.4: Primjer 1 - Originalna slika[1]



Slika 1.5: Primjer 1 - Rezultat maskiranja

1.2.2 Uklanjanje šuma

Uklanjanje šuma predstavlja jedan od najbitnijih koraka u pretprocesiranju slike. Šum ne sadrži nikakvu korisnu informaciju na osnovu koje bi regione od interesa izdvajali od ostatka slike ili na osnovu koje bi vršili klasifikaciju na samom regionu od interesa. Zbog takvih svojstava šuma teži se njegovom uklanjaju.

Nelokalno usrednjavanje Algoritam za nelokalno usrednjavanje (eng.*Non-local means*) je korišten iz razloga što ulazne slike sadrže dosta različitih objekata, odnosno veliki broj informacija. Ukoliko bi bio primijenjen neki lokalni filter sa uklanjanje šuma, postoji mogućnost da bi veliki broj informacija bio izgubljen što u pretprocesiranju slike nije od interesa. Non-local means algoritam koristi srednju vrijednost svih piksela na slici pojačanu tako da ima približnu vrijednost pikselu koji se trenutno obrađuje. Na taj način obezbijeđeno je očuvanje bitnih informacija, dok je šum u velikoj dozi uklonjen. Rezultat algoritma je prikazan na slici 1.7.

Slika 1.6: Primjer 1 - Originalna slika

Slika 1.7: Primjer 1 - Rezultat nelokalnog usrednjavanja

[1]Rezolucija slike: 900×600

Na slikama se vidi da je uklanjanje šuma donijelo očekivane rezultate gdje su sitni detalji koji ne sadrže bitne informacije uklonjeni i sam skup slika je pogodniji za dalju obradu.

Maskiranje neoštrina Maskiranje neoštrina se odražava u smanjenju oštrih prijelaza na slikama. Posljedica takvog pristupa je zamagljivanje (eng. *blurring*), što može predstavljati problem u zavisnosti od informacija koje sadrže ulazne slike. Na konkretnom primjeru ulaznih slika, maskiranje neoštrina ne predstavlja koristan korak u pretprocesiranju slika, jer ja krajnji cilj detekcija motornih vozila. Na slikama 1.9 i 1.11 su prikazani rezultati maskiranja neoštrina koje je izvršeno pomoću dodavanja Gausovog šuma (eng. *Gaussian blur*).

Slika 1.8: Primjer 1 - Originalna slika

Slika 1.9: Primjer 1 - Rezultat maskiranja neoštrina

Slika 1.10: Primjer 1 - Rezultat nelokalnog usrednjavanja

Slika 1.11: Primjer 1 - Rezultat maskiranja neoštrina

Na osnovu dobijenih rezultata zaključak je da maskiranje neoštrina nema primjenu na ulaznom skupu slika, kao ni na skupu slika sa uklonjenim šumom.

U implementaciji je korištena brza verzija algoritma Non-local means dostupna u Python biblioteci OpenCV.

```
import cv2
import glob
import os


def denoising(img):
    denoised_img = cv2.fastNlMeansDenoisingColored(img, None, 5, 5, 7,
       21)
    return denoised_img


def unsharp_masking(src):
    gaussian = cv2.GaussianBlur(src, (9, 9), 10.0)
    unsharp_img = cv2.addWeighted(src, 1.5, gaussian, -0.5, 0, src)
    return unsharp_img


print("Working on it, please have patience")
src_dir = 'dataset/training_imagery'
dst_dir = 'dataset/denoised_training'
dst_dir2 = 'dataset/unsharped_training'
unsharp_original_flag = False
for filename in glob.glob(os.path.join(src_dir, '*')):
    im = cv2.imread(filename)
    name = filename.replace(src_dir, '')
    denoised = denoising(im)
    if unsharp_original_flag:
        unsharped = unsharp_masking(im)
    else:
        unsharped = unsharp_masking(denoised)
    cv2.imwrite(dst_dir + name, denoised)
    cv2.imwrite(dst_dir2 + name, unsharped)
print("Done")
```

1.2.3 Poboljšavanje kvaliteta slika

U ovom dijelu zadatka trebala su se implementirati najmanje tri filtera koja pospješuju kontrast, osvjetljenje i ujednačavaju histogram. Testiranjem mnogih algoritama nad slikama, odabrana su tri najbolja za naš dataset. U nastavku će bit opisani algoritmi koji su u najvećoj mjeri pomogli poboljšanju neke od osobina.

Poboljšavanje osvijetljenja Metode koje se koriste pri poboljšanju osvijetljenja usko su vezane za poboljšanje kontrasta. Za naš dataset odabrana je gamma korekcija. Za gamma manje od 1 histogram slike se pomjera udesno, pa je nastala slika svjetlija od originalne. Metodom pokušaja i pogrešaka, birane su različite vrijednosti korekcionog faktora gamma i odabrana je vrijednost 0.5. Korištena funkcija radi po formuli

$$I = O^{\gamma}$$

nakon što svaki piksel skalira na neku vrijednost od 0 do 1. Sljedeća slika prikazuje poboljšanje osvjetljenosti obrađivane slike.

Slika 1.12: Prikaz originalne slike (lijevo) i slike s poboljšanom osvjetljenošću (desno)

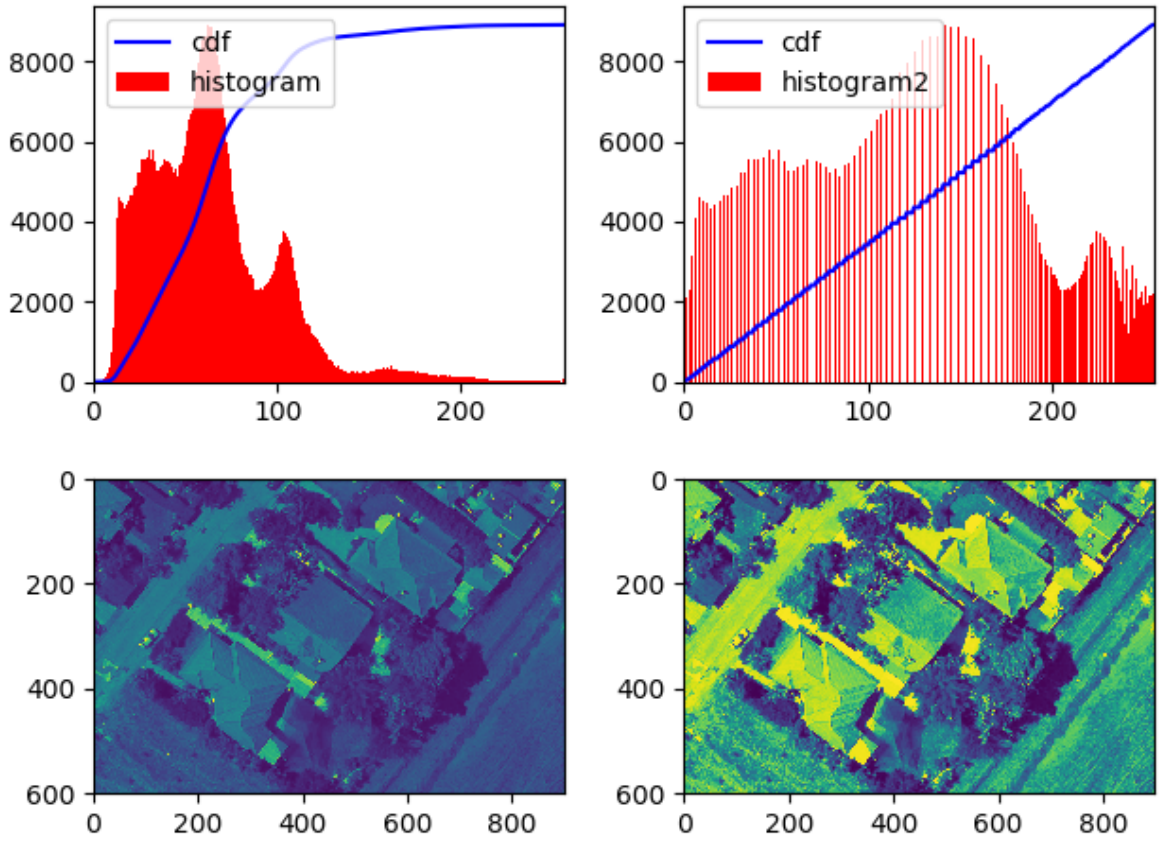
Poboljšavanje kontrasta Kao i za traženje prikladnog algoritma za poboljšanje osvjetljenosti, tako su i za poboljšanje kontrasta, testirane različite metode. Od testiranih metoda, odabrana je sigmoidna funkcija. Ova funkcija radi po formuli

$$O = 1/(1 + e^{gain*(cutoff-I)})$$

Ova funkcija je sadržana u biblioteci scikit-image i prima četiri parametra: sliku, cut off vrijednost, gain i bool varijablu. Cut off vrijednost pomjera karakteristiku udesno, a gain predstavlja konstantu, kojom se izraz množi, dok bool varijabla označava vraća li se inverzna sigmoid funkcija ili ne. Sljedeća slika prikazuje sliku s poboljšanim kontrastom nad kojom je primjenjena sigmoid funkcija s parametrima: 0.5, 10, false.

Slika 1.13: Prikaz originalne slike (lijevo) i slike nad kojom je primjenjena sigmoidna funkcija (desno)

Ujednačavanje histograma Različite nivoe histograma je moguće posmatrati kao slučajne varijable u rasponu [0, L-1]. Ujednačavanje histograma ima za cilj transformirati sliku tako da funkcija raspodjele vjerojatnoće bude uniformnog oblika, odnosno postizanje jednake zastupljenosti svih vrijednosti intenziteta. Ova transformacija rezultira dobrim kontrastom na slici.



Slika 1.14: Prikaz histograma originalne slike (lijevo) i prikaz primjenjenog globalnog histograma na sliku (desno)

Prethodni slučaj obuhvaća ravnomjerno izjednačavanje, što uvijek nije povoljno, pa se pribjegava lokalnom izjednačavanju histograma. Contrast Limited Adaptive Histogram Equalization (CLAHE) je varijanta adaptivnog ujednačavanja histograma. U ovoj metodi, kontrastno pojačanje je ograničeno kako bi se smanjio problem šuma. Kontrastno pojačanje je opisano nagibom funkcije, koji je proporcionalan nagibu kumulativne funkcije raspodjele susjedstva. Ovime je ograničeno rezanje histograma na neku unaprijed definiranu vrijednost, prije nego što se izračuna funkcija raspodjele, pa ograničenje histograma ovisi o normalizaciji histograma. Vrijednosti koje premašuju ograničenje se ne odbacuju već redistribuiraju pri dnu histograma. Korištena biblioteka OpenCV posjeduje i gotovu metodu za CLAHE algoritam. Slika je podijeljena na 8x8 dijelova (defaultna vrijednost). Svaki od tih dijelova se ujednačava. Sljedeća slika prikazuje originalnu sliku, te sliku kada se na nju primjeni CLAHE i obično ujednačavanje histograma. Ovdje se jasno vidi prednost CLAHE u odnosu na globalno ujednačavanje histograma.

U nastavku rada koristit će se CLAHE algoritam zbog boljeg ujednačavanja histograma.

Slika 1.15: Prikaz originalne slike (lijevo) i prikaz primjenjenog lokalnog histograma na sliku (desno)

Svi navedeni filteri su implementirani u istoj skripti. U nastavku slijedi prikaz pomenutog koda:

```
from skimage import exposure
import cv2
import glob
import os


def clahehist(img):  # for histogram equalization
    # create a CLAHE object (Arguments are optional).
    clahe = cv2.createCLAHE(clipLimit=3.0, tileGridSize=(8, 8))
    img_yuv = cv2.cvtColor(img, cv2.COLOR_BGR2YUV)
    img_yuv[:, :, 0] = clahe.apply(img_yuv[:, :, 0])
    img_output = cv2.cvtColor(img_yuv, cv2.COLOR_YUV2BGR)
    return img_output


def gamma_correction(img):  # for brigthening (gamma < 1) && contrast (gamma > 1)
    # Gamma
    gamma_corrected = exposure.adjust_gamma(img, 0.5)
    return gamma_corrected


def logarithmic_correction(img):  # for brigthening/contrast
    # Logarithmic
    logarithmic_corrected = exposure.adjust_log(img, 1)
    return logarithmic_corrected


def sigmoid_correction(img):  # for contrast
    # Sigmoid
    sigmoid = exposure.adjust_sigmoid(img, 0.5, 10, False)
    return sigmoid


print("Working on it, please have patience")
src_dir = 'dataset/training_imagery'
dst_dir = 'dataset/filtered_training/'
```

```
37
38 for filename in glob.glob(os.path.join(src_dir, '*')):
39     im = cv2.imread(filename)
40     name = filename.replace(src_dir, '')
41
42     bright = gamma_correction(im)
43     contrast = sigmoid_correction(bright)
44     histogram = clahehist(contrast)
45     cv2.imwrite(dst_dir + name, histogram)
46
47     # cl1 = np.hstack((im, bright, contrast, histogram))  # stacking
           images side-by-side
48     # cv2.imwrite(dst_dir + name + c, cl1)
49 print("Done")
```

Naredne slike prikazuju postepenu primjenu filtera nad jednom sliku, te konačni rezultat.

Slika 1.16: a) Prikaz originalne slike, b) posvijetljena slika, c) posvijetljena slika s poboljšanim kontrastom d)slika s poboljšanim osvjetljenjem, kontrastom i ujednačenim histogramom = konačna slika

Slika 1.17: Konačna slika

2 Projektni zadatak II

2.1 Izbor modela za prepoznavanje koji odgovara problemu

Kao model koji bi zadovoljio naše zahtjeve zadatka, odnosno klasifikaciju malih motornih vozila, odabrana je konvolucijska neuronska mreža. Konvolucijske mreže su zamišljene za obradu podataka koji imaju posebnu topologiju, posebno gdje je osobito važno ostvariti invarijantnost na translaciju. Kako se naš analizirani objekt (vozilo) može naći na bilo kojem mjestu unutar slike, ovakav model predstavlja savršen izbor.

Ideja CNN-a je da se postavi veći broj slojeva za otkrivanje bitnih osobina ulaznih podataka.

Za dizajniranje i treniranje dubokih neuronskih mreža korištena je programska biblioteka Keras. Pored primjenjivanja konvolucijskih filtera na sliku i kreiranja mapa, ovaj model vrši i redukciju rezolucije. Bitan dio za CNN predstavljaju konvolucijski slojevi (*layers*). Oni uzimaju mape na ulazu sloja te rade 2D konvoluciju s jezgrama (*Conv2D()*). Nakon odrađene konvolucije radi se poliranje (*MaxPooling2D()*), odnosno reducira se veličina slike koliko je to moguće. Ovakve slike se proslijede funkciji koja nad njima vrši izravnavanje (*Flatten()* funkcija), nakon čega se vrši kreiranje sloja. Primjer modela koji koristi konvolucijske neuronske mreže i klasificira mala motorna vozila je dan u nastavku.

```
from tensorflow import python as tf
from keras.preprocessing.image import ImageDataGenerator
import pandas as pd


train_datagen = ImageDataGenerator(rescale=1./255,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=False)

test_datagen = ImageDataGenerator(rescale=1./255)



training_set = train_datagen.flow_from_directory('data2/
    masked_rotated_h_train/small vehicle/',
                                                 target_size=(30, 75),
                                                 batch_size=32,
                                                 class_mode='categorical
                                                     ')

test_set = test_datagen.flow_from_directory('data2/masked_rotated_h_test
    /small vehicle/',
                                            target_size=(30, 75),
                                            batch_size=32,
                                            class_mode='categorical')

# Initialising
cnn_classifier = tf.keras.models.Sequential()

# 1st conv. layer
cnn_classifier.add(tf.keras.layers.Conv2D(64, (3, 3), input_shape=(30,
    75, 3), activation='relu'))
```

```
cnn_classifier.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))

# 2nd conv. layer
cnn_classifier.add(tf.keras.layers.Conv2D(128, (3, 3), activation='relu'))
cnn_classifier.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))

# 3nd conv. layer
cnn_classifier.add(tf.keras.layers.Conv2D(256, (3, 3), activation='relu'))
cnn_classifier.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))

# Flattening
cnn_classifier.add(tf.keras.layers.Flatten())


# Full connection
cnn_classifier.add(tf.keras.layers.Dense(units=256, activation='relu'))
cnn_classifier.add(tf.keras.layers.Dropout(0.5))
cnn_classifier.add(tf.keras.layers.Dense(units=6, activation='sigmoid'))

cnn_classifier.summary()
# Compiling the CNN
cnn_classifier.compile(optimizer='adam',
                       loss='categorical_crossentropy',
                       metrics=['accuracy'])

cnn_classifier.fit_generator(training_set,
                             steps_per_epoch=6406,
                             epochs=10,
                             validation_data=test_set,
                             validation_steps=903)

# saving model and weights
cnn_classifier.save_weights('data2/vehicle_classification_weights_dropout.h5')
cnn_classifier.save('data2/vehicle_classification_model_dropout.h5')
```

2.2 Izbor deskriptora koji odgovara problemu

Kako je naš set podataka došao s gotovim *.csv* dokumentom u kojem se nalaze sve detaljno opisane osobine klasificiranog objekta, smatrali smo da nije bilo potrebe praviti dodatni deskriptor, već je korišten ovaj unaprijed definirani. U sljedećoj tablici 2 je prikazano koje sve kategorije posjeduje ovaj deskriptor.

Iz tablice se može primijetiti da svako vozilo posjeduje detaljan opis. Pored samih koordinata, tu su i boja vozila, klasa, te podklasa, kao i mnoge druge osobine vidljive iz tablice 2.

2.3 Izbor metoda poboljšavanja iz 1. Projektnog zadatka koje će biti primijenjene nad slikama

U prvom projektnom zadatku nad slikom su primjenjivani razni filteri za poboljšanje osvijetljenja, kontrasta i histograma. Ove metode korištene se i u nastavku rada.

tag naziv slike	33775 12193
koordinata Px1	3467.501709
koordinata Py1	22.04482651
koordinata Px2	3464.8464924
koordinata Py2	0.608573914
koordinata Px3	3510.498291
koordinata Py3	-5.044826031
koordinata Px4	3513.153076
koordinata Py4	16.39142609
general class subclass	small vehicle sedan
sunroof	0
luggage carrier	0
open cargo area	0
enclosed cab	0
spare wheel	0
wrecked	0
flatbed	-1
ladder	-1
enclosed box	-1
softshell box	-1
enclosed cab	-1
harnesed to cart	-1
ac vents	-1
color	blue

Tablica 2: Prikaz svih osobina koje deskriptor posjeduje

Ulazni podaci za učenje neuronskih mreža nisu uvijek u obliku kakvom bismo željeli da budu, stoga je bilo neizbježno korištenje metoda za pretprocesiranje podataka. Kako je naš region od interesa malo motorno vozilo, uz pomoć deskriptora izdvojena su sva mala motorna vozila na našem setu podataka, koja su zatim rotirana i dimenzionirana tako da svako vozilo ima jednaku rezoluciju. Region od interesa je pretprocesiran sljedećim kodom.

```
import cv2
import numpy as np
import glob
import os
import pandas as pd
import filtering
import denoising
import masking


def subtract_roi(img, image_name, csv):
    image_id_new = int(image_name)
    selection = csv[csv.image_id == image_id_new]
    objects_vehicles = []
```

```
    for idx, row in selection.iterrows():
        point1 = [row.p1_x, row.p1_y]
        point2 = [row.p2_x, row.p2_y]
        point3 = [row.p3_x, row.p3_y]
        point4 = [row.p4_x, row.p4_y]
        objects_points = [[point1, point2, point3, point4]]
        objects_vehicles += objects_points
    mask = np.zeros(img.shape)
    for x in objects_vehicles:
        vrx = np.array(x, np.int32)
        vrx = vrx.reshape((-1, 1, 2))
        mask = cv2.fillPoly(mask, [vrx], (255, 255, 255))
    masked = np.where(mask < 1, img, np.zeros(img.shape))
    return masked


print("Working on it, please have patience")
src_dir = 'data2/train'
dst_dir = 'data2/preprocessed_train'
df = pd.read_csv('data2/train.csv', sep=',')
br = 0
for filename in glob.glob(os.path.join(src_dir, '*.jpg')):
    im = cv2.imread(filename)
    name = filename.replace(src_dir, '')
    img_name = name.replace('.jpg', '')
    img_name = img_name.replace('\\', '')
    denoised = denoising.denoise(im)
    bright = filtering.gamma_correction(denoised)
    contrast = filtering.sigmoid_correction(bright)
    histogram = filtering.clahehist(contrast)
    subtracted = subtract_roi(im, img_name, df)
    mask1 = masking.create_mask(histogram, img_name, df)
    dst = cv2.addWeighted(subtracted, 1, mask1, 1, 0)
    cv2.imwrite(dst_dir + name, dst)
    br += 1
    print(br)
print("Done")
```

Nakon što je region od interesa pripremljen, sva motorna vozila su izdvojena zasebno sa slika i rotirana. Ovo je napravljeno iz razloga što se pribjegavalo da model dubokog učenja ima ulazne podatke standardnih vrijednosti.

Kod koji vrši rotaciju vozila je sljedeći:

```
import numpy as np
import cv2
import pandas as pd
import imutils
import glob
import os


def rotate(im, tag, df):
    image_id_new = int(tag)
    selection = df[df.tag_id == image_id_new]

    objects_points = np.array([])
    for idx, row in selection.iterrows():
        if str(row.tag_id) == tag:
```

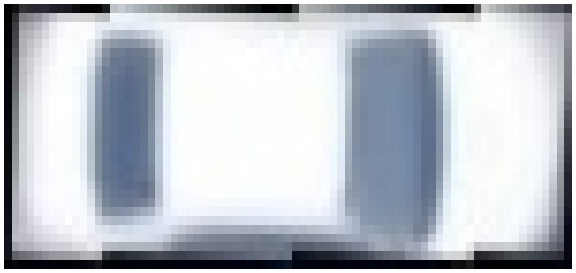
```
            point1 = [int(row.p1_x), int(row.p1_y)]
            point2 = [int(row.p2_x), int(row.p2_y)]
            point3 = [int(row.p3_x), int(row.p3_y)]
            point4 = [int(row.p4_x), int(row.p4_y)]
            objects_points = np.array([point1, point2, point3, point4])

    rect = cv2.minAreaRect(objects_points)
    angle = rect[2]
    rotated = imutils.rotate_bound(im, -angle)

    image = rotated
    edged = cv2.Canny(image, 10, 250)
    cnt = cv2.findContours(edged, cv2.RETR_EXTERNAL, cv2.
        CHAIN_APPROX_SIMPLE)[0]
    x, y, w, h = cv2.boundingRect(cnt)
    croped = image[y:y + h, x:x + w]
    return croped


print("Working on it, please have patience")
src_dir = 'data/masked_real_val/large vehicle/truck'
df = pd.read_csv('data/train.csv', sep=',')
br = 0
for filename in glob.glob(os.path.join(src_dir, '*.jpg')):
    im = cv2.imread(filename)
    name = filename.replace(src_dir, '')
    img_name = name.replace('.jpg', '')
    img_name = img_name.replace('\\', '')
    rot = rotate(im, img_name, df)
    cv2.imwrite('data/masked_rotated_val/large vehicle/truck/' +
        img_name + '.jpg', rot)
    br += 1
    print(br)
print("Done")
```

Kako bi region od interesa bio uniforman, sve slike su dodatnom rotacijom postavljene horizontalno. Ovim poboljšanjima se smanjila varijacija između pojedinih osobina regiona od interesa.

```
import numpy as np
import cv2
import pandas as pd
import imutils
import glob
import os


def rotate_horizontal_and_resize(im, size):
    if len(im) > len(im[0]):
        rotated_h = imutils.rotate_bound(im, 90)
    else:
        rotated_h = im
    resized_image = cv2.resize(rotated_h, size)
    return resized_image


print("Working on it, please have patience")
src_dir = 'data/masked_rotated_val/small vehicle/van'
```

```
br = 0
for filename in glob.glob(os.path.join(src_dir, '*.jpg')):
    im = cv2.imread(filename)
    name = filename.replace(src_dir, '')
    img_name = name.replace('.jpg', '')
    img_name = img_name.replace('\\', '')
    rot = rotate_horizontal_and_resize(im, (75, 30))
    cv2.imwrite('data/masked_rotated_h_val/small vehicle/van/' +
        img_name + '.jpg', rot)
    br += 1
    print(br)
print("Done")
```

Sljedeća slika prikazuje jedan takav izdvojen region od interesa.



Slika 2.1: Primjer jednog izdvojenog regiona od interesa

2.4 Izračunavanje performansi modela: sp, sens, acc

Testiranjem konačne verzije modela sa testnim podacima, kao parametri sensitivity, accuracy i specificity za svaku od podklasa promatranog regiona dobiveni su sljedeći rezultati prikazani u tablici 3.

klasa	sensitivity	specificity	accuracy	precision
hatchback	0.64	0.82	0.76	0.62
jeep	0.27	0.96	0.89	0.42
minivan	0.19	0.97	0.93	0.22
pickup	0.47	0.99	0.98	0.78
sedan	0.88	0.80	0.84	0.80
van	0.52	0.99	0.98	0.55

Tablica 3: Performansi modela: sens, sp, acc, prec

2.5 Poboljšavanje performansi modela za prepoznavanje na osnovu performansi testiranja modela

Kako se kao model za prepoznavanje koristila konvolucijska neuronska mreža, prilikom testiranja modela povećavao se i smanjivao broj slojeva neuronske mreže. Veći broj slojeva daje bolju točnost, međutim to je drastično utjecalo na brzinu izvršavanja koda. Za treniranje ovakve neuronske mreže trebalo je i po pet sati da se algoritam izvrši.

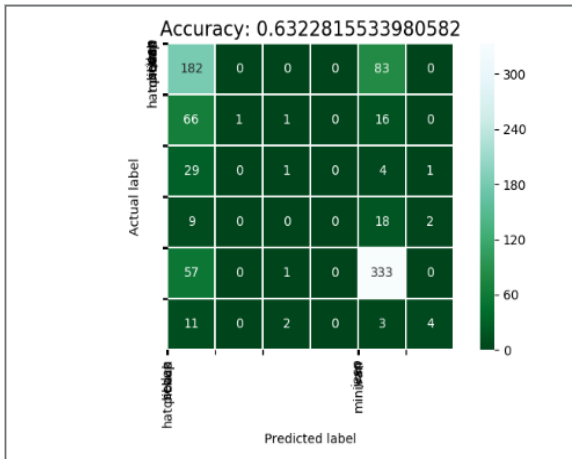
Prvi rezultati dobiveni s maskiranim slikama, bez posebno izdvojenog svakog regiona od interesa nisu bili zadovoljavajući. Naime, točnost našeg modela je bila 9%, što je poražavajuće loše. Za učenje neuronske mreže s izdvojenim regionima od interesa i promjenom parametara slojeva, točnost se mijenjala.

Zbog dugog vremenskog izvršavanja algoritma, podaci za testiranje i treniranje su izmijenjeni. Skup podataka je znatno reduciran. Izbačeni su svi formati slika koji nisu *.jpg*, jer ovaj format uzima najmanje vremena pri izvršavanju.

Zbog korištenja konvolucijske neuronske mreže kao modela za prepoznavanje, sve slike su transformirane u format koji podržava CNN pozivom funkcije *img_to_array()* iz biblioteke Keras i *expand_dim()* iz biblioteke NumPy.

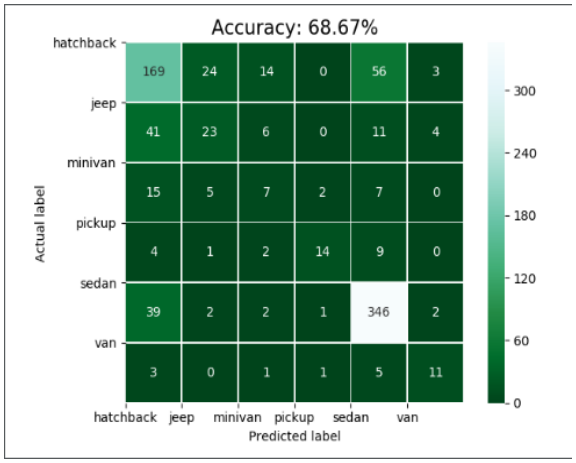
Također je i klasifikacija vozila znatno reducirana. Klasificirana su samo mala vozila na šest skupina: van, sedan, pickup, hatchback, jeep i minivan.

Pri jednom od testiranja, točnost je dosegnula nešto više od 60%. Sljedeća slika prikazuje točnost modela, te broj pogođenih vozila.



Slika 2.2: Primjer dobivenih rezultata nakon treniranja neuronske mreže

Dodatnim treniranjem, mijenjanjem broja slojeva i epoha modela, dobiven je krajnji rezultat koji smatramo zadovoljavajućim. Njegova točnost je približno 69%, te su dobiveni rezultati prikazani na slici ispod.



Slika 2.3: Primjer krajnjih dobivenih rezultata nakon treniranja neuronske mreže

2.6 Export modela za prepoznavanje

Linije koda koje exportuju model za prepoznavanje su prikazane u cnn.py skripti prikazanoj u poglavlju 2.1. :

```
cnn_classifier.save_weights('vehicle_classification_weights_dropout.h5')
cnn_classifier.save('vehicle_classification_model_dropout.h5')
```

3 Projektni zadatak 3

U nastavku je prikazana main funkcija koja kao parametar prima putanju foldera u kojem se nalaze slike za klasifikaciju.

```
from tensorflow import python as tf
import numpy as np
import os
import matplotlib.pyplot as plt
from random import shuffle
from label_vehicles import label_vehicle
import pandas as pd
import glob
import cv2
from preprocessing import preprocessing
from rotate import rotate
from rotate_horizontal import rotate_horizontal_and_resize
from preparation import create_mask_of_tag
from load_model import pretty_cm, transform_image, evaluation_indices,
    report


def main(path):
    print("Working on it, please have patience")
    src_dir = 'data2/' + path
    df = pd.read_csv('data2/val_csv.csv', sep=',')
    br = 0

    for filename in glob.glob(os.path.join(src_dir, '*.jpg')):
        im = cv2.imread(filename)
        name = filename.replace(src_dir, '')
        img_name = name.replace('.jpg', '')
        img_name = img_name.replace('\\', '')
        preprocessed = preprocessing(im, img_name, df)
        create_mask_of_tag(preprocessed, img_name, df)
        br += 1
        print(br)
    src_dir = 'data2/masked_val/small vehicle'
    dst_dir = 'data2/masked_rotated_h_val/'
    for filename in glob.glob(os.path.join(src_dir, '*.jpg')):
        im = cv2.imread(filename)
        name = filename.replace(src_dir, '')
        tag_name = name.replace('.jpg', '')
        tag_name = tag_name.replace('\\', '')
        rot = rotate(im, tag_name, df)
        rot_h = rotate_horizontal_and_resize(rot, (30, 75))
        tag_id_new = int(tag_name)
        selection = df[df.tag_id == tag_id_new]

        for idx, row in selection.iterrows():
            if str(row.general_class) == "small vehicle":
                add_dst = 'small vehicle/'
                if str(row.sub_class) == "sedan":
                    cv2.imwrite(dst_dir + add_dst + 'sedan/' + str(row.
                        tag_id) + '.jpg', rot_h)
                elif str(row.sub_class) == "hatchback":
                    cv2.imwrite(dst_dir + add_dst + 'hatchback/' + str(
                        row.tag_id) + '.jpg', rot_h)
```

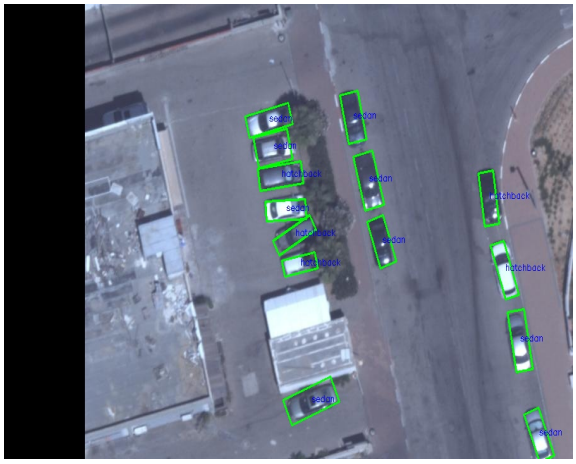
```
                elif str(row.sub_class) == "minivan":
                    cv2.imwrite(dst_dir + add_dst + 'minivan/' + str(row
                        .tag_id) + '.jpg', rot_h)
                elif str(row.sub_class) == "van":
                    cv2.imwrite(dst_dir + add_dst + 'van/' + str(row.
                        tag_id) + '.jpg', rot_h)
                elif str(row.sub_class) == "jeep":
                    cv2.imwrite(dst_dir + add_dst + 'jeep/' + str(row.
                        tag_id) + '.jpg', rot_h)
                elif str(row.sub_class) == "pickup":
                    cv2.imwrite(dst_dir + add_dst + 'pickup/' + str(row.
                        tag_id) + '.jpg', rot_h)
        br += 1
        print(br)

    # preparing data for predictions
    size = (30, 75)
    X_eval = list()
    y_eval = list()
    X_tag = list()

    # hatchback
    files = os.listdir('data2/masked_rotated_h_val/small vehicle/
        hatchback/')
    files.sort()

    for i in range(0, len(files)):
        X_eval.append(transform_image('data2/masked_rotated_h_val/small
            vehicle/hatchback/' + files[i], size))
        y_eval.append(0)
        tag = files[i].replace('.jpg', '')
        X_tag.append(int(tag))
    # jeep
    files = os.listdir('data2/masked_rotated_h_val/small vehicle/jeep/')
    files.sort()

    for i in range(0, len(files)):
        X_eval.append(transform_image('data2/masked_rotated_h_val/small
            vehicle/jeep/' + files[i], size))
        y_eval.append(1)
        tag = files[i].replace('.jpg', '')
        X_tag.append(int(tag))
    # minivan
    files = os.listdir('data2/masked_rotated_h_val/small vehicle/minivan
        /')
    files.sort()

    for i in range(0, len(files)):
        X_eval.append(transform_image('data2/masked_rotated_h_val/small
            vehicle/minivan/' + files[i], size))
        y_eval.append(2)
        tag = files[i].replace('.jpg', '')
        X_tag.append(int(tag))
    # pickup
    files = os.listdir('data2/masked_rotated_h_val/small vehicle/pickup/
        ')
    files.sort()

```

```
    for i in range(0, len(files)):
        X_eval.append(transform_image('data2/masked_rotated_h_val/small vehicle/pickup/' + files[i], size))
        y_eval.append(3)
        tag = files[i].replace('.jpg', '')
        X_tag.append(int(tag))
    # sedan
    files = os.listdir('data2/masked_rotated_h_val/small vehicle/sedan/')
    files.sort()

    for i in range(0, len(files)):
        X_eval.append(transform_image('data2/masked_rotated_h_val/small vehicle/sedan/' + files[i], size))
        y_eval.append(4)
        tag = files[i].replace('.jpg', '')
        X_tag.append(int(tag))
    # van
    files = os.listdir('data2/masked_rotated_h_val/small vehicle/van/')
    files.sort()

    for i in range(0, len(files)):
        X_eval.append(transform_image('data2/masked_rotated_h_val/small vehicle/van/' + files[i], size))
        y_eval.append(5)
        tag = files[i].replace('.jpg', '')
        X_tag.append(int(tag))
    # stacking the arrays
    X_eval = np.vstack(X_eval)

    labels_index = {0: "hatchback", 1: "jeep", 2: "minivan", 3: "pickup", 4: "sedan", 5: "van"}

    # load the model
    cnn_classifier = tf.keras.models.load_model('data2/vehicle_classification_model_dropout.h5')

    cnn_pred = cnn_classifier.predict_classes(X_eval, batch_size=32)

    pretty_cm(cnn_pred, y_eval, labels_index)
    correctly_classified_indices, misclassified_indices = evaluation_indices(cnn_pred, y_eval)

    plt.figure(figsize=(36, 6))
    shuffle(correctly_classified_indices)
    plt.show()

    for plot_index, good_index in enumerate(correctly_classified_indices[0:5]):
        plt.subplot(1, 5, plot_index + 1)
        plt.imshow(X_eval[good_index])
        plt.title('Predicted: {}, Actual: {}'.format(labels_index[cnn_pred[good_index]],
                                                     labels_index[y_eval[good_index]]),
                                                     fontsize=5)
    plt.show()
```

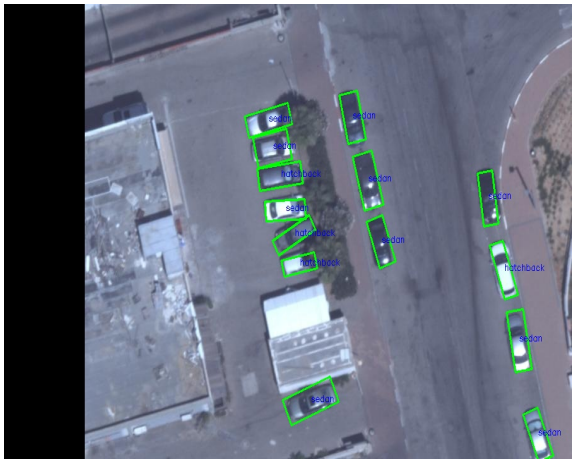
```
146     print(X_tag)
147     print(list(cnn_pred))
148     print(y_eval)
149     predicted_dict = dict(zip(X_tag, list(cnn_pred)))
150     actual_dict = dict(zip(X_tag, y_eval))
151 
152     print(len(X_tag))
153     print(len(list(cnn_pred)))
154     print(len(y_eval))
155 
156     df = pd.read_csv('data2/val_csv.csv', sep=',')
157     path = 'data2/val'
158     src_dir = path
159     dst_dir = 'data2/labeled_predicted_val'
160     for filename in glob.glob(os.path.join(src_dir, '*.jpg')):
161         im = cv2.imread(filename)
162         name = filename.replace(src_dir, '')
163         img_name = name.replace('.jpg', '')
164         img_name = img_name.replace('\\', '')
165         labeled_img_p = label_vehicle(im.copy(), img_name, df,
              predicted_dict)
166         cv2.imwrite(dst_dir + name, labeled_img_p)
167         labeled_img_a = label_vehicle(im.copy(), img_name, df,
              actual_dict)
168         cv2.imwrite('data2/labeled_actual_val' + name, labeled_img_a)
169 
170     report(y_eval, list(cnn_pred), labels_index)
171     print("Done")
172 
173 
174 if __name__ == "__main__":
175     main('val')
```

3.1 Labelirane slike

Na svim ulaznim slikama su označena sva vozila, te se pored njih nalazi opis podklase kojoj pripadaju. Isto je urađeno i sa rezultatima dobivenim nakon izvršenja modela. Slika 3.1 prikazuje stvarna vozila i naziv njihovih podklasa, dok slika 3.2 prikazuje vozila klasificirana ovim modelom.



Slika 3.1: Prikaz regiona od interesa s nazivom njegove podklase na jednoj slici



Slika 3.2: Prikaz prepoznatih vozila i njihova klasifikacija korištenjem konvolucijske neuronske mreže

Za labeliranje svih objekata korištene su metode *drawcontour()*, koja na osnovu vraćenih koordinata crta pravougaonike oko detektiranih vozila, te ispisuje u njih njihovu prepoznatu podklasu.

```
import numpy as np
import cv2
import pandas as pd
import glob
import os


def label_vehicle(img, image_name, csv, dict):
    image_id_new = int(image_name)
    selection = csv[csv.image_id == image_id_new]
    windowed_with_name = img
    labels_index = {0: "hatchback", 1: "jeep", 2: "minivan", 3: "pickup"
        , 4: "sedan", 5: "van"}
    for idx, row in selection.iterrows():
        p1 = [row.p1_x, row.p1_y]
        p2 = [row.p2_x, row.p2_y]
        p3 = [row.p3_x, row.p3_y]
        p4 = [row.p4_x, row.p4_y]

        contour = [p1, p2, p3, p4]
        ctr = np.array(contour).reshape((-1, 1, 2)).astype(np.int32)
        windowed = img
        if str(row.general_class) == 'small vehicle':
            windowed = cv2.drawContours(img, [ctr], -1, (0, 255, 0), 2)

        miniX = min(row.p1_x, row.p2_x, row.p3_x, row.p4_x)
        maxiX = max(row.p1_x, row.p2_x, row.p3_x, row.p4_x)

        miniY = min(row.p1_y, row.p2_y, row.p3_y, row.p4_y)
        maxiY = max(row.p1_y, row.p2_y, row.p3_y, row.p4_y)

        x = round(miniX + (maxiX - miniX) / 2)
        y = round(miniY + (maxiY - miniY) / 2)
        for tag, value in dict.items():
```

```
34              if tag == int(row.tag_id):
35                  windowed_with_name = cv2.putText(windowed, labels_index.
                        get(value),
36                                                   (x, y), cv2.
                                                       FONT_HERSHEY_SIMPLEX
                                                       , 0.4, 255)
37
38      return windowed_with_name
```