**Tema: Cat Face Detection**

Prepoznavanje oblika i obrada slike

**Studenti:**

Armin Šarak

Berina Smajović

Lejla Solak

**SADRŽAJ**

[**1.** **Opis teme** 2](#_Toc533366301)

[**2.** **Dataset** 2](#_Toc533366302)

[**3.** **DataPreprocessing** 2](#_Toc533366303)

[3.1. Uklanjanje šuma 3](#_Toc533366304)

[3.2. Poboljšanje kvaliteta slika 5](#_Toc533366305)

[3.3. Podjela na disjunktne skupove 6](#_Toc533366306)

[**4. Izbor modela za prepoznavanje, deskriptora i metoda poboljšavanja** 7](#_Toc533366307)

[4.1. Model za prepoznavanje 7](#_Toc533366308)

[4.2. Deskriptor 7](#_Toc533366309)

[4.3. Metode za poboljšavanje 7](#_Toc533366310)

[**5. Kreiranje deskriptora** 7](#_Toc533366311)

[**6. Kreiranje i treniranje modela za prepoznavanje** 7](#_Toc533366312)

[**7. Testiranje modela za prepoznavanje** 8](#_Toc533366313)

[**8. Poboljšavanje performansi modela za prepoznavanje** 9](#_Toc533366314)

[**9. Eksport modela** 10](#_Toc533366315)

[**10. Main** 10](#_Toc533366316)

# **Opis teme**

Odabrana tema za projekat iz predmeta Prepoznavanje oblika i obrada slike je Cat Face Detection u okviru koje će se vršiti prepoznavanje lica mačaka na slikama iz odabranog dataseta.

Link na projekat: <https://github.com/berinas/cat_detection>

# **Dataset**

Obzirom da je u pitanju binarni (DA/NE) klasifikator, kreirani dataset se sastoji od dvije klase. Jedna klasa („*cat*“) sadrži slike sa mačkama a druga slike sa različitim objektima („*non\_cat*“). Svaka od ovih klasa broji po 20 slika što je 40 slika ukupno u data setu.

# **DataPreprocessing**

Označavanje regiona od interesa je izvedeno ručno korištenjem *Labelbox*[[1]](#footnote-1) platforme, a anotacije su sačuvane kao xlsx fajl („*anotacije*“).

Slike za finalnu validaciju su izdvojene u poseban folder („*slike\_finalnaValidacija*“) koji sadrži ukupno 4 slike (po dvije iz svake klase).

1. Kreiranje maski na osnovu regiona od interesa:

createAndApplyMask(imeSlike, upL, bottomR)

1. Primjena maske na slike:

maskedImg = useMask(img, mask)

1. Spremanje nove slike u novi folder:

saveImage(maskedImg, path, imgName)



*Slika1*. Primijenjena maska (lijevo)

## Uklanjanje šuma

Za uklanjanje šuma korišten je mediana filter koji se bazira na medijani uzorka. Efikasniji je od filtera na bazi prosjeka kod smanjenja šuma jer je upotrebom ovog filtera manji efekat zamagljenja slike.

1. Uklanjanje šuma:

processed\_image = denoiseRemoval(image)

Slike sa primjenjenim mediana filterom su smještene u novi folder („*uklanjanje\_suma*“).

Maskiranje neoštrina je izvedeno nad slikama iz prethodnog koraka, obzirom da se primjenom filtera za uklanjanje šuma dobije efekat blagog zamagljenja slike.



*Slika2*. Blago zamagljena slika nakon primjene mediana filtera (desno)

1. Maskiranje neoštrina:

unsharp\_image = unsharpMasking(image)



*Slika3*. Blago izoštrena slika (desno)

## Poboljšanje kvaliteta slika

Obzirom da filteri za poboljšavanje kontrasta slike ne odgovaraju najbolje slikama iz dataset-a, korišten je filter sa efektom crno-bijele slike.

1. Crno-bijeli efekat:

new\_image = toBlackAndWhite(image)

Crno-bijeli efekat je postignut korištenjem metode *convert* klase *Image* koja vraća konvertovanu kopiju slike. Korišten je „L“ mod za konverziju u crno-bijelu sliku.

https://i.snag.gy/FxVNMG.jpg

Kada se prevodi slika u boji u crno-bijelu sliku (mod „L“) , biblioteka koristi ITU-R 601-2 luma transformaciju:

L = R \* 299/1000 + G \* 587/1000 + B \* 114/1000



*Slika4*. Primijenjen crno-bijeli efekat

1. Poboljšanje osvjetljenja (*brightness*):

new\_image = changeBrightness(image)

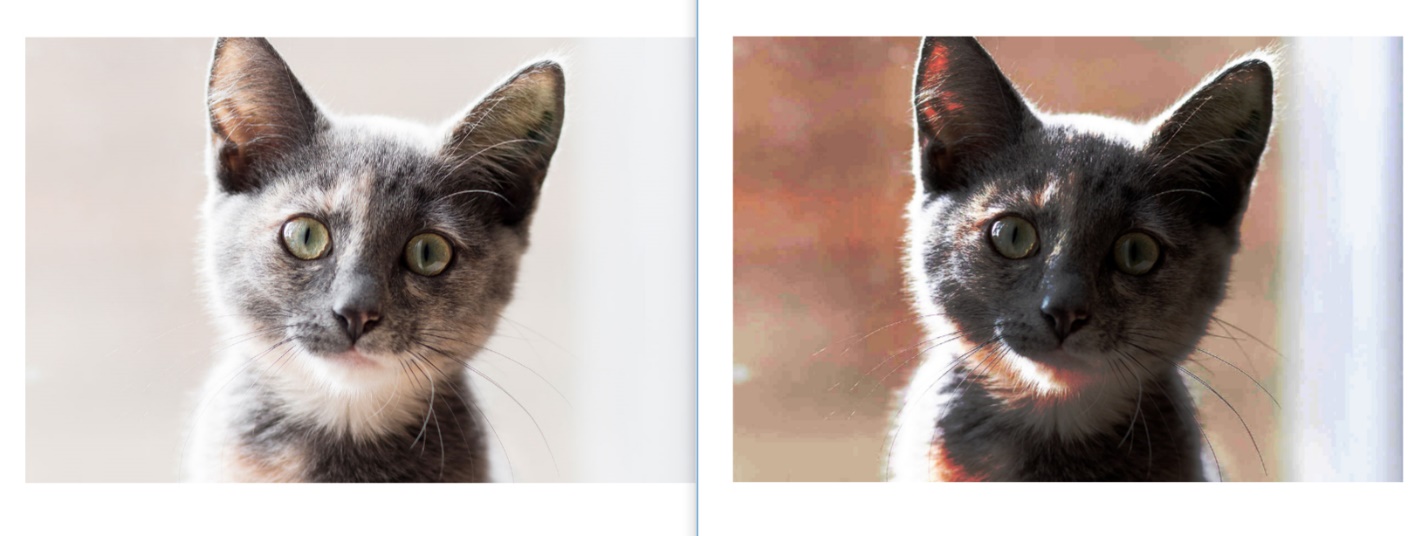
Za poboljšanje osvjetljenja korišten je python modul *ImageEnhance* koji sadrži klasu *Brightness* koja kontroliše osvjetljenje slike. *ImageEnhance*.*Brightness(image)* metoda kreira „enhancement“ objekat za podešavanje osvjetljenja slike. Faktor 1.0 daje kopiju originalne slike a faktor 0.0 daje crnu sliku.



*Slika5.* Promjena osvjetljenja

1. Ujednačavanja histograma:

new\_image = histogramEq(image)



*Slika6.* Ujednačavanje histograma

## Podjela na disjunktne skupove

Slike su podijeljene na dva disjunktna podskupa train i test smještena u dva nova foldera („*test*“, „*train*“):

train\_data ,test\_data = train\_test\_split(image\_list, test\_size=0.2)

# **4. Izbor modela za prepoznavanje, deskriptora i metoda poboljšavanja**

## Model za prepoznavanje

Izabrani model za prepoznavanje je svm.SVC koji se kreira na sljedeći način:

clf = svm.SVC(kernel=**'linear'**, C=1.0)

## Deskriptor

Deskriptor koji će biti korišten je HOG (Histogram of Oriented Gradient).

## Metode za poboljšavanje

Metode za poboljšavanje iz projektnog zadatka 1 koje će biti korištene su osvjetljenje i ujednačavanje histograma. Maske su crop-ane na regione od interesa i smještene u poseban folder („*cropped\_masks*“ ) nakon čega su nad njima su primijenjene navedene metode za poboljšavanje („*poboljsane\_slike*“). Nad tim slikama će se raditi i u nastavku.

new\_image = poboljsaj(image)

# **5. Kreiranje deskriptora**

Kreiranje odabranog deskriptora HOG uključuje konvertovanje slike u grayscale i resize slike.

Način pozivanja funkcije koja kreira deskriptor svih slika je sljedeći:

fd, hog\_image = createDescriptor(image)

# **6. Kreiranje i treniranje modela za prepoznavanje**

Nakon kreiranja modela kako je navedno ranije, treniranje se vrši na sljedeći način:

clf.fit(np.array(trainData), labels)

*trainData* – HOG features cijelog seta za treniranje

*labels* – oznake klasa (0-not cat; 1-cat)

# **7. Testiranje modela za prepoznavanje**

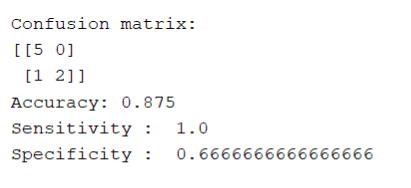
Obavljanje testiranja modela za prepoznavanje je prikazano sljedećim kodom:

y\_pred = clf.predict(testData)

Nakon testiranja, performanse se računaju kao:

conf = metrics.confusion\_matrix(labels, y\_pred)  
print(**"Confusion matrix:\n%s"** % conf)  
  
acc = accuracy\_score(y\_pred, labels)  
print(**"Accuracy: {}"**.format(acc))  
  
sensitivity1 = conf[0,0]/(conf[0,0]+conf[0,1])  
print(**'Sensitivity : '**, sensitivity1)  
  
specificity1 = conf[1,1]/(conf[1,0]+conf[1,1])  
print(**'Specificity : '**, specificity1)

Rezultat je:

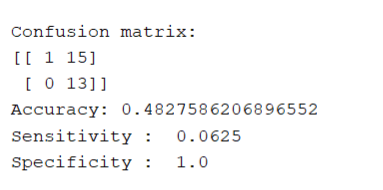


# **8. Poboljšavanje performansi modela za prepoznavanje**

1. Drugačija podjela podataka na trening/test skup

train\_data ,test\_data = train\_test\_split(image\_list, test\_size=0.8)

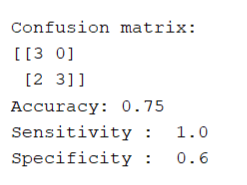
Rezultat je:



1. Izmjena parametara odgovarajućeg modela

clf = svm.SVC(kernel=**'poly'**, C=12.5)

Rezultat je:



# **9. Eksport modela**

Eksport modela se obavlja na sljedeći način:

filename = **'finalized\_model.sav'**joblib.dump(clf, filename)

# **10. Main**

Main funkcija prima kao parametar putanju foldera u kojem su slike za finalnu validaciju izdvojene na samom početku.

main(path)

U main funkciji se poziva funkcija za poboljšavanje slika iz navedenog foldera, implementirana ranije:

new\_image = poboljsaj(image)

Kreiraju se i deskriptori koji su korišteni pri treniranju:

hf, h\_image = createDescriptor(image)

Koristeći eksportovani model za prepoznavanje oblika radi se detekcija objekata na slikama. Implementiran je sliding window kojeg u main-u pozivamo na sljedeći način:

indices, patches = zip(\*sliding\_window(new\_image))

Crtanje pravougaonika na slikama je izršeno sljedećim kodom:

fig, ax = plt.subplots()

ax.imshow(test\_image, cmap='gray')

ax.axis('off')

Ni, Nj = positive\_patches[0].shape

indices = np.array(indices)

for i, j in indices[labels == 1]:

ax.add\_patch(plt.Rectangle((j, i), Nj, Ni, edgecolor='red',

alpha=0.3, lw=2, facecolor='none'))

1. Link na stranicu: <https://www.labelbox.com/> [↑](#footnote-ref-1)