UNIVERZITET U TUZLI
FAKULTET ELEKTROTEHNIKE

SEMINARSKI RAD

Prepoznavanje uzoraka

Klasifikacija ptica

Student: Haseljić Eldar

Profesor:
Dr.sc. Damir Demirović
vanr.prof.

Tuzla, april/travanj 2020.

Sažetak

U ovom seminarskom radu ćemo govoriti o klasifikaciji ptica, pomoću jednog od kernela koji su dati na stranici na sljedećem linku. Polako kako budemo prolazili koroz jednu od kernel skripti sa stranice, vršit ćemo pregled i pojašnjenje šta koja od linija koda radi. Uz pomoć ovog klasifikatora imamo mogućnost klasifikacije ptice čija je tačnost 93%. Također ćemo napraviti jedan kraći uvod o pticama, navesti neke njihove najvažnije karateristike te navesti neke važne stvari vezane za ptice.

Ključne riječi: ptica, klasifikator, CNN, konvolucija

Abstract

In this seminar work we will talk about the classification of birds, using one of the kernels provided on the page at the following link. Slowly, as we go through one of the kernel scripts from the page, we'll review and clarify what each of the code line does. With this classification there is a possibility of bird classification which is 93% accuracy. We'll also make a brief introduction about birds, outline some of their greatest features, and list some important things about birds.

Keywords: bird, classifier, CNN, convolution

Sadržaj

Popis slika

Popis skraćenica

CNN - Convolutional Neural Networks

GPU - Graphics Processing Unit

TPU - Tensor Processing Unit

1. Uvod

Slika 1.Golub pećinar

Ptice (lat. *aves*) su razred dvonožnih, toplokrvnih kralježnjaka koji polažu jaja. Ptice su tijekom jure evoluirale od dinosaura podreda *Theropoda*, a najranija poznata ptica iz kasne jure je *Archaeopteryx*. Veličinom variraju od sitnih kolibrića do krupnih nojeva. Postoji između 9 i 10 tisuća poznatih vrsta ptica i najraznovrsniji su razred među kopnenim kralježnjacima (Teropoda). Današnje ptice odlikuju perje, kljun bez zuba, polaganje jaja sa čvrstom ljuskom, visoka stopa metabolizma, srce s dvjema komorama i dvjema pretkomorama, te lagan ali jak kostur. Mnoge ptice imaju prednje udove razvijene kao krila kojima mogu letjeti, iako su nojevke i nekoliko drugih, poglavito endemskih otočnih vrsta, izgubile tu sposobnost. Mnoge ptičje vrste svake godine kreću na selidbe u udaljene krajeve, a još više ih poduzima migracije koje su kraće i manje redovne. Ptice su društvene životinje i komuniciraju vizualnim signalima, glasovnim pozivima i pjevanjem, sudjeluju u društvenom ponašanju što uključuje zajednički lov, pomoć pri odgajanju podmlatka i ponašanje karakteristično za jato.

Neke vrste ptica su isključivo monogamne, druge prvenstveno monogamne uz povremeno parenje s drugim jedinkama. Druge vrste su poligamne ili poliandrične. Jaja obično polažu u gnijezdima gdje se ona inkubiraju, i većina ptica produljeno vrijeme provodi u podizanju mladih nakon izlijeganja. Ljudi iskorištavaju ptice kao važan izvor hrane kroz lov i peradarstvo. Neke vrste, poglavito pjevice i papige omiljene su kao kućni ljubimci. Ptice su istaknuto zastupljene u svim pogledima ljudske kulture, od religije, poezije do popularne glazbe. Oko 120 do 130 ptičjih vrsta izumrlo je kao rezultat ljudskog djelovanja od 1600. godine, a prije toga još i više. Danas mnogim vrstama ptica prijeti izumiranje i zbog različitih ljudskih aktivnosti, pa se ulažu napori kako bi ih se zaštitilo.

Slika 2.Primjeri raznih vrsta ptica

2. Data set

U ovom seminarskom radu cemo koristit data set sa linka, te u prilogu ćemo navesti neke osnovne karakteristike istog.

2.1. Opis data seta

Skup podataka u trenutnku sastavljanja seminarskog rada je sadržavao 170 vrsta ptica, 22914 slika s treninga, 850 testnih slika (5 po vrsti) i 850 validacijskih slika (5 po vrsti). Sve slike su dimenzija 224 x 224 x 3 u boji u jpg formatu. Data set također uključuje "konsolidirani" skup slika koji kombinira slike vježbanja, testiranja i provjere valjanosti u jedan skup podataka.

Ovo je korisno za korisnike koji žele stvoriti vlastite skupove za obuku, testiranje i provjeru valjanosti. Slike svake vrste nalaze se u zasebnom poddirektoriju,te je ovo zgodno ako koristite Keras protok iz direktorija kao sredstvo za unos podataka. Slike su prikupljene iz internetskih pretraživanja po imenu vrste. Nakon što su preuzete slikovne datoteke za neku vrstu, izvršena je provjra duplikata između slika pomoću python programa za otkrivanje dvostrukih slika. Svi otkriveni duplikati izbrisani su kako bi se spriječilo da budu zajedničke slike između treninga, ispitivanja i provjere valjanosti. Nakon toga slike su obrezane tako da ptica zauzima najmanje 50% piksela na slici. Potom su slike promijenjene u veličinu 224 x 224 x 3 u jpg formatu. Obrezivanje osigurava da, kada ih CNN obrađuje, odgovarajuće informacije na slikama stvaraju vrlo precizan klasifikator. Sve su datoteke također numerirane uzastopno, počevši od jedan za svaku vrstu. Tako su testne slike nazvane 1.jpg do 5.jpg. Slično vrijedi i za validacijske slike. Slike sa treninga također su numerirane sukcesivno s "nulama", npr. 001.jpg, 002.jpg itd. Skup za treniranje nije uravnotežen, ima različit broj datoteka po vrstama. Međutim, svaka vrsta ima najmanje 100 datoteka sa slikama. Ta neuravnoteženo nije utjecala na klasifikator kernela jer je na testnom setu postignuto 98% točnosti.Jedna značajna neravnoteža u skupu podataka jest omjer slika muških vrsta i slika ženskih vrsta. Otprilike 80% slika su muškarci i 20% žene. Mužjaci koji su tipični su mnogo raznoliko obojeni, dok su ženke malo blaže obojene. Gotovo sve slike ispitivanja i validacije uzimaju se od mužjaka vrste. Zbog toga klasifikator možda neće biti dobro izveden na slikama ženskih vrsta.

3. Perje kao najuočljivija karakteristika

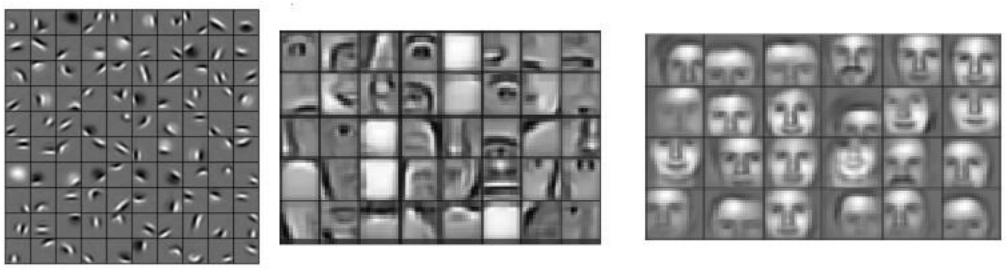
Najuočljivija karakteristika koja ptice ističe među ostalim živim bićima je pokrivenost perjem. Perje je epidermalni izrast pričvršćen za kožu koji ima nekoliko funkcija: pomaže u održavanju tjelesne topline štiteći pticu od hladnog vremena i vode, neophodno je za letenje, a također služi za davanje signala, kamuflažu i pokazivanje. Postoji nekoliko tipova perja od kojih svaki ima posebnu zadaću. Perje zahtijeva održavanje i ptice ga svojim kljunovima svakodnevno čiste i uređuju, uklanjajući strana tjelešca i nanoseći sloj voštane izlučevine iz specijalizirane žlijezde koji pomaže u održanju savitljivosti, a također ima i ulogu zaštite protiv mikroba, čime se zaustavljaju bakterije koje uzrokuju raspadanje perja. Taj proces može biti potpomognut i trljanjem perja tijelima insekata koji luče mravlju kiselinu, što je vrlo čest način kojim ptice uklanjaju parazite iz perja.

Slika 3. Perje afričkog ćuka *Otus senegalensis* omogućuje joj kamuflažu

4. Razni klasifikatori ptica

4.1. Simple Fastai tutorijal

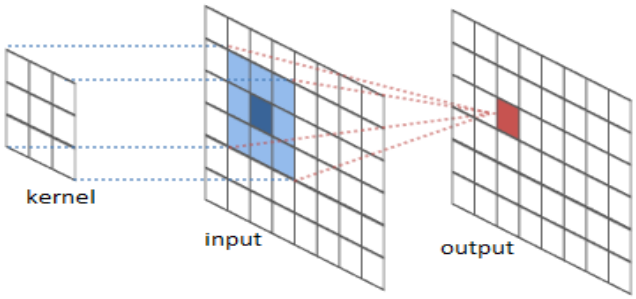
Feature engineering je postupak izdvajanja korisnih uzoraka iz ulaznih podataka koji će modelu predviđanja bolje razumjeti stvarnu prirodu problema. Dobra osobina učenja predstavit će uzorke na način koji značajno povećava tačnost i performanse primijenjenih algoritama mašinskog učenja na način koji bi u suprotnom bio nemoguć ili preskup samim mašinskim učenjem. Algoritmi učenja uzoraka nalaze zajedničke uzorke koji su važni za razlikovanje željenih klasa i automatski ih izdvajaju. Nakon ovog postupka, spremni su za upotrebu u klasifikaciji ili regresijskom problemu. Velika prednost CNN-ova je što su oni neobično dobri u pronalaženju karakteristika u slikama koje rastu nakon svake razine, što rezultira vrhunskim uzorcima na kraju. Završni slojevi (mogu biti jedan ili više) koriste sve ove generirane uzorke za klasifikaciju ili regresiju.U osnovi, Convolutional Neural Networks je vaš najbolji prijatelj da automatski radite Feature Engineering bez da trošite previše vremena stvarajući vlastite kodove i bez prethodne potrebe za ekspertizom u polju Feature Engineeringa.



Slika 4.Primjer učenja uzoraka, počevši od manjih osobina do većih osobina

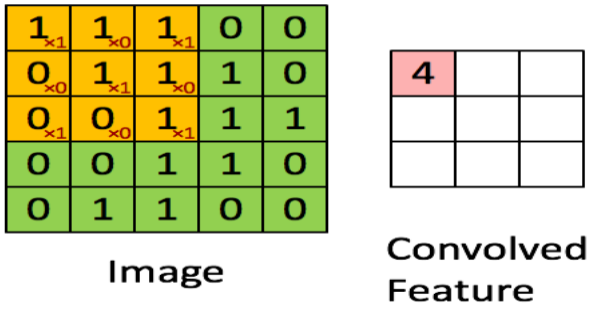
4.1.1.Princip rada konvolucije

Na slici 4, vidimo kernel koji je matrica oblika 3x3, on se pozicionira na vrh ulaznog uzorka. Tada možemo možemo razmišljati o jednodimenzionalnoj konvoluciji kao kliznoj funkciji (1x1 ili 1x2 filter) koja se kreće kroz ulaz , te izvršava odgovarajuće opracije i rezultat se snima na vrh matrice izlazne matrice (1 dimenzionalni niz, umjesto izvorne matrice).



Slika 5.Prikaz operacija za jednu poziciju kernela

Na slici 5. , koristili smo filter dimenzija 3 × 3 (5x5 se također može koristiti, ali bi bio previše složen). Vrijednosti iz filtra pomnožene su elementima izvorne matrice (ulazna slika), a zatim se sabiraju. Da bi dobio potpunu konvolucijsku matricu, algoritam neprestano ponavlja ovaj mali postupak za svaki element pomicanjem filtra preko čitave originalne matrice. Cijeli postupak je prikazan na slici 6. ali pošto nemamo mogućnost da pokrenemo animaciju u wordu zamolio bi vas da pogledate animaciju slike kikom na link.



Slika 6.Animacija prikaza kako kernel vrši interakciju sa matricom koja predstavlja sliku

Kod koji se nalazi u prilogu seminarskog se nalazi kao .ipynb fajl i kao .py fajl. Izvorni kod je napisan u .ipynb fajlu tj uz pomoc Jupyter Notebooka.Jupyter Notebook je web-aplikacija otvorenog koda koja vam omogućuje stvaranje i razmjenu dokumenata koji sadrže jednadžbe, vizualizacije i narativni tekst. Upotrebe uključuju: čišćenje i transformaciju podataka, numeričku simulaciju, statističko modeliranje, vizualizaciju podataka, mašinsko učenje i još mnogo toga.

U Jupyter Notebook-u kod je organizovan u manjim tzv. Ćelijama gdje se svaka može zasebno posmatrati. Kod se može konvertovati iz .ipynb formata u .py format.U nastavku ćemo koristiti kod u .py formatu ali ćemo gledati po ćelijama.

```
1: # %% [code]
2: #Importing
3: import matplotlib.image as mpimg
4: import matplotlib.pyplot as plt
5: import numpy as np  # linear algebra
6: import pandas as pd
7: from fastai.vision import *
8: from PIL import Image
9: from scipy import misc, signal
```

U gore navedenom kodu vršimo import tj unos odgovarajućih biblioteka koje će nam biti cijele ili pojedini dijelovi istih potrebne.

```
10: # %% [code]
11: loc = '/kaggle/input/100-bird-species/test/ANHINGA/2.jpg'
12: im = Image.open(loc)
13: image_gr = im.convert("L")     # convert("L") translate color images into black and white
14:                                # uses the ITU-R 601-2 Luma transform (there are several
15:                                # ways to convert an image to grey scale)
16: print("\n Original type: %r \n\n" % image_gr)
17:
18: # convert image to a matrix with values from 0 to 255 (uint8)
19: arr = np.asarray(image_gr)
20: print("After conversion to numerical representation: \n\n %r" % arr)
```

Nakon što smo učitali odgovarajuće biblioteke, u varijablu loc snimamo putanju gdje se nalaze naš podatak, u ovom primjeru slika, za analizu.Nakon sto smo odabrali sliku 2.jpg u direktoriju ANHINGA, otvoaramo sliku i te informaciju da li smo otovrili sliku snimamo u im varijablu.

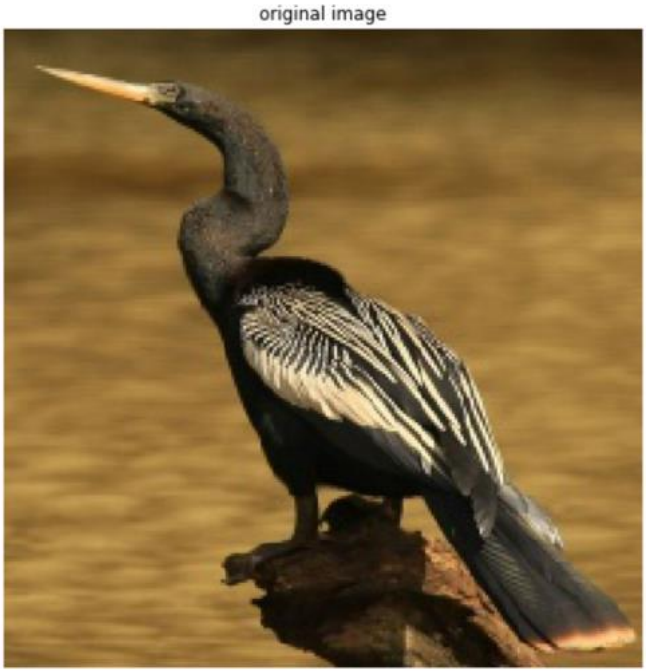
U 13. liniji koda koristimo Luma transgformaciju da konvertujemo sliku u crno/bijeli format. Da je uspjesno konvertovana dobivamo u liniji 16 sto mozemo vidjeti i na slici u prilogu. Te nakon toga vršimo konverziju slike u matricu čije su vrijednosti u intervalu od 0 – 255, čije su vrijednosti prikazane na slici u prilogu.

```
Original type: <PIL.Image.Image image mode=L size=224x224 at 0x7FEEB9A21D68>


After conversion to numerical representation:

array([[ 57,  58,  58, ...,  39,  39,  41],
       [ 63,  64,  64, ...,  39,  39,  40],
       [ 67,  67,  67, ...,  40,  40,  41],
       ...,
       [116, 115, 114, ..., 115, 117, 117],
       [108, 107, 107, ..., 108, 110, 108],
       [101, 102, 102, ..., 105, 109, 108]], dtype=uint8)
```

Slika 7.Ispis slike kao matrice čije su varijable vrijednosti od 0 do 255

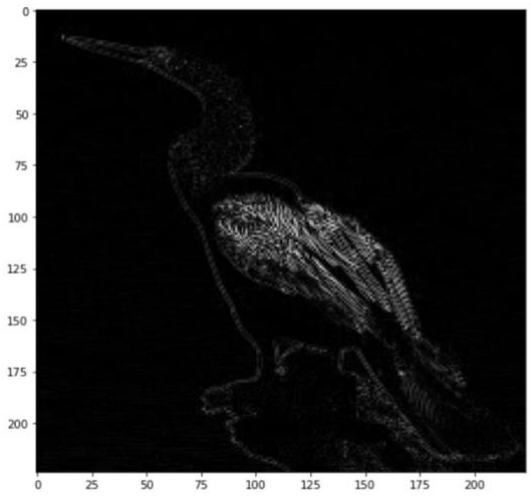
U linijama od 22-32 vršimo plotanje slike pomoču matplotlib.pyplot koju smo učitali na početku kao plt varijablu nad kojom pozivamo funkciju imshow, kojoj prosljeđujemo varijablu img u kojoj se nalazi naša slika. Sliku smo doblili pomoću matplotlib.image koju smo također učitali na početku kao varijablu mpimg, nad kojom smo pozvali imread funkciju koja uzima putanju do naše slike koja je spremljena u varijabli loc. Izlaz tog koda je prikazan na slici 8. u nastavku.

```
22: # %% [code]
23:
24: ### Activating matplotlib for Ipython
25:
26: ### Plot image
27:
28: plt.figure(1,figsize=(12,8))
29: img = mpimg.imread(loc)
30: plt.imshow(img)
31: plt.title('original image');
32: plt.axis('off');
```



Slika 8.Prikaz slike u RGB formatu

```
33: # %% [code]
34: plt.figure(1,figsize=(12,8))
35: imgplot = plt.imshow(arr)
36:  imgplot.set_cmap('gray')    #you  can  experiment  different  colormaps
(Greys,winter,autumn)
37: print("\n Input image converted to gray scale: \n")
38: plt.axis('off')
39: plt.show(imgplot)
```



Slika 10. GRADIENT MAGNITUDE - Mapa značajki

Slika 9. Prikaz učitane slike u gray formatu

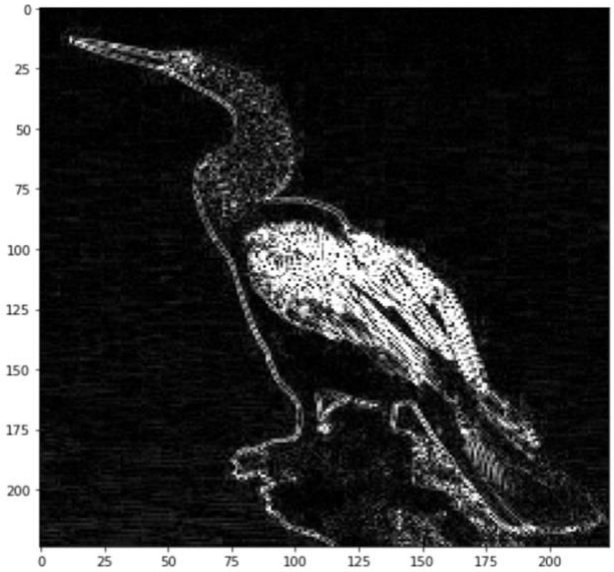
U linijama koda 33-39 vršimo sličan plot kao na slici 8. Međutim.u ovom slučaju smo konvertovali ulaznu sliku u sliku sive boje pomoću funkcije set_cmap(), kojoj smo proslijedili string tipa „gray“ i dobili smo navedenu sliku koja je prikazana na slici 9. Također je moguće proslijediti druge vrste stringova kao što su „winter“ ili „autumn“.

```
40: # %% [markdown]
41: # ### Now lets make a conv kernel
42: # %% [code]
43: kernel = np.array([[ 0, 1, 0],
44:                    [ 1,-4, 1],
45:                    [ 0, 1, 0],])
46:
47: grad = signal.convolve2d(arr, kernel, mode='same', boundary='symm')
48: print('GRADIENT MAGNITUDE - Feature map')
49:
50: fig, aux = plt.subplots(figsize=(10, 8))
51: aux.imshow(np.absolute(grad), cmap='gray');
```

U navedenom kodu iznad smo kreirali kernel kao matricu sa odgovarijućim proizvoljnim vrijednostima.Nakon toga smo nad originalnom slikom uradili konvoluciju sa kreiranim krenelom te pizivajuci funkciju convolve2d() iz objekta signal kojeg smo učitali na početku.Nakon izvršene konvolucije prikazati ćemo izlaznu sliku uz pomoć imgshow () u sivoj boji ma slici 10.Sada već na slici 10. vidimo prepoznavanje nekih od osnovnih ivica slike, odnosno nekih prepoznatljivih dijelova odgovarajuće ptice.

Kada se bavimo većinom stvarnih aplikacija CNN-ova, obično pretvaramo vrijednosti piksela u raspon od 0 do 1.Taj se proces naziva normalizacija. Na ovaj se način nekoliko kernela koristi za otkrivanje različitih osbina, taj proces je prikazan u nastavku, te rezultat tih istih krenela nam na slici 11, prikazuje veci broj osobina date slike.

```
52: # %% [code]
53: type(grad)
54:
55: grad_biases = np.absolute(grad) + 100
56:
57: grad_biases[grad_biases > 255] = 255
58: plt.figure(1,figsize=(18,8))
59: plt.imshow(np.absolute(grad_biases),cmap='gray');
```



Slika 11.Primjena većeg broja krenela na slici

4.2. Upotreba fastai vision

U nastavku ćemo koristiti fastai vision biblioteku da napravimo CNN koji prepoznaje ove ptice i uglavnom koristi isti princip,te je puno jednostavniji za korištenje.Fastai vision biblioteka pojednostavljuje brzo i precizno treniranje neuronskih mreža koristeći najbolje vježbe.Temelji se na istraživanju najboljih najboljih praksi deep learning-a koje se provode na fast.ai, uključujući podršku za modele vida, teksta,tabela i kolaborativne modele.

```
60: # %% [code]
61: from fastai.vision import *
62: import numpy as np # linear algebra
63: import pandas as pd
```

U navedenom kodu pored što smo izvršili import fastai vision biblioteke, također smo importovali i biblioteku za linearnu algebru, te biblioteku pandas za data processing.

```
64: # %% [code]
65: path = Path('/kaggle/input/100-bird-species/consolidated/')
66: tfms = get_transforms(do_flip=True,max_lighting=0.1,max_rotate=0.1)
67: data = (
68:         ImageDataBunch.from_folder
69:             (
70:                 path,train='.', valid_pct=0.15,
71:                 ds_tfms=tfms,size=224,num_workers=4
72:             ).normalize(imagenet_stats)
73:         )
74: # valid size here its 15% of total images,
75: # train = train folder here we use all the folder
76: # from_folder take images from folder and labels them like wise
77: data.show_batch(rows=5)
78: len(data.classes), len(data.train_ds), len(data.valid_ds)
```

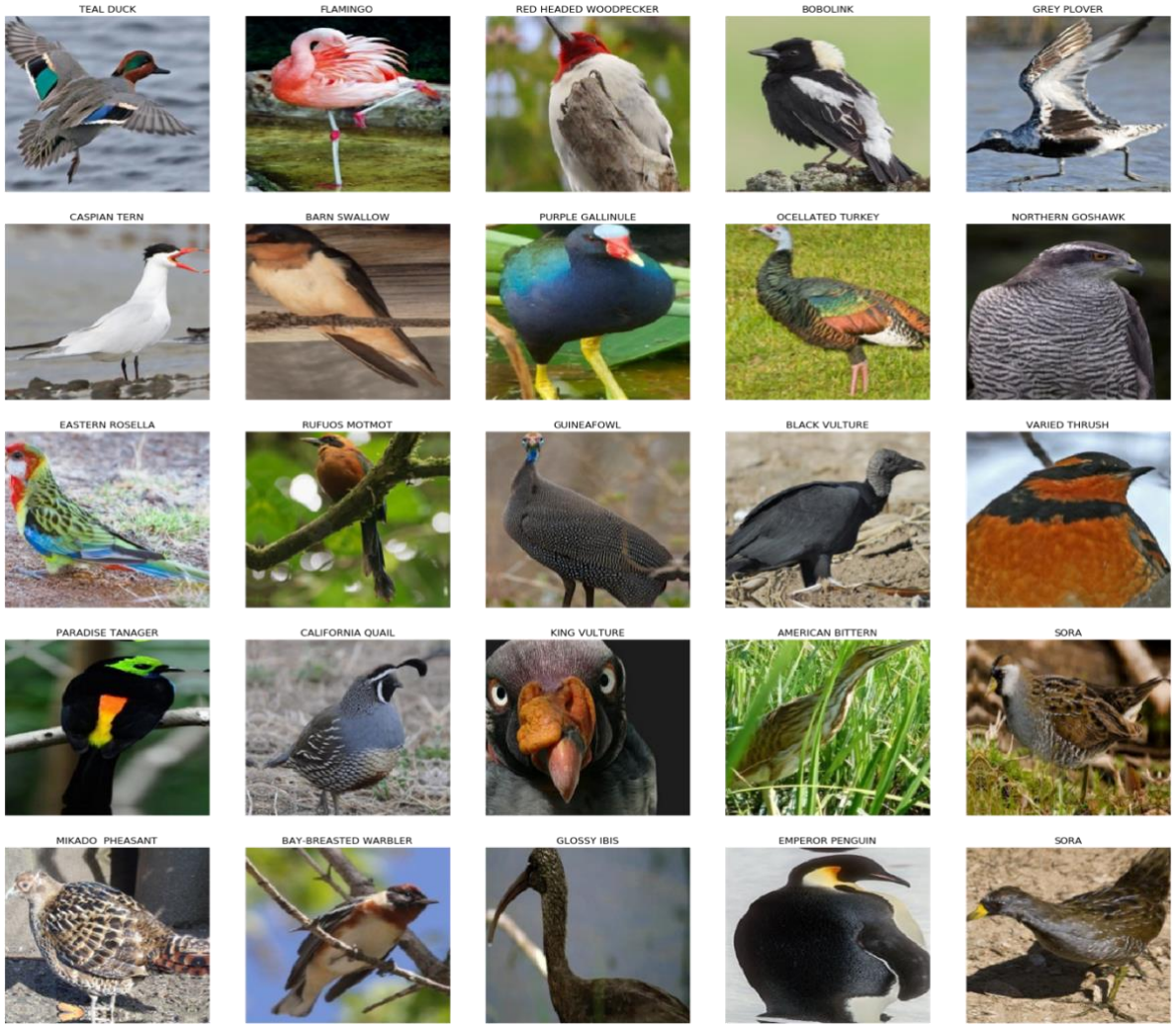
U liniji 65. postavljamo putanju do zeljenog diretorija sa slikama, te istu spremamo u path varijablu, tj formiramo bazu podataka koja se unosi u CNN.Da bi dobili skup trasnformacija sa zadanim vrijednostima koji prilično dobro funkcioniraju u širokom rasponu zadataka, često je najlakše koristiti get_transforms() funkciju, tu funkciju koristimo u liniji 66.Ovisno o prirodi slika u vašim podacima, možda ćete htjeti prilagoditi nekoliko argumenata funkcije, kao npr.

- Do_flip koji ako je postavljen na true slika se nasumično okreće(zadano ponašanje)
- Flip_vert nam omogućava da ograničimo flip na vodotavno okretanje(false) ili na horizontalne i okomite (true) i dr.

Mi smo u ovom primjeru koristili do_flip koji je true, max_lighting je postavljen na 0.1 i max_rotate je postavljen na 0.1.Data funkcija nam vraća hrpu od dvije liste transformacija,jedna je set za treniranje i jedna za skup validacije.Nakon toga sa date putanje u path varijabli, izvršimo treniranje nad cijelim sadržajem sa date putanje, gdje je valid_pct=0.15 i drugi postavljeni parametri u linijama 67-73.From folder uzima sve slike iz te mape te ih označava kao mudre,te nad tim sadržajem izvršimo normalizaciju, te rezultat svih operacija snimio u data.Nakon toga vršimo prikaz data varijable i također imamo i ispis koliko je klasa, koliko je slika za treniranje i slika koje ispunjavaju zadane uslove.

```
Out[19] (175, 21596, 3811)
```

Slika 12. len(data.classes), len(data.train_ds), len(data.valid_ds)

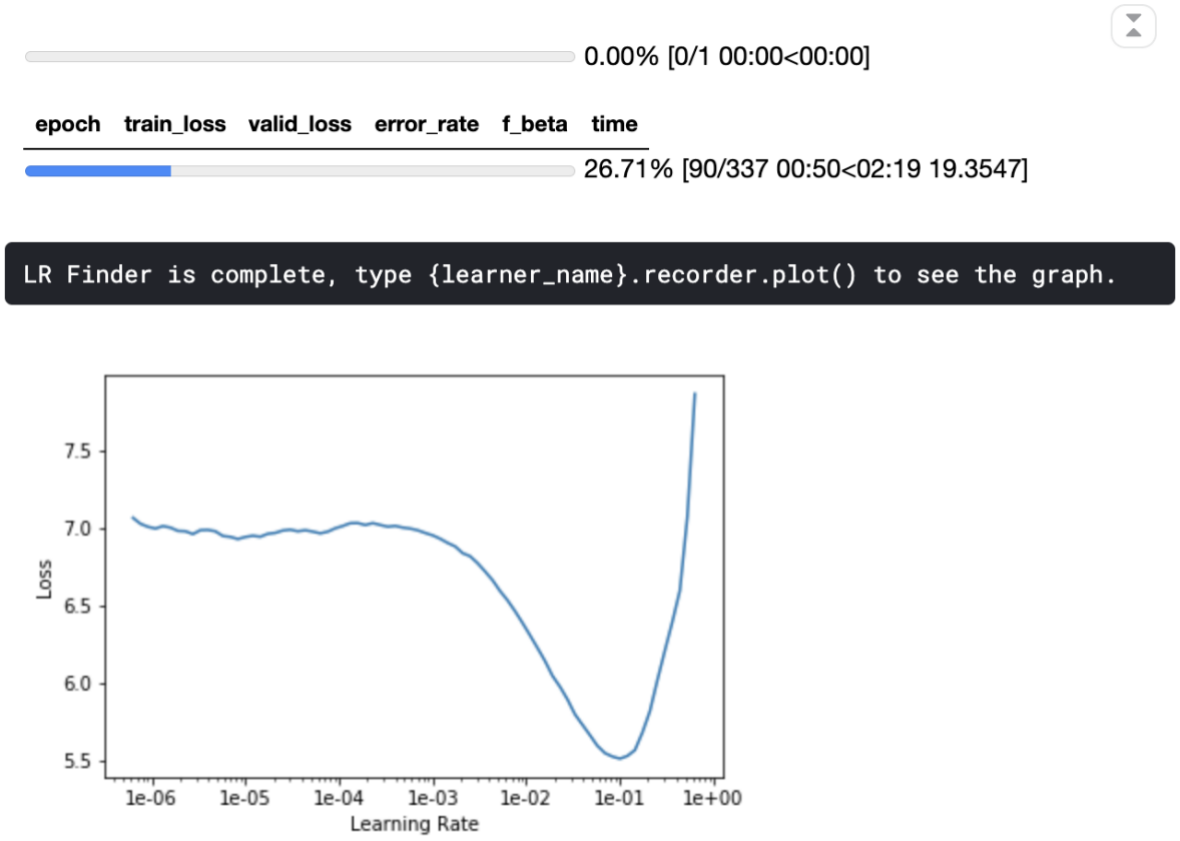


Slika 13. Prikaz 5 redova sadržaja varijable data

```
79: # %% [code]
80: fb = FBeta()
81: fb.average = 'macro'
82: # We are using fbeta macro average in case some
83: # class of birds have less train images
84: learn = cnn_learner(data, models.resnet18,
85:                     metrics=[error_rate,fb],
86:                     model_dir='/kaggle/working/')
```

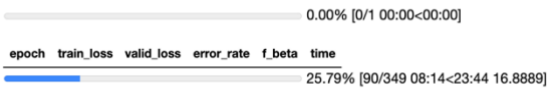
Sada je vrijeme da napravimo metodologiju za učenje.Koristimo fbera „makro“ prosjek u slučaju da neke klase ptica imaju manje slika za učenje.Nakon toga koristimo funkciju cnn_learner(), koja nam pomaže da automatski dobijemo pretraženi model iz date arhitekture s prilagođenom osnovom koja je prikladna za naše podatke.Od modela koristitili smo resnet18 model s 18 slojeva,jer su naučne osobine često prenosive na različite podatke. Npr, model obučen na velikom skupu slika ptica sadržavat će naučene uzorke poput ivica ili vodoravnih linija koje bi mogle biti prenosive na naše podatke

```
87: # %% [code]
88: learn.lr_find()
89: learn.recorder.plot()
```



Slika 14.Određivanje dobre stope učenja

Kada pokrenemo lr_find() funkciju ona nam pomaže da odaberemo odgovarajuću stopu učenja.Na osnovu rezultata koje smo dobili vidimo da stopa učenja koja odgovara minimaloj vrijendosti već je malo previsoka, jer smo na rubu poboljšanja i apsolutne greške. Postupak je izvršen na GPU i potrebno je vrijeme bilo vidimo iz priloženog skoro pa minuta.Ako ponovimo ovaj dio na TPU tada vrijeme za određivanje dobre stope učenja je 8:50 minuta, sto vidimo na slici 15.



Slika 15.Određivanje dobre stope učenja na TPU

```
90: # %% [code]
91: lr = 1e-2 # learning rate
92: learn.fit_one_cycle(5,lr,moms=(0.8,0.7))  # moms
```

epoch	train_loss	valid_loss	error_rate	f_beta	time
0	1.244587	1.059303	0.254264	0.730642	03:08
1	1.018038	0.667701	0.180005	0.810592	03:06
2	0.665768	0.399841	0.108371	0.886332	03:10
3	0.347666	0.257343	0.069273	0.928241	03:08
4	0.247927	0.230623	0.062976	0.933637	03:04

Slika 16.Statiskika po epohama

Nakon sto postavimo stopu učenja na 1e-2 u liniji 91, koju smo odabrali sa diagrama na slici 14, pokrećemo funkciju fit_one_cycle() koja pokreće treniranje upotrebom jednog ciklusa kako bi nam pomogao da brže obučimo naš model. Na slici 16 vidimo da nakon sto je funkcija zavrsila svoju obavezu da za samo 5 epoha imamo tačnost od 93% kao i makro prosjek F1_score oko 0.93.Treniranje je izvršeno na GPU od kaggle stranice i vidimo da je za svako epohu odprilike trebalo do tri minute za izvršenje, znači oko 15 min ukupno, dok je na TPU za ovakvu bazu slika potrebno 30 min po epohi, znaci 2,5 h ukupno za 5 epoha.

```
93: # %% [code]
94: interp = ClassificationInterpretation.from_learner(learn)
```

ClassificationInterpretation predstavlja interpretaciju metoda za klasifikacione metode. Nad njom smo pozvali funkciju from_learner() koji nam daje samo bolji način da kreiramo instancu naše kalase, te tu vrijednost snimamo u varijablu interp.

Slika 17.Prikaz slika sa njihovim predviđanjima,gubitkom i vjerovatnoćom stvarne klase

```
96: # %% [code]
97: interp.plot_top_losses(12,figsize=(20,8))
```

Na slici 17. smo imali prikaz top_losses zajedno sa njihovim predviđanjima, stvarnom klasom, gubitkom i vjerovatnoćom date klase, te prikaz istih smo dobili pomoću funkcije plot_top_losses().

```
98: # %% [code]
99: interp.most_confused(min_val=3)
```

```
Out[47] [('CINNAMON TEAL', 'TEAL DUCK', 3),
         ('MASKED BOOBY', 'ALBATROSS', 3),
         ('NORTHERN GOSHAWK', 'PEREGRINE FALCON', 3)]
```

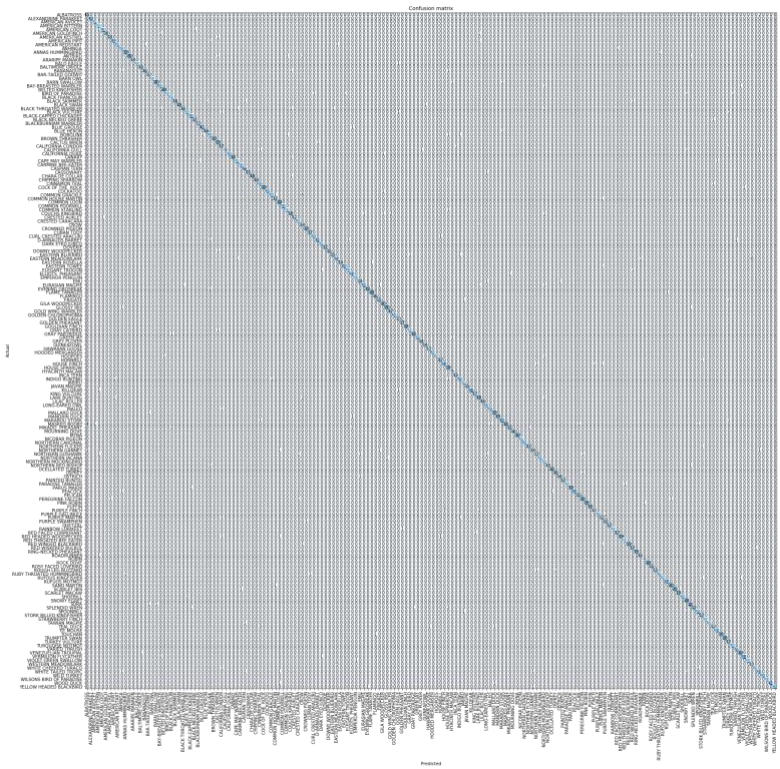
Slika 18.Lista nedijagonalnih entiteta od matrice kofuzije

U liniji 99 pozivamo funkciju most_confused() sa parametrom min_val = 3. Funkcija nam vraća sortiranu silaznu listu nedijagonalnih entitera od matrice konfuzije prikazanih kao stvarni, predivđeni i broj pojava. Sadržaj je prikazan na slici 18. Također matrica konfuzije je prikazana na slici 19, a za prikaz veće slike istu imate u prilogu seminarskog.

Da bi dobili ovu matricu potrebno je dodati sljedeću liniju koda poslije 99. linije :

```
interp.plot_confusion_matrix(figsize=(25,50))
```

Nakon ovoe linije dobiti ćemo navedenu matricu prikazana na slici 19.



Slika 19.Matrica konfuzije

U nastavku se nalaze neka predviđanja, tj provjeravamo da li naš klasifikator funkcioniše kako treba.

4.3. Predviđanja

4.3.1. American Avocet

```
100: # %% [code]
101: file = '/kaggle/input/100-bird-species/predictor test set/013.jpg'
102: img = open_image(file)
103: # open the image using open_image func from fast.ai
104: print(learn.predict(img)[0])
105: # lets make some prediction
106: img
```

```
AMERICAN AVOCET
```

Out[58]:

Slika 20.Uspješno određena vrsta ptice - American Avocet

4.3.2. Anhinga

```
100: # %% [code]
101: file = '/kaggle/input/100-bird-species/predictor test set/049.jpg'
102: img = open_image(file)
103: # open the image using open_image func from fast.ai
104: print(learn.predict(img)[0])
105: # lets make some prediction
106: img
```

```
ANHINGA
```

Out[101]:

Slika 21.Uspješno određena vrsta ptice - Anhinga

4.3.3.Bananaquit

```
100: # %% [code]
101: file = '/kaggle/input/100-bird-species/predictor test set/075.jpg'
102: img = open_image(file)
103: # open the image using open_image func from fast.ai
104: print(learn.predict(img)[0])
105: # lets make some prediction
106: img
```

```
BANANAQUIT
```

Out[102]:

Slika 22.Uspješno određena vrsta ptice – Bananaquit

Na osnovu navedena tri primjera možemo primijetiti da naš klasifikator radi poprilično dobro, tj. Vidimo iz priloženog da na osnovu nekih random slika on može u 93% slučajeva da odredi vrstu ptice.

NAPOMENA - Ako bi kojim slučajem zaboravili ukucati linije 91 i 92 u svom kodu, tada bi dolazilo do grešaka prilikom određivanja vrste ptice.Također je potrebno provjeriti prilikom odabira putanje do odgovarajućeg direkrorija provjeriti da li je putanja ispravna.Prilikom izrade seminarskog su korištene putanje koje su odgovarale diretorijima na Kaggle stranici direrktno, jer je projekat rađen pomoću Kaggle online inteface-a.

5. Prikaz svih vrsta ptica u bazi

U nastavku će biti priložena jedna kraća skirpta, koja će nam poslužiti za prikaz svih vrsta koje se trenutno nalaze u bazi podataka koju mozemo preuzeti sa Kaggle stranice sa linka koji je naveden ranije. Prvo vidimo da u narednom kodu ponovo uključujemo tj importujemo odgovarajuće biblioteke koje su nam potrebne prilikom izrade slika. Nakon toga postavljamo putanju do direktorija u kojem se nalaze slike,te nakon toga prolazimo korz sve direktorije i vršimo prikaz odgovarajuće vrste ptice sa njenim nazivom iznad slike.Izlaz ovog malog koda je prikazan na slici 23.

```
1: # %%
2: from IPython import get_ipython
3: import os
4: import matplotlib.pyplot as plt
5: get_ipython().run_line_magic('matplotlib', 'inline')
6: import matplotlib.image as mpimg
7: # %% [code]
8: dir=r'/kaggle/input/100-bird-species'
9: dir_list=os.listdir(dir)
10: print (dir_list)
11: # %% [code]
12: test_dir=r'/kaggle/input/100-bird-species/180/test'
13: classes=len(os.listdir(test_dir))
14: fig = plt.figure(figsize=(20,100))
15: if classes % 5==0:
16:     rows=int(classes/5)
17: else:
18:     rows=int(classes/5) +1
19: for row in range(rows):
20:     for column in range(5):
21:         i= row * 5 + column
22:         if i>classes:
23:             break
24:         specie=species_list[i]
25:         species_path=os.path.join(test_dir, specie)
26:         f_path=os.path.join(species_path, '1.jpg')
27:         img = mpimg.imread(f_path)
28:         a = fig.add_subplot(rows, 10, i+1)
29:         imgplot=plt.imshow(img)
30:         a.axis("off")
31:         a.set_title(specie)
```

Slika 23.Prikaz svih vrsta ptica koje se nalaze trenutno u bazi podataka

6. Literatura

[1] https://hr.wikipedia.org/wiki/Ptice
[2] https://www.kaggle.com/gpiosenka/100-bird-species?rvi=1
[3] https://www.kaggle.com/ianmoone0617/bird-classifier-simple-fastai-tutorial
[4] https://outdooreducationafrica.com/different-bird-species/
[5] https://docs.fast.ai/index.html
[6] https://docs.fast.ai/train.html
[7] https://www.kaggle.com/gpiosenka/explore-bird-data