

I – SUNS: ZADANIE 03

Meno: Róbert Šumšala

ID: 111 464

ÚVOD

Náplň zadania:

Náplňou zadania bolo implementovať program, v ľubovoľnom jazyku, ktorý bude klasifikovať zviera na obrázku.

Programovací jazyk a knižnice:

Ako programovací jazyk sme vybrali Python, a využili knižnice keras, tensorflow, pandas, numpy, matplotlib, PIL, seaborn, sklearn, math.

1. časť: Načítanie a analýza dát

Načítanie dát:

Obrázky zvierat boli rozdelené do priečinkov podľa kategórií. Každá kategória zodpovedala názvu zvieraťa, ktorého obrázky sa v priečinku nachádzali. Trénovacie a testovacie dáta boli v separátnych priečinkoch. Preto sme dáta sme načítali nasledovným spôsobom:

- Trénovacie (z ktorých sme jednu desatinu použili ako validačné dáta):

```
train_data_path = './data/train'

train_dataset = tf.keras.utils.image_dataset_from_directory(
    train_data_path,
    shuffle=True,
    validation_split=0.1,
    subset="training",
    seed=123,
    image_size=(img_height, img_width),
    batch_size=batch_size,
```

- Validačné (jedna desatina z trénovacích dát):

```
valid_dataset = tf.keras.utils.image_dataset_from_directory(
    train_data_path,
    shuffle=True,
    validation_split=0.1,
    subset="validation",
    seed=123,
    image_size=(img_height, img_width),
    batch_size=batch_size,
```

- Testovacie:

```
test_dataset = tf.keras.utils.image_dataset_from_directory(
    test_data_path,
    shuffle=False, # No need to shuffle the test dataset
    image_size=(img_height, img_width),
    batch_size=batch_size,
```

Veľkosť všetkých obrázkov sme nastavili na 224x224, a pracovali sme so všetkým troma farebnými zložkami, takže shape našich dát bol (224,224,3). Hodnotu shuffle sme pri testovacej dali na false, keďže nemá význam aby boli obrázky pri testovaní miešané.

Analýza dát:

Ako prvé sme si dali vypísať početnosti v jednotlivých triedach pre trénovacie aj testovacie dátá, ako aj celkový počet tried.

```
Found 4860 files belonging to 90 classes.
Using 4374 files for training.
Found 4860 files belonging to 90 classes.
Using 486 files for validation.
Found 540 files belonging to 90 classes.
```

Počet obrázkov pre trénovacie, validačné a testovacie dátá

```
Class: hippopotamus, Number of Pictures: 54
Class: sparrow, Number of Pictures: 54
Class: gorilla, Number of Pictures: 54
Class: cat, Number of Pictures: 54
Class: rhinoceros, Number of Pictures: 54
Class: wombat, Number of Pictures: 54
Class: seahorse, Number of Pictures: 54
Class: butterfly, Number of Pictures: 54
Class: donkey, Number of Pictures: 54
```

*Počet obrázkov v jednotlivých kategóriach –
trénovacie (aj validačné) dátá*

```
Class: hippopotamus, Number of Pictures: 6
Class: sparrow, Number of Pictures: 6
Class: gorilla, Number of Pictures: 6
Class: cat, Number of Pictures: 6
Class: rhinoceros, Number of Pictures: 6
Class: wombat, Number of Pictures: 6
Class: seahorse, Number of Pictures: 6
Class: butterfly, Number of Pictures: 6
Class: donkey, Number of Pictures: 6
```

*Počet obrázkov v jednotlivých kategóriach –
testovacie dátá*

Na základe daných výpisov sme vedeli zhodnotiť, že pri trénovacích dátach pracujeme s 4374 obrázkami, pri validačných s 486 obrázkami a pri testovacích dátach s 540 obrázkami. Celkovo obsahuje model 90 rôznych zvierat, teda kategórií, pričom každá kategória pri testovacích, tým pádom aj validačných (kedže boli vytvorené z testovacích) dátach obsahuje 54 vzoriek, resp. obrázkov. A každá trieda pri testovacích dátach obsahuje 6 obrázkov.

Ďalej sme si zobrazili jedného reprezentanta z každej triedy.



Reprezentanti jednotlivých tried

Následne sme použili model ResNet50, natrénovaný na probléme ImageNet, bez toho aby sme ho menili alebo trénovali, na zobrazenie predikcií obrázkov z našich testovacích dát.

```
model_resnet50 = tf.keras.applications.resnet50.ResNet50(include_top=True,  
weights='./ImageNetWeights/official_resnet50_weights.h5',  
input_tensor=None,  
input_shape=(224, 224, 3),  
pooling=None,  
classes=1000,  
  
classifier_activation='softmax'  
)
```

Vypísané dáta obsahovali najlepšiu predikciu modelu – názov kategórie a skóre, druh zvieratá v našom datasete a koľko zo šiestich obrázkov z danej kategórie určil správne.

```
Top prediction byt ImageNet: badger -> 1.00
Animal name in the dataset: badger
Correctly predicted 6/6
Top prediction byt ImageNet: zebra -> 1.00
Animal name in the dataset: zebra
Correctly predicted 6/6
Top prediction byt ImageNet: wombat -> 0.98
Animal name in the dataset: wombat
Correctly predicted 6/6
Top prediction byt ImageNet: orangutan -> 0.96
Animal name in the dataset: orangutan
Correctly predicted 6/6
Top prediction byt ImageNet: tiger -> 0.91
Animal name in the dataset: tiger
Correctly predicted 6/6
Top prediction byt ImageNet: chimpanzee -> 0.89
Animal name in the dataset: chimpanzee
Correctly predicted 5/6
```

```
Top prediction byt ImageNet: timber_wolf -> 0.45
Animal name in the dataset: wolf
Correctly predicted 0/6
Top prediction byt ImageNet: black_swan -> 1.00
Animal name in the dataset: swan
Correctly predicted 0/6
Top prediction byt ImageNet: great_white_shark -> 0.99
Animal name in the dataset: shark
Correctly predicted 0/6
```

```
Top prediction byt ImageNet: macaw -> 0.53
Animal name in the dataset: woodpecker
Correctly predicted 0/6
Top prediction byt ImageNet: prairie_chicken -> 0.46
Animal name in the dataset: turkey
Correctly predicted 0/6
Top prediction byt ImageNet: koala -> 1.00
Animal name in the dataset: bear
Correctly predicted 0/6
```

Niekteré z najpresnejších určení

Niekteré z vysvetliteľne nepresných určení

```
Top prediction byt ImageNet: albatross -> 0.85
Animal name in the dataset: dolphin
Correctly predicted 0/6
Top prediction byt ImageNet: centipede -> 0.32
Animal name in the dataset: seahorse
Correctly predicted 0/6
Top prediction byt ImageNet: fountain -> 0.92
Animal name in the dataset: ox
Correctly predicted 4/6
Top prediction byt ImageNet: triceratops -> 0.65
Animal name in the dataset: octopus
Correctly predicted 0/6
```

Niekteré z nepresných určení

Mnohé obrázky boli kategorizované správne, najúspešnejší bol napríklad jazvec s najlepšou predikciou 1,00 a taktiež bolo správne kategorizovaných 6/6 obrázkov. Niektoré obrázky boli kategorizované správne, ale názov kategórie bol odlišný, resp. bol uvedený konkrétny druh daného zvieraťa, napr. kategorizácia žraloka bola veľmi úspešná, ale názov kategórie bol žralok biely. A niektoré obrázky neboli správne zadelené vôbec, napríklad chobotnica bola zaradená ako triceratops, albatros ako delfín a podobne. Zaujímavá bola napríklad kategorizácia vola. Ten bol súčasťou 4-krát zo šiestich určení správne, ale najvyššie skóre dostalo priradenie obrázku do kategórie fontány, čo sme si nevedeli vysvetliť.

2. časť: Naša konvolučná siet'

Príprava dát:

- Dáta sme rozdelili na trénovaciu, validačnú a testovaciu množinu pomocou generátorov dát (viď 1. časť – *Načítanie dát*).
- Pre zlepšenie učenia sme dáta augmentovali – náhodný zoom, náhodný horizontálny flip, náhodná malá rotácia

```
augmentation = tf.keras.Sequential(  
    [  
        layers.RandomFlip("horizontal", input_shape=(img_height,  
                                                    img_width, 3)),  
        layers.RandomZoom(0.2),  
        layers.RandomRotation(0.2),  
    ]  
)
```

- Dáta sme normalizovali (v prvej vrstve nášho modelu, viď úryvok kódu v časti *Trénovanie modelu*). Normalizácia v tomto prípade spočíva vo vydelení hodnoty každého pixelu hodnotou 255, aby každý pixel nadobúdal hodnotu z intervalu <0,1>, namiesto štandardných hodnôt pre 8-bitové obrázky (hodnoty z intervalu <0,255>).
- Pri generátore trénovacích a validačných dát sme nastavili parameter shuffle na True. Aby sme dosiahli náhodnosť prichádzajúcich obrázkov, čo zníži šancu, že sa náš model naučí príznaky obrázkov na základe ich poradia. Pri generátore testovacích dát sme nastavili shuffle na False, keďže v tomto prípade nemá význam premiešavať prichádzajúce obrázky. (viď 1. časť – *Načítanie dát*)

Trénovanie modelu:

Náš model, tensorflow.keras.Sequential, sme zostavili nasledovane (popisujeme náš najúspešnejší model). Ako prvé sme do neho pridali nami zadefinovanú augmentáciu (viď časť *Príprava dát*) a normalizáciu dát spolu so zadefinovaním tvaru vstupných dát, v našom prípade (128, 128, 3). Následne sme vytvorili hlavnú časť s tromi 2D konvolučnými vrstvami, ktoré mali 32, 64 a 64 filtrov, každá mala aktivačnú funkciu relu a prvé dve mali nastavenú L2 regularizáciu. Vybrali sme L2, pretože tá zabráni aby sa model stal príliš komplexný, zatiaľ, čo L1 je vhodnejšia ak predpokladáme, že niektoré príznaky sú zbytočné alebo sa zbytočne opakujú. Po každej 2D konvolučnej vrstve sme pridali MaxPooling vrstvu. Ďalej sme pridali dropout vrstvu, ako ďalšiu formu regularizácie, nasledovanú flatten vrstvou, ktorá slúži na konverziu 2D polí na jeden vektor. Na záver sme pridali jednu dense vrstvu obsahujúcu 128 neurónov a s aktivačnou funkciou relu, a ešte jednu dense vrstvu, predstavujúcu výstupnú vrstvu, s 90 neurónmi. (hodnota určená počtom tried zvierat nášho datasetu)

```
model = tf.keras.Sequential(  
    [  
        augmentation,  
        layers.Rescaling(1. / 255, input_shape=(img_height, img_width, 3)),  
        layers.Conv2D(32, 3, activation='relu',  
kernel_regularizer=regularizers.l2(0.001)),  
        layers.MaxPooling2D(),  
        layers.Conv2D(64, 3, activation='relu',  
kernel_regularizer=regularizers.l2(0.001)),  
        layers.MaxPooling2D(),  
        layers.Conv2D(64, 3, activation='relu'),  
        layers.MaxPooling2D(),  
        layers.Dropout(0.4),
```

```

        layers.Flatten(),
        layers.Dense(128, activation='relu'),
        layers.Dense(90, name="outputs")
    )
)

```

Následne sme model skomplilovali, pričom sme ako optimizer zvolili Adam, a hodnotu loss sme nastavali na SparseCategoricalEntropy.

```

model.compile(optimizer=optimizer,
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

```

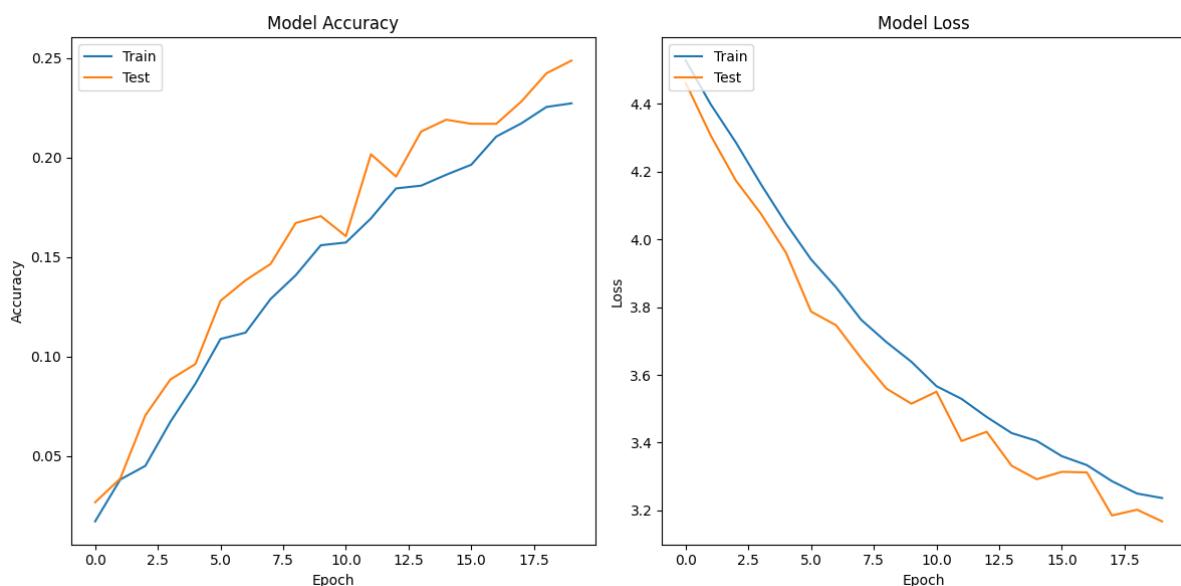
Uskutočnili sme päť rôznych trénovaní, s rôznymi nastaveniami hyperparametrov a počtom vrstiev a podobne. Jednotlivé trénovania a ich úspešnosti sú zaznamenané v nasledujúcej tabuľke.

MODEL	EPOCHS	LEARNING RATE	TEST ACCURACY	TRAIN ACCURACY
Rovnaký ako popísaný vyššie	10	0,001	18,33%	20,69%
Rovnaký ako popísaný vyššie	20	0,001	20,74%	31,82
Pridaná jedna 2DConv vrstva so 128 neurónmi bez regularizácie, tretej vrstve bola pridaná L2 regularizácia a dropout rate bol znížený na 0,35	15	0,003	11,11%	13,19%
Rovnaký ako popísaný vyššie	20	0,0001	20,00%	24,87%
Rovnaký ako popísaný vyššie	25	0,0001	19,63%	24,19%

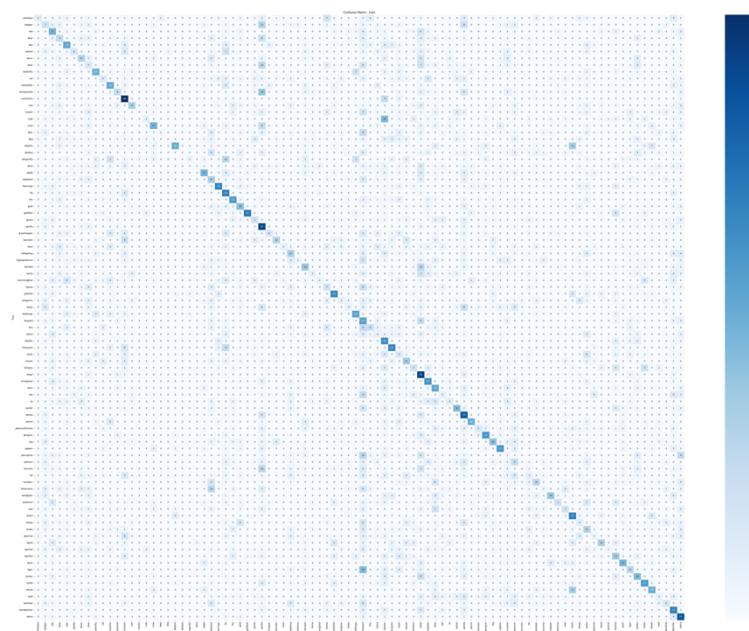
Tabuľka s dátami z jednotlivých trénovaní

Vyhodnotenie najlepšieho modelu:

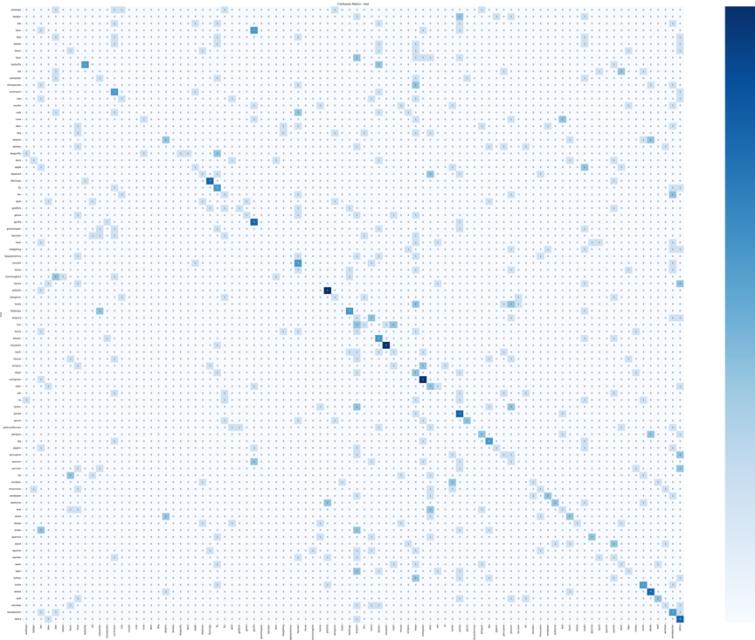
Na základe dát získaných z viacerých trénovaní sme dospeli k záveru, že najlepší bol model zo štvrtého trénovania. (zvýraznený v tabuľke, presnejšie popísaný na začiatku v časti *Trénovanie modelu*) Tento model dosiahol súčasne niečo menšiu testovaciu presnosť ako model z druhého trénovania, ale rozdiel medzi trénovacou a testovacou presnosťou mal menší, čo znamená že bol menej pretrénovaný (ovplyvnené hodnotou learning rate).



Priebeh trénovania – (vľavo zmena presnosti,
vpravo zmena straty)



Konfúzna matica – trénovacia množina



Konfúzna matica – testovacia množina

Na grafoch priebehu trénovalia nebadáme žiadne známky pretrénovania, ani podtrénovania. A na oboch konfúznych maticiach je viditeľná diagonála. Na základe získaných dát sme označili tento model za úspešný/postačujúci pre toto zadanie.

3. časť: Využitie transfer learningu na zlepšenie výsledkov

Extrakcia príznakov:

Na extrakciu príznakov sme využili základný model VGG16, natrénovalený na probléme ImageNet. Model sme nastavili tak, aby neslúžil ako klasifikátor do jednotlivých tried, ale aby boli jeho výstupy príznaky pre jednotlivé príznaky. Dosiahli sme to nastavením parametra `include_top` na False.

```
base_model = tf.keras.applications.vgg16.VGG16(include_top=False,
                                                weights='imagenet')
```

Takto sme získali príznaky pre každý obrázok z priečinku trénovacích dát. Všetky získané príznaky sme spolu s adresou obrázku a názvom kategórie zvieratá na obrázku uložili do dataframu. Každý riadok v dataframe predstavoval jeden obrázok a každý stĺpec predstavoval jeden príznak. Keďže VGG16 vráti 512 príznakov, náš dataframe mal spolu so stĺpcami adresy a názvu zvieratá 514 stĺpcov.

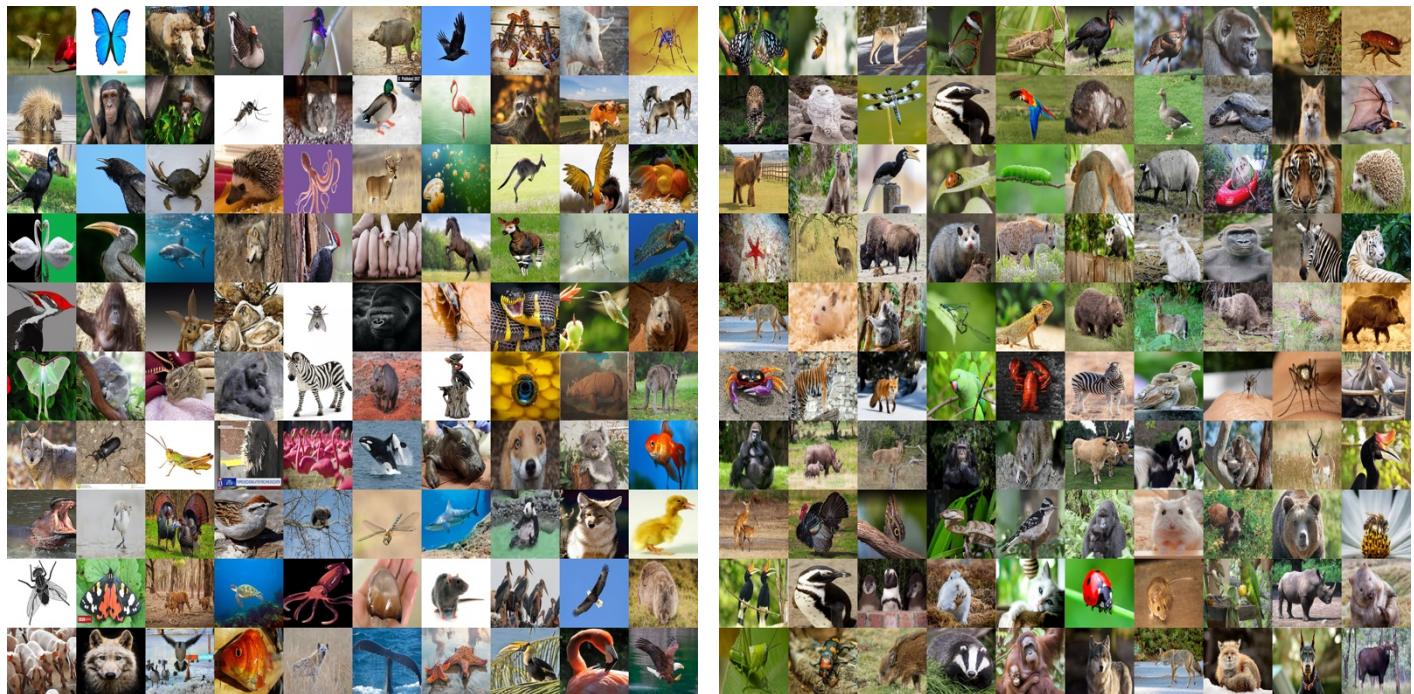
```
img = tf.keras.preprocessing.image.load_img(image_path,
                                             target_size=(img_height, img_width))
x = tf.keras.preprocessing.image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = tf.keras.applications.vgg16.preprocess_input(x)

features = base_model.predict(x)
features = features.flatten()
```

Vytvorenie a analýza zhlukov:

Následne sme z daných príznakov vytvorili zhluky pomocou vybraného zhlukovacieho algoritmu. Vyskúšali sme Kmeans, AgglomerativeClustering aj DBSCAN (všetky z knižnice sklearn). Najlepšie a najjasnejšie rozdelenie, také že sme videli nejaké súvislosti medzi vybranými obrázkami, zabezpečil algoritmus AgglomerativeClustering.

Taktiež sme vyskúšali rôzne počty zhlukov a najlepšie výsledky sme dosiahli s rozdelením do 3 zhlukov. Ak ich bolo viac, začínaťalo byť rozdelenie nejasné.



Zhluk 1

Zhluk 2



Zhluk 3

100 náhodných obrázkov z každého zhluku

The image consists of three side-by-side panels of abstract animal art. Each panel features a large, semi-transparent animal figure over a background of soft, blended colors. Panel 1 on the left shows a large bird of prey, possibly an eagle or hawk, with its wings spread, set against a background of green and blue. Panel 2 in the center shows a dark-colored dog, possibly a Doberman Pinscher, standing in a forest setting with brown and green tones. Panel 3 on the right shows a white animal, likely a deer or moose, with its head turned back, set against a background of blue and yellow.

Priemerné obrázky jednotlivých zhľukov

Analýzou vyššie uvedených podmnožín so 100 náhodne vybratými obrázkami pre každý zhluk, sme odhadli typy obrázkov a súvislosti medzi nimi nasledovne:

- Zhluk 1: Obrázky s pestrým, kontrastným farebným obsahom
 - Zhluk 2: Obrázky s prevažujúcou zelenou farbou, poprípade menej farebne výrazne obrázky
 - Zhluk 2: Obrázky vtákov.

Tieto pravidlá sa nedajú určiť so sto percentnou presnosťou, ale pre väčšinu obrázkov sa dajú považovať za pravdivé. Tieto naše odhady potvrdzujú aj priemerné obrázky. Môžeme vidieť, že priemerný obrázok prvého zhluku je farebne najpestrejší a najsýtejší. Priemerný obrázok pre druhý zhluk je farebne menej výrazný a prevažujú zelené a sivo-hnedé odtiene a na treťom priemernom obrázku vieme rozoznať niekoľko obrázkov vtákov.

Dotrénovanie konvolučnej siete:

Dotrénovanie konvolučnej siete spočíva vo vybraní základného modelu, v našom prípade model trénovaný na probléme ImageNet, a v nahradení jeho top vrstiev nášimi, ktoré dotrénujeme. Náš postup vyzeral nasledovne:

- Prednáčíťali sme si dát aby sme urýchliili trénovanie

```
AUTOTUNE = tf.data.AUTOTUNE  
train_dataset_pref = train_dataset.prefetch(buffer_size=AUTOTUNE)  
valid_dataset_pref = valid_dataset.prefetch(buffer_size=AUTOTUNE)  
test_dataset_pref = test_dataset.prefetch(buffer_size=AUTOTUNE)
```

- Pripravili sme si augmentáciu dát

```
data_augmentation = tf.keras.Sequential([
    tf.keras.layers.RandomFlip('horizontal'),
    tf.keras.layers.RandomRotation(0.2),
])
```

- Pripravili/upravili sme dátá tak aby boli kompatibilné s naším vybraným základným modelom, MobileNetV2

```
preprocess_input = tf.keras.applications.mobilenet_v2.preprocess_input
```

- Vytvorili sme základný model, MobileNetV2 (tvar vstupných dát bol 128, 128, 3), parameter include_top sme nastavali na False, aby neboli vrchné vrstvy zahrnuté

- Základný model sme nastavili tak aby sa netrénoval

```
base_model.trainable = False
```

- Pripravili sme vrstvy, ktoré budú prirobené k základnému modelu a budú trénované

```
inputs = tf.keras.Input(shape=(img_height, img_width, 3))
x = data_augmentation(inputs)
x = preprocess_input(x)
x = base_model(x, training=False)
x = global_average_layer(x)
x = layers.Dropout(0.2)(x)
outputs = prediction_layer(x)
```

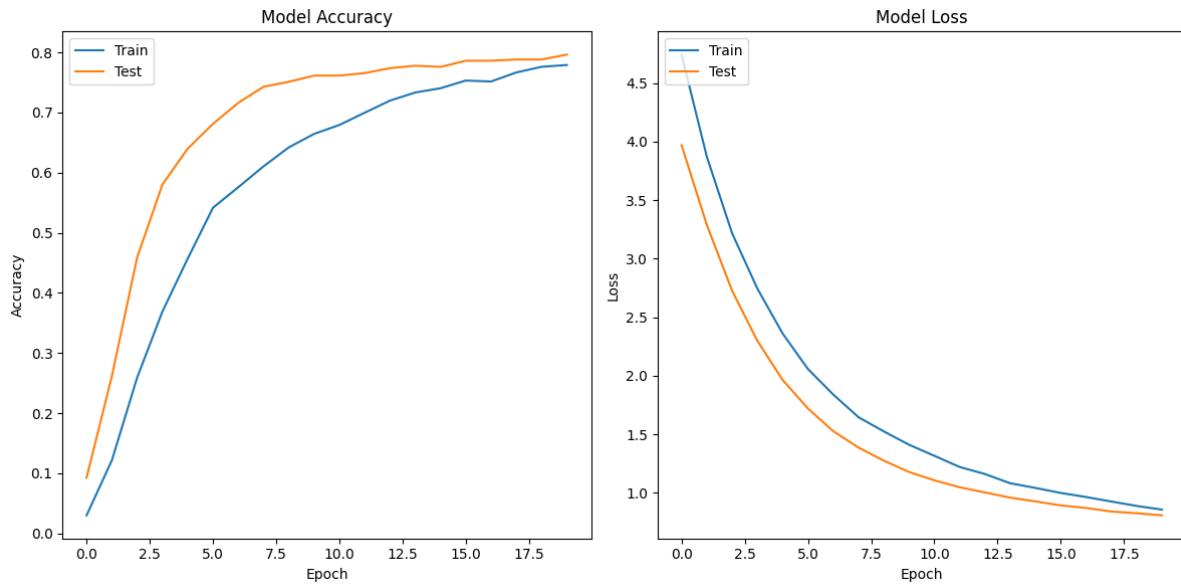
- Skombinovali sme základný model s našimi vrstvami

```
model = tf.keras.Model(inputs, outputs)
```

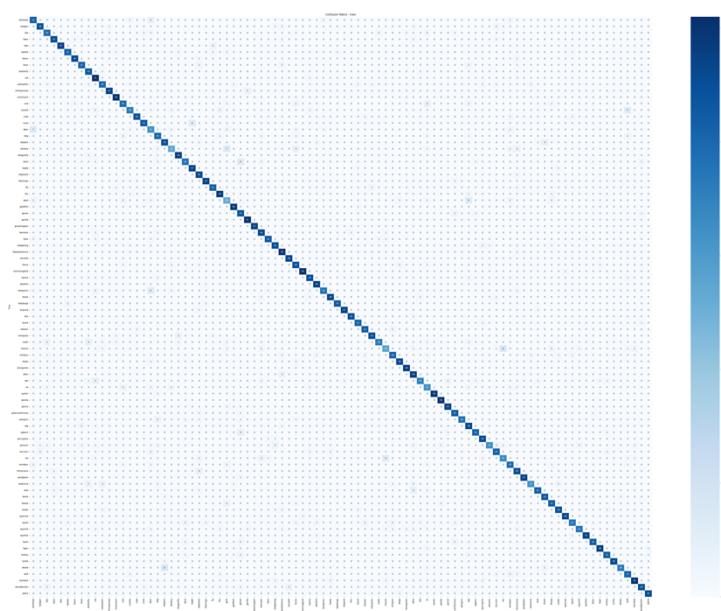
Finálnu verziu modelu sme následne skomplikovali. Za optimizer sme zvolili Adam s mierou učenie 0,0001. Hyperparameter loss sme nastavili na SparseCategorical-Crossentropy. Následne sme model natrénovali v priebehu 20 epoch.

Vyhodnotenie finálneho modelu:

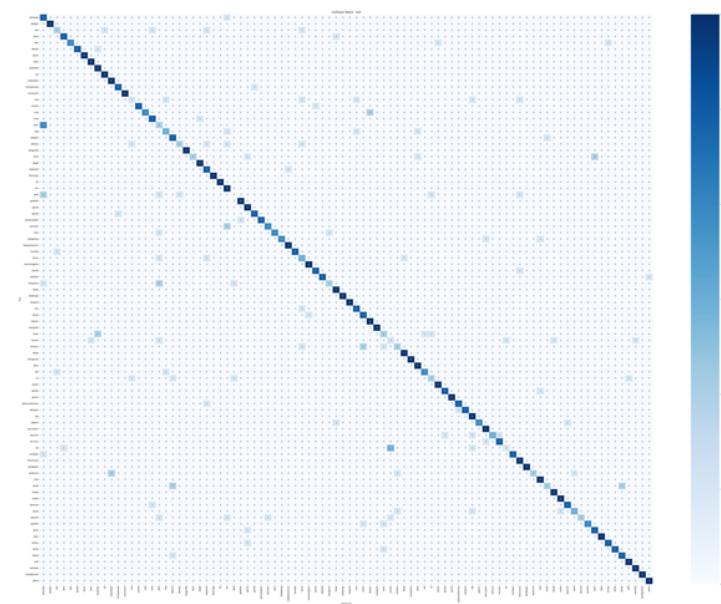
Náš finálny model dosiahol lepšie výsledky ako naša prvá konvolučná sieť. Model dosiahol úspešnosť 88% na trénovacej, a na testovacej 80,15%. To že tento model dosiahol dobré výsledky dokazujú aj nižšie uvedené grafy.



Priebeh trénovania finálneho modelu –
(vľavo zmena presnosti, vpravo zmena straty)



*Konfúzna matica finálneho modelu – trénovacia
množina*



*Konfúzna matica finálneho modelu – testovacia
množina*

BONUS 1: Predikovanie na vlastných obrázkoch

Správnu funkcionality nášho modelu sme overili na obrázkoch, ktoré neboli súčasťou testovacieho, ani trénovacieho datasetu. Jednalo sa o 17 obrázkov mačky, ktoré neboli stiahnuté z internetu.

Vytvorili sme priečinok, ktorý obsahoval niekoľko priečinkov zvierat z testovacieho datasetu, vrátane priečinka *cat*, a taktiež dva priečinky s našimi obrázkami mačky. (obrázky sme rozdelili do dvoch priečinkov, aby sme zachovali 6 obrázkov pre priečinok, tak ako to bolo pri testovacom datasete) Tieto priečinky sme označili *macka* a *macka_02*.



Priečinky s našimi obrázkami mačky

Najprv sme zistili pravú triedu mačky z testovacieho datasetu, tou bola trieda 9. Túto istú triedu sme pridelili ako pravú aj naším dvom priečinkom s obrázkami mačky. Následne sme nechali urobiť model predikcie na obrázky z vyššie spomínaného priečinku.

Na základe výpisov, na ktorých sme videli, že väčšinu z našich obrázkov určil ako mačku, sme označili náš model za úspešne natrénovaný, fungujúci v reálnom svete.

Príklad výpisu predikcií pri kategórii bear

Predikcie pri kategórií

```
macka
True label: 9
Predicted class: 9
True label: 9
Predicted class: 9
True label: 9
Predicted class: 81
True label: 9
Predicted class: 9
True label: 9
Predicted class: 9
True label: 9
Predicted class: 9
macka_02
True label: 9
Predicted class: 9
```

Predikcie pri našich fotkách mačky

ZÁVER

Pri tomto zadaní sme sa oboznámili s konvolučnými sieťami. Ako fungujú, ako s nimi pracovať, ako vytvárať naše vlastné. Tak ako aj s princípmi transfer learningu.

Výsledkom nášho zadania je konvolučná sieť v jazyku Python, ktorá dokáže určiť zviera na obrázku.

ZDROJE

Tak ako sú uvedené v priloženom zdrojovom kóde:

```
# RESOURCES
    # Image generator - Stackoverflow, seminar10

    # Display pictures as collage - ChatGPT

    # Help with using pretrained ImageNet -
https://learnopencv.com/image-classification-pretrained-imagenet-models-tensorflow-keras/

    # Help with our CNN - tensorflow documentation

    # Confusion matrix - seminar11

    # Features extraction, Transfer learning/Fine-tuning - keras
        documentation
```