Slovenská technická univerzita v Bratislave

Fakulta elektrotechniky a informatiky

Strojové videnie a výpočtová inteligencia

SVVI

Zadanie 2.

2026 Viktor Barczi

Zadanie

Našou úlohou je vytvoriť konvolučnú neurónovú sieť na rozpoznávanie 4 a viac kategórii obrázkov. Testovanie mame vykonávať na viacerých druhov sieti a následne vyhodnotiť ich úspešnosť.

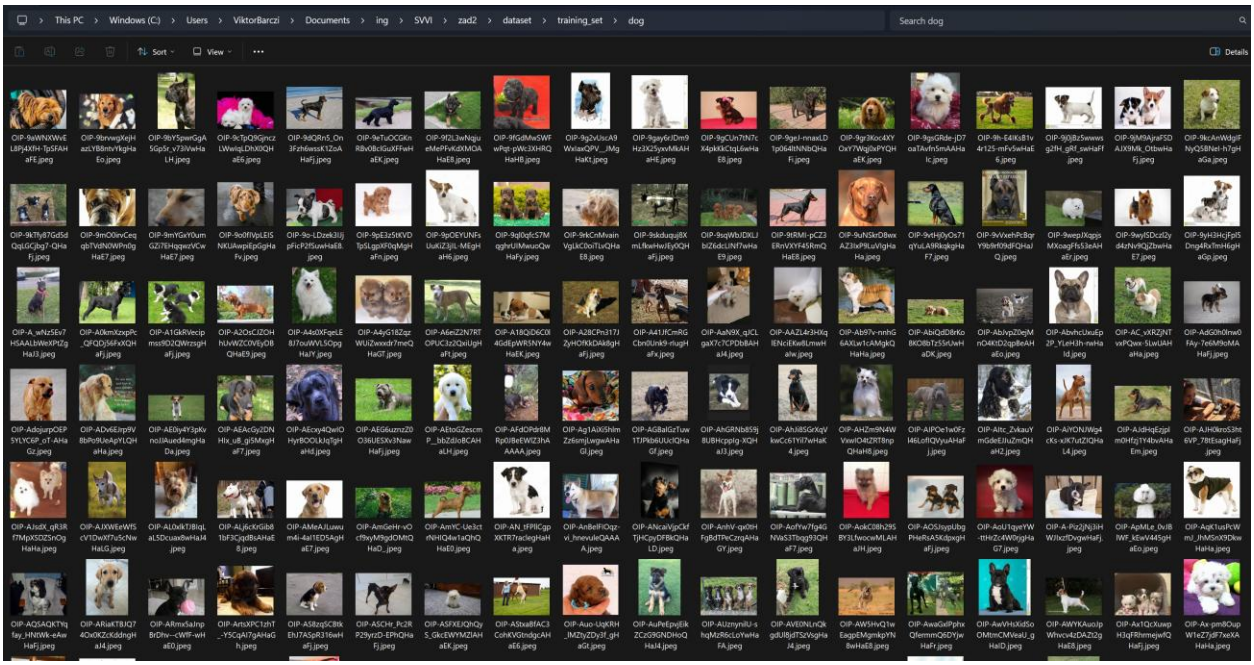
Riešenie

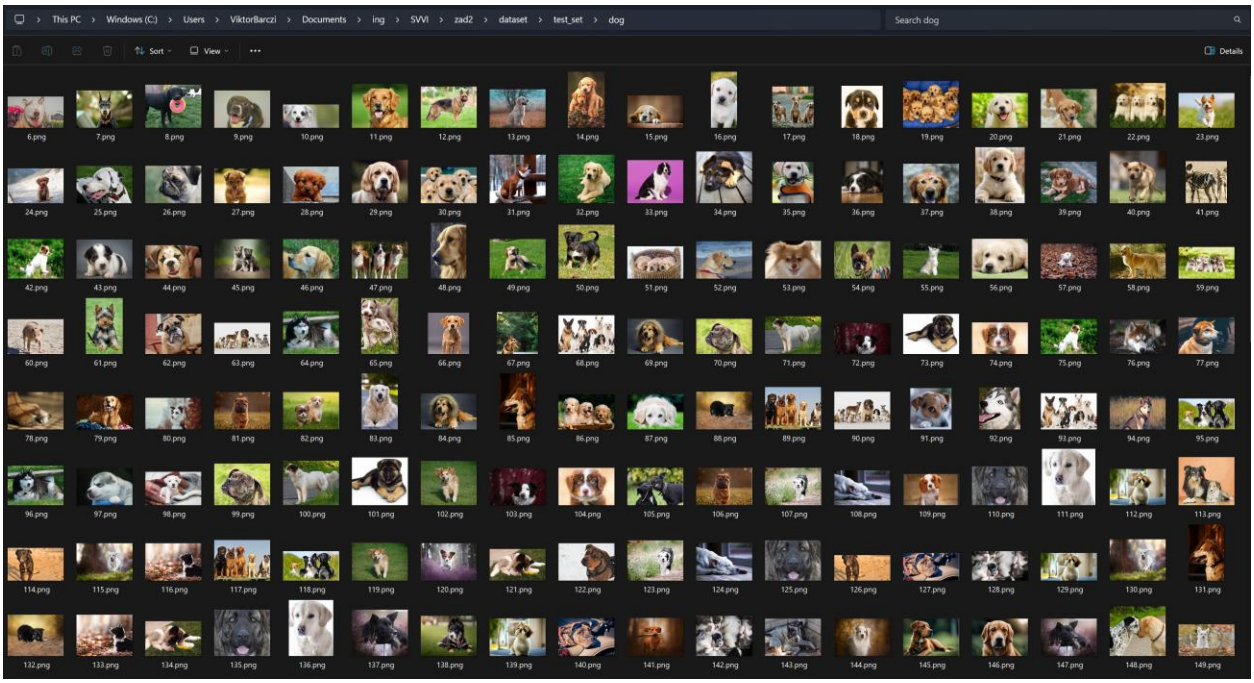
Dataset

Dataset ktorý som používal na tréningovanie je dostupný na tomto odkaze: [Animals-10]

Dataset ktorý som používal na testovanie je dostupný na tomto odkaze: https://www.scidb.cn/en/detail?dataSetId=e2ebd46cb1304a82bab54a8873cb3004

Trénovanie bolo na 4 kategóriách (kôň, baran, slon, pes) pričom každá obsahuje 1200 obrázkov. Pri testovaní každá kategória obsahuje 350 obrázkov. Názornú ukážku obrázkov môžeme vidieť tuto:





Na preprocessovanie obrázkov využijeme **tf.keras.preprocessing.ImageDataGenerator**. augmentáciu. Obrázky zmenšíme na veľkosť 64×64, zastavíme shuffle a nastavíme správne class_mode.

```
img_size = (64, 64)
batch_size = 32

EXPECTED_CLASSES = ["dog", "elephant", "sheep", "horse"]

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

training_set = train_datagen.flow_from_directory(
    'zad2/dataset/training_set',
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False
)

test_set = test_datagen.flow_from_directory(
    'zad2/dataset/test_set',
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False
)
```

Found 4800 images belonging to 4 classes. -> Trénovací dataset

Found 1400 images belonging to 4 classes. -> Testovací dataset

Trénovanie CNN

Konvolučná neurónová sieť bola navrhnutá v sekvenčnej architektúre a pozostáva z dvoch konvolučných blokov typu Conv2D + MaxPooling2D, ktoré slúžia na extrakciu obrazových príznakov a redukciu rozmerov dát. Následne je výstup z konvolučnej časti zploštený vrstvou Flatten a spracovaný plne prepojenou vrstvou Dense so 128 neurónmi. Výstupná vrstva so Softmax aktiváciou určuje pravdepodobnostné rozdelenie medzi jednotlivými triedami.

```
cnn = Sequential()

cnn.add(Conv2D(filters=32, kernel_size=3, activation='relu', input_shape=(64, 64, 3)))
cnn.add(MaxPooling2D(pool_size=2, strides=2))

cnn.add(Conv2D(filters=32, kernel_size=3, activation='relu'))
cnn.add(MaxPooling2D(pool_size=2, strides=2))

cnn.add(Flatten())
cnn.add(Dense(units=128, activation='relu'))

cnn.add(Dense(units=num_classes, activation='softmax'))

cnn.compile(
    optimizer='adam',
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

cnn.summary()
```

```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 62, 62, 32)	896
max_pooling2d (MaxPooling2D)	(None, 31, 31, 32)	0
conv2d_1 (Conv2D)	(None, 29, 29, 32)	9,248
max_pooling2d_1 (MaxPooling2D)	(None, 14, 14, 32)	0
flatten (Flatten)	(None, 6272)	0
dense (Dense)	(None, 128)	802,944
dense_1 (Dense)	(None, 2)	258

```
 Total params: 813,346 (3.10 MB)
 Trainable params: 813,346 (3.10 MB)
 Non-trainable params: 0 (0.00 B)
Model úspešne načítaný.
```

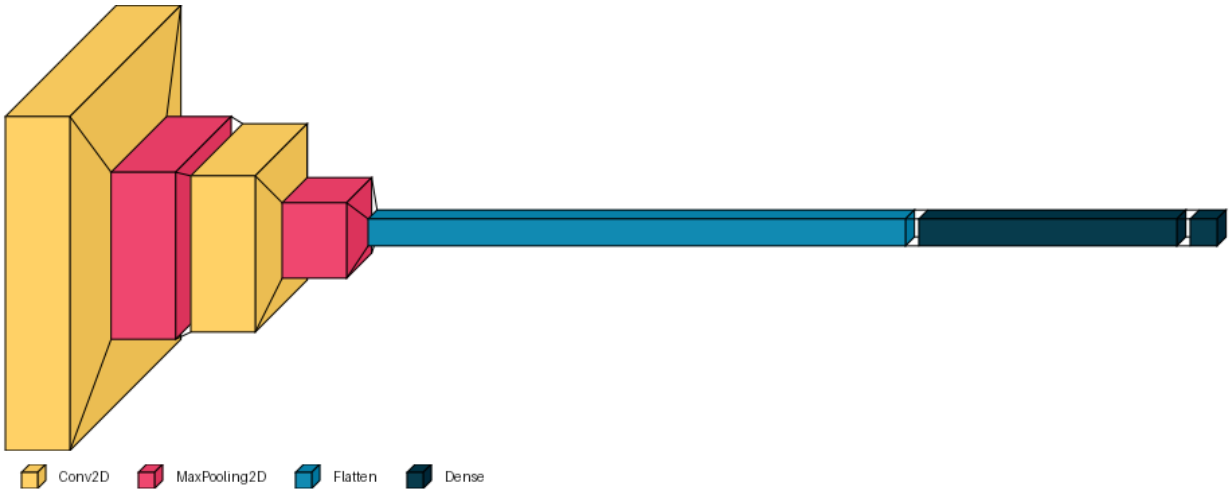
```
Počet tried: 4
Indexy tried: {'dog': 0, 'elephant': 1, 'horse': 2, 'sheep': 3}
Nájdené triedy: ['dog', 'elephant', 'horse', 'sheep']
index_to_class: {0: 'dog', 1: 'elephant', 2: 'horse', 3: 'sheep'}
```

Kód na výpis:

```
print(test_set.class_indices)

predictions = cnn.predict(test_set)
print(predictions)

predictions = np.argmax(predictions, axis=1)
print(predictions)
```

```
{'dog': 0, 'elephant': 1, 'horse': 2, 'sheep': 3}
44/44 ━━━━━━━━━━━━━━━━━━━━ 5s 123ms/step
[[7.0087439e-01 9.1309696e-02 7.8686751e-02 1.2912914e-01]
 [1.6764868e-02 1.6310789e-03 2.6954912e-02 9.5464927e-01]
 [8.2819492e-01 2.8191390e-04 1.7136461e-01 1.5861091e-04]
 ...
 [3.3486715e-01 4.8823154e-01 2.9842248e-02 1.4705910e-01]
 [4.1936934e-01 2.6340050e-01 2.5696088e-02 2.9153401e-01]
 [4.8424699e-03 5.1243092e-05 2.8437890e-02 9.6666837e-01]]
[0 3 0 ... 1 0 3]
```

Obrázok vizualizácie siete

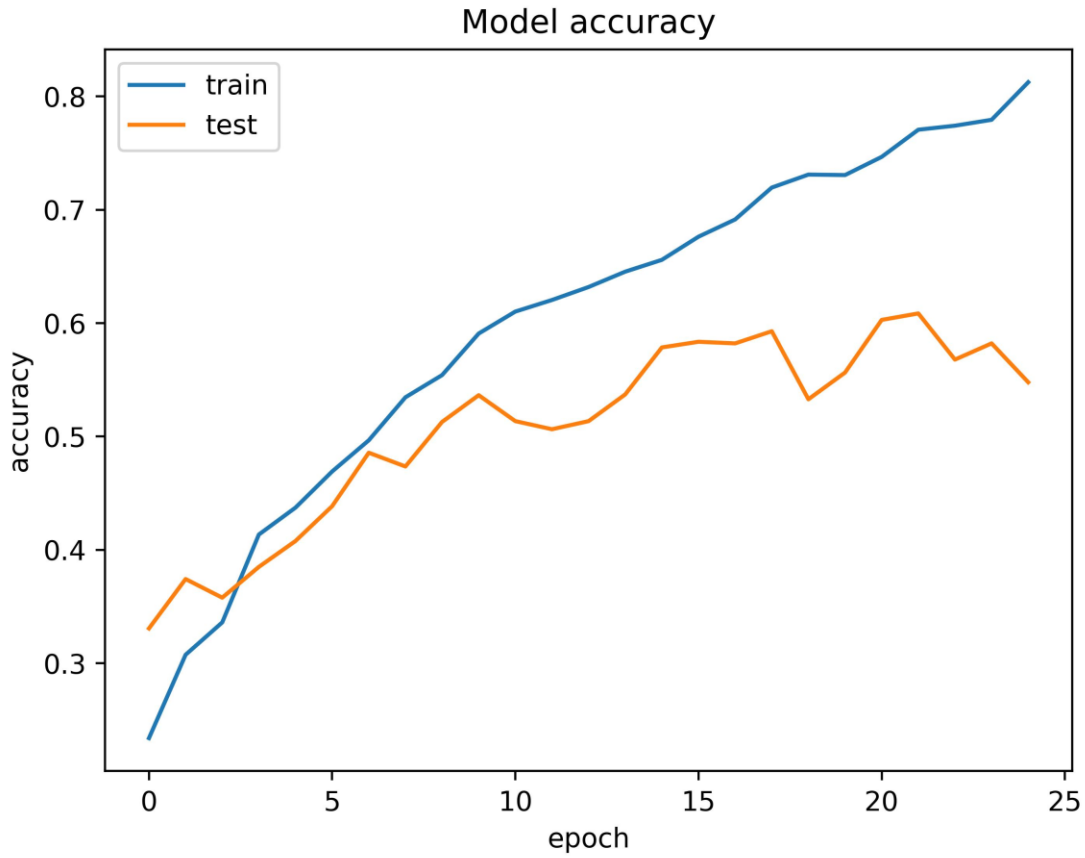



```python
# ---------------------------------------------------------
# FIX pre visualkeras – dopočítanie output_shape pre vrstvy
# ---------------------------------------------------------

# uistíme sa, že model je postavený
cnn.build((None, 64, 64, 3))

input_shape = cnn.input_shape
for layer in cnn.layers:
    # ak vrstva nemá output_shape, skúsime ho dopočítať
    if not hasattr(layer, "output_shape") or layer.output_shape is None:
        try:
            out_shape = layer.compute_output_shape(input_shape)
            layer.output_shape = out_shape
            input_shape = out_shape
        except Exception as e:
            print(f"Nemôžem vypočítať output_shape pre vrstvu {layer.name}: {e}")


# =========================================================
# Vizualizácia architektúry
# =========================================================

try:
    visualkeras.layered_view(
        cnn,
        to_file='cnn_architecture.png',
        legend=True
    )
    print("Architecture saved to cnn_architecture.png")
except Exception as e:
    print("Visualkeras warning:", e)
```

Graf priebehu trénovania

Na vizualizáciu u priebehu trénovania použijeme:

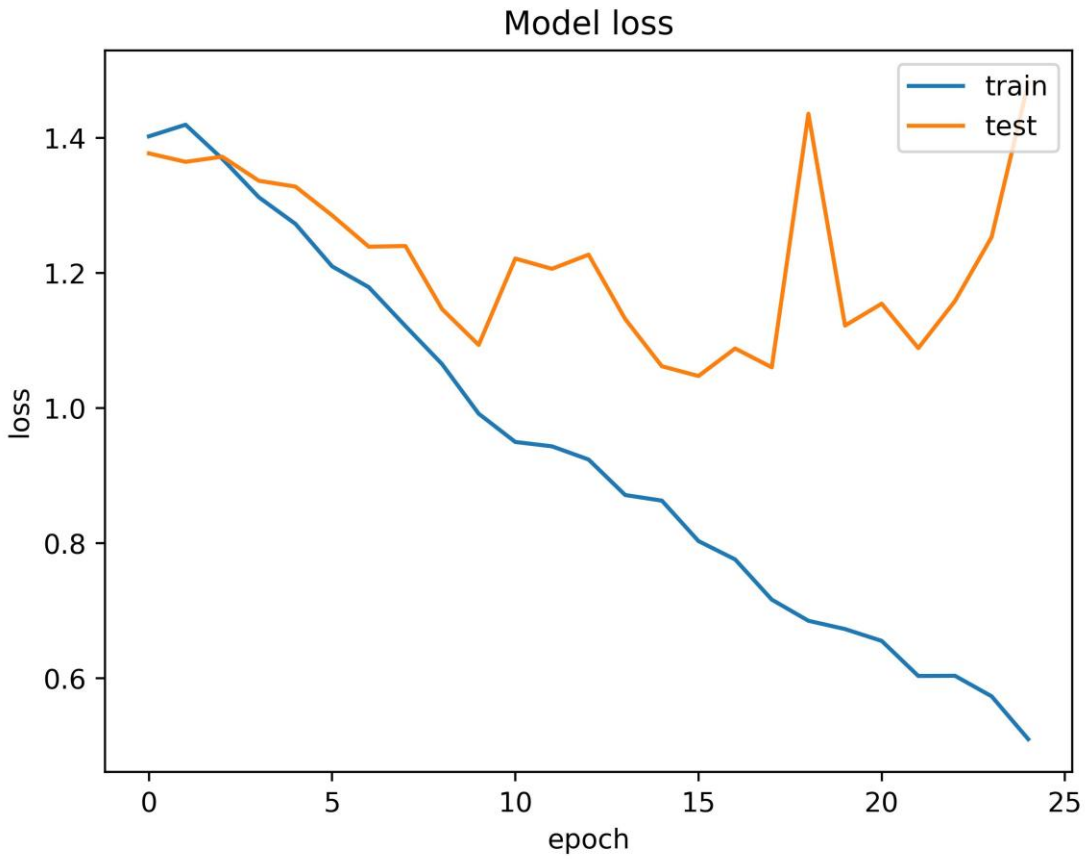
```
plt.figure()
plt.plot(hist.history['accuracy'])
plt.plot(hist.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.savefig('Model_accuracy.jpg', dpi=500)
plt.show()
```



Vizualizácia loss

Na vizualizáciu u loss funkcie použijeme:

```
plt.figure()
plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.title('Model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper right')
plt.savefig('Model_loss.jpg', dpi=500)
plt.show()
```



Testovanie a výpis predikcie

Na generovanie predikcie a predikčných obrázkov bola použitá nasledujúca funkcia:

```python
def predict_image(model, img_name, image_path, img_size, index_to_class,
                  save_dir="zad2/dataset/predict"):

    file_name = os.path.basename(image_path)
    print(file_name)

    # načítanie a príprava obrázka pre model
    img = load_img(image_path, target_size=img_size)
    arr = img_to_array(img)
    arr = np.expand_dims(arr, axis=0) / 255.0

    # predikcia
    result = model.predict(arr)
    print(result)
    print(training_set.class_indices)

    pred_idx = int(np.argmax(result, axis=1)[0])
    pred_class_name = index_to_class[pred_idx]
    print(f"{pred_class_name}")

    # uloženie obrázka s TEXTOM
    os.makedirs(save_dir, exist_ok=True)
    save_path = os.path.join(save_dir, f"{img_name}.jpg")

    img_cv = cv2.imread(image_path)
    if img_cv is None:
        print(f"CHYBA: cv2.imread() nevedelo načítať '{image_path}'")
    else:
        text = f"{pred_class_name}"
        cv2.putText(
            img_cv,
            text,
            (10, 35),                       # pozícia textu
            cv2.FONT_HERSHEY_SIMPLEX,
            1,
            (0, 255, 0),                    # zelený text
            2,
            cv2.LINE_AA
        )

        ok = cv2.imwrite(save_path, img_cv)
        if not ok:
            print(f"CHYBA: nepodarilo sa uložiť '{save_path}'")
        else:
            print(save_path)

    return pred_class_name
```

Na testovanie sme použili 4 obrázky (jpg alebo png), funkcia predict_image by mala úspešne uhádnuť triedu a zapísať triedu zvieraťa do obrázka (dog, elephant, sheep, horse).

```
single_images = {
    "dog":      "zad2/dataset/single_prediction/dog.png",
    "elephant": "zad2/dataset/single_prediction/elephant.png",
    "sheep":    "zad2/dataset/single_prediction/sheep.jpg",
    "horse":    "zad2/dataset/single_prediction/horse.jpg"
}

for class_name, path in single_images.items():
    pred = predict_image(
        model=cnn,
        img_name=class_name,
        image_path=path,
        img_size=img_size,
        index_to_class=index_to_class
    )

    print(f"Skutočná trieda: {class_name}, predikovaná trieda: {pred}")
```

Vidíme že sieť kvalifikoval všetky obrázky úspešne:


horse


elephant


dog

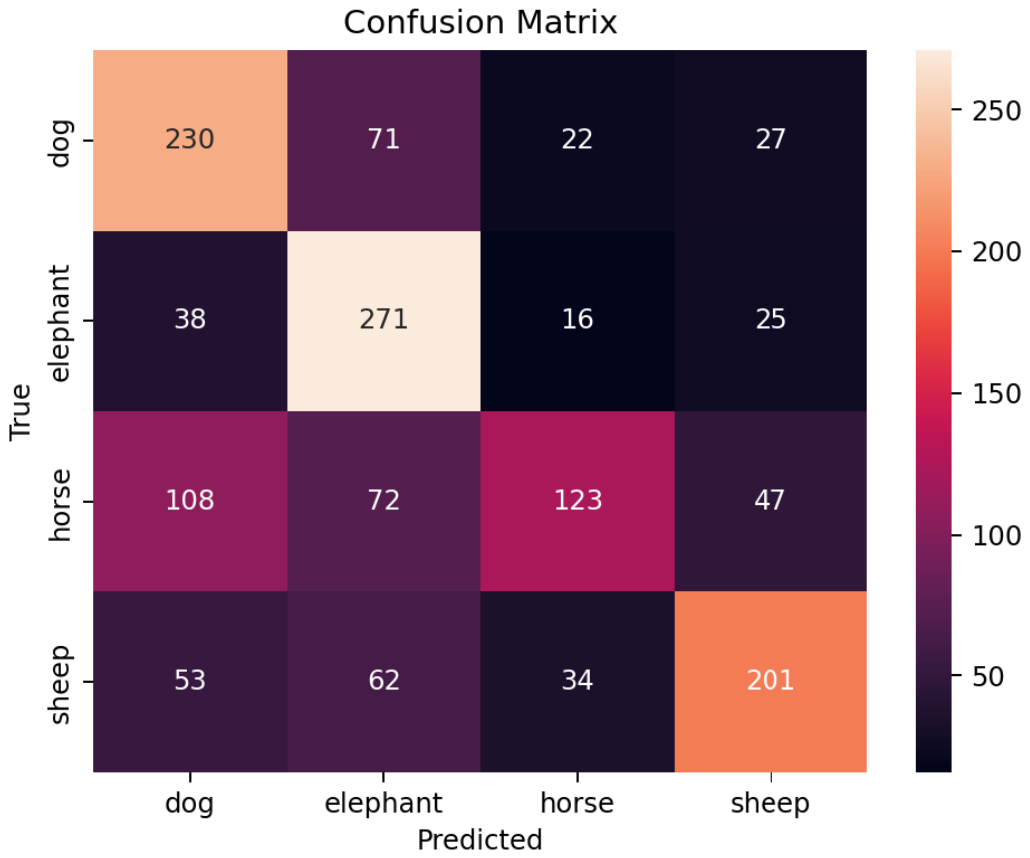
Výsledky v textovej podobe:

```
dog.png
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step
[[9.0496778e-01 7.8299339e-04 8.0743901e-02 1.3505343e-02]]
{'dog': 0, 'elephant': 1, 'horse': 2, 'sheep': 3}
dog
zad2/dataset/predict\dog.jpg
Skutočná trieda: dog, predikovaná trieda: dog
elephant.png
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
[[0.04226469 0.67919606 0.2349177  0.04362158]]
{'dog': 0, 'elephant': 1, 'horse': 2, 'sheep': 3}
elephant
zad2/dataset/predict\elephant.jpg
Skutočná trieda: elephant, predikovaná trieda: elephant
sheep.jpg
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
[[0.12410765 0.01342384 0.00592488 0.8565436 ]]
{'dog': 0, 'elephant': 1, 'horse': 2, 'sheep': 3}
sheep
zad2/dataset/predict\sheep.jpg
Skutočná trieda: sheep, predikovaná trieda: sheep
horse.jpg
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
[[1.4646464e-03 1.5385379e-04 9.9821413e-01 1.6740264e-04]]
{'dog': 0, 'elephant': 1, 'horse': 2, 'sheep': 3}
horse
zad2/dataset/predict\horse.jpg
Skutočná trieda: horse, predikovaná trieda: horse
```

Skóre presnosti, konfúzna matica, klasifikačný report

Výpis presnosti matice podľa funkcie *evaluate()* dosiahol 59%. Presnosť matice cez *accuracy_score()* dosahuje tiež 59%. Klasifikačná matica nám ukazuje koľko výsledkov máme správnych podľa našich tried. Snahou je dosiahnuť najviac čísel na diagonále. Ako môžeme vidieť naša sieť si pomerne často zamieňala kôň so psom, alebo slon so psom. Túto neduhu sa budeme snažiť odstrániť v experimentoch.

```
results_validation = cnn.evaluate(test_set, batch_size=batch_size)
print("test loss, test acc:", results_validation)

acc = accuracy_score(test_set.classes, predictions)
print("Accuracy_score:", acc)

cm = confusion_matrix(test_set.classes, predictions)
print("Confusion matrix:\n", cm)

plt.figure()
sns.heatmap(cm, annot=True, fmt='d',
            xticklabels=list(index_to_class.values()),
            yticklabels=list(index_to_class.values()))
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()

print("Classification report:")
print(classification_report(
    test_set.classes,
    predictions,
    target_names=list(index_to_class.values())
))
```




```
test loss, test acc: [1.2451571226119995, 0.5892857313156128]
Accuracy_score: 0.5892857142857143
Confusion matrix:
 [[230  71  22  27]
 [ 38 271  16  25]
 [108  72 123  47]
 [ 53  62  34 201]]
Classification report:
              precision    recall  f1-score   support

         dog       0.54      0.66      0.59       350
    elephant       0.57      0.77      0.66       350
       horse       0.63      0.35      0.45       350
       sheep       0.67      0.57      0.62       350

    accuracy                           0.59      1400
   macro avg       0.60      0.59      0.58      1400
weighted avg       0.60      0.59      0.58      1400
```

Uloženie a načítanie modelu

Na uloženie a načítanie modelu bol používaný kód:

```
# ============================================================
# ULOŽENIE MODELU
# ============================================================

model_json = cnn.to_json()
with open('cnn.json', 'w') as json_file:
    json_file.write(model_json)

save_model(cnn, 'weights.hdf5')
print("Model uložený.")


# ============================================================
# NAČÍTANIE MODELU
# ============================================================

with open('cnn.json', 'r') as json_file:
    json_saved_model = json_file.read()

network_loaded = model_from_json(json_saved_model)
network_loaded.load_weights('weights.hdf5')

network_loaded.compile(
    loss='categorical_crossentropy',
    optimizer='adam',
    metrics=['accuracy']
)

network_loaded.summary()
print("Model úspešne načítaný.")
```

Experimenty

V experimentoch budeme manipulovať so sieťami, napríklad: s počtom filtrov, počtom neurónov, batch size atď..

Experiment 1 – „Silnejšia sieť“ (viac filtrov + Dropout)

Cieľ: lepšie vytiahnuť detaily, aby si sieť menej mýlila podobné zvieratá.

Kód:

```
cnn = Sequential()
cnn.add(Conv2D(32, 3, activation='relu', input_shape=(64, 64, 3)))
cnn.add(MaxPooling2D(2, 2))

cnn.add(Conv2D(32, 3, activation='relu'))
cnn.add(MaxPooling2D(2, 2))

cnn.add(Conv2D(64, 3, activation='relu'))
cnn.add(MaxPooling2D(2, 2))

cnn.add(Flatten())
cnn.add(Dense(256, activation='relu'))
cnn.add(tf.keras.layers.Dropout(0.5))
cnn.add(Dense(num_classes, activation='softmax'))

cnn.compile(
    optimizer='adam',
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

cnn.summary()
```

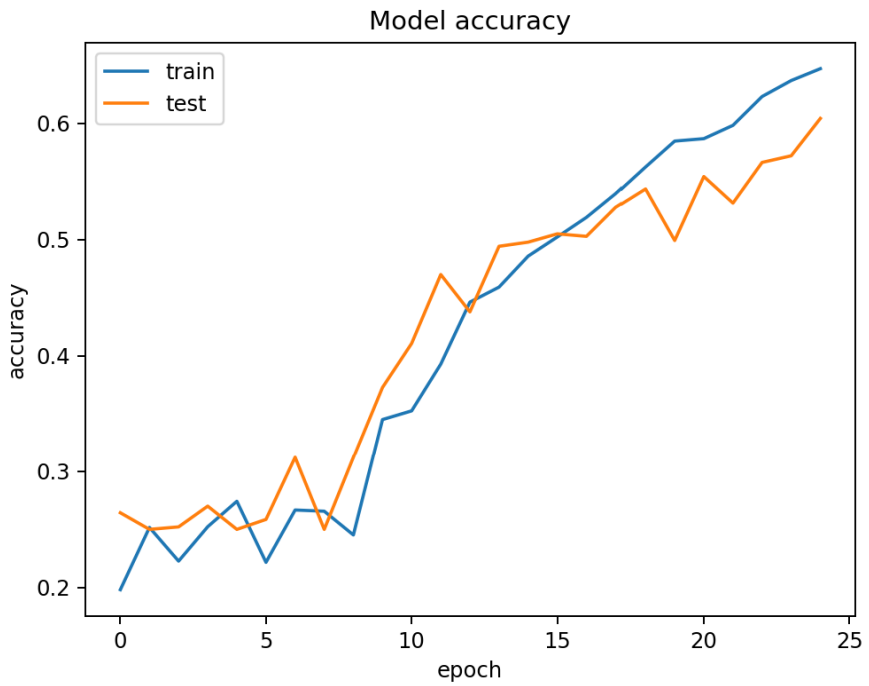
Architektúra:

Model: "sequential"

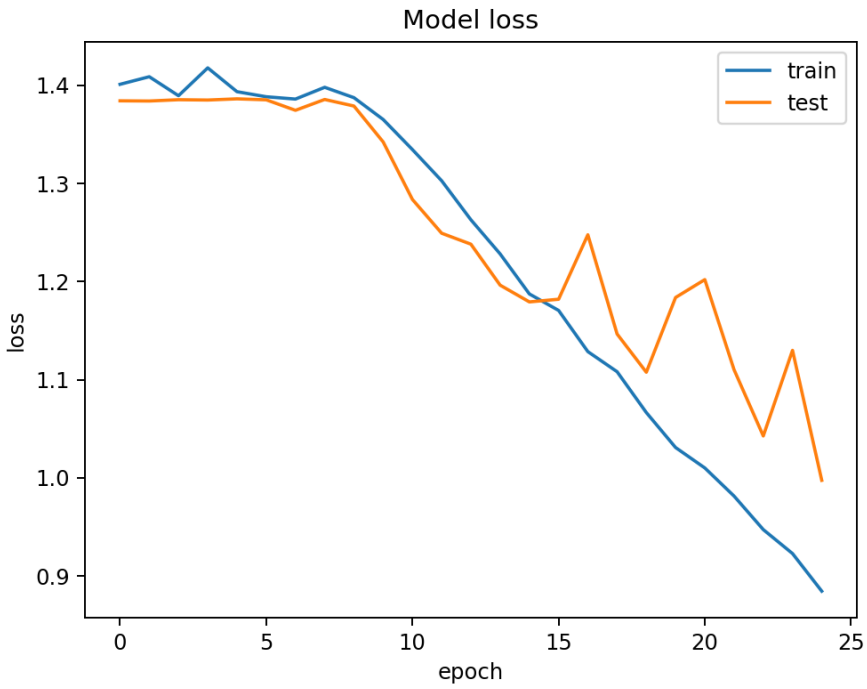
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 62, 62, 32)	896
max_pooling2d (MaxPooling2D)	(None, 31, 31, 32)	0
conv2d_1 (Conv2D)	(None, 29, 29, 32)	9,248
max_pooling2d_1 (MaxPooling2D)	(None, 14, 14, 32)	0
conv2d_2 (Conv2D)	(None, 12, 12, 64)	18,496
max_pooling2d_2 (MaxPooling2D)	(None, 6, 6, 64)	0
flatten (Flatten)	(None, 2304)	0
dense (Dense)	(None, 256)	590,080
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 4)	1,028

Total params: 619,748 (2.36 MB)
Trainable params: 619,748 (2.36 MB)
Non-trainable params: 0 (0.00 B)

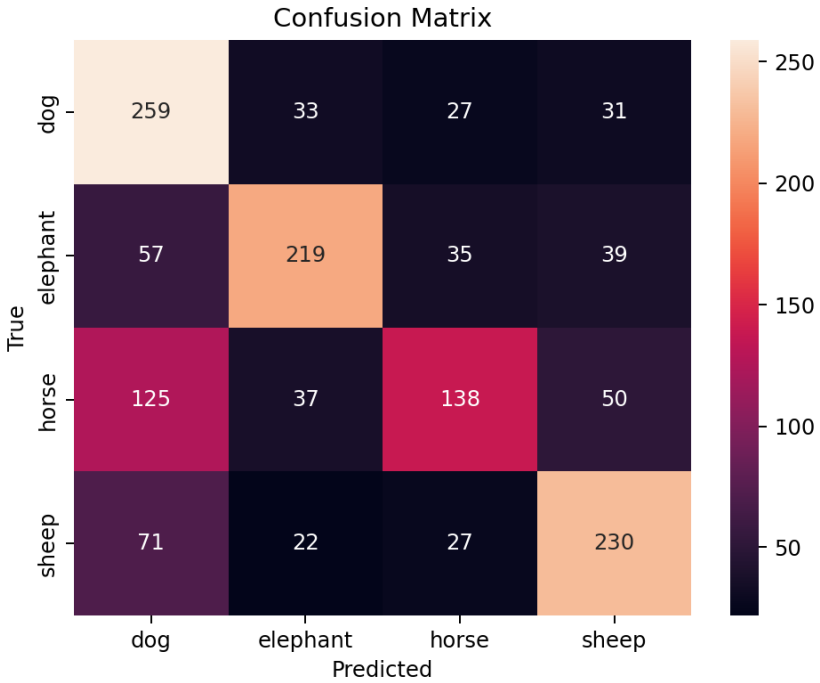
Presnosť:



Loss:



Skóre presnosti, konfúzna matica, klasifikačný report:



```
test loss, test acc: [0.9973952770233154, 0.6042857170104981]
Accuracy_score: 0.6042857142857143
Confusion matrix:
 [[259  33  27  31]
 [ 57 219  35  39]
 [125  37 138  50]
 [ 71  22  27 230]]
Classification report:
              precision    recall  f1-score   support

         dog       0.51      0.74      0.60       350
    elephant       0.70      0.63      0.66       350
       horse       0.61      0.39      0.48       350
       sheep       0.66      0.66      0.66       350

    accuracy                           0.60      1400
   macro avg       0.62      0.60      0.60      1400
weighted avg       0.62      0.60      0.60      1400
```

Takmer rovnaká ako pôvodný sieť (0.59) → takže zmena architektúry priniesla len malé zlepšenie.

V experimente 1 bola použitá rozšírená architektúra siete s vyšším počtom filtrov a treťou konvolučnou vrstvou. Test accuracy sa mierne zvýšila z hodnoty 59 % (pôvodná sieť) na 60 %.

Napriek tomu trieda horse zostáva najslabšou kategóriou, pričom veľká časť obrázkov koňa je stále nesprávne klasifikovaná ako dog alebo sheep. Z toho vyplýva, že zvýšenie kapacity siete síce pomohlo, ale model stále nedokáže zachytiť jemné tvarové a štruktúrne rozdiely medzi týmito triedami.

V ďalšom experimente sa preto zameriame na zlepšenie generalizácie pomocou regularizácie a augmentácie dát, keďže predpokladáme, že časť chýb môže byť spôsobená preučením modelu na vizuálne podobné obrázky.

Experiment 2 – „Regularizácia a silnejšia augmentácia“

Cieľ: znížiť overfitting, sieť nebude mať dobrú presnosť len na train, ale aj na test; tým sa zlepší matrix.

Augmentácia (náhodné rotácie, posuny, zmeny) a L2 regularizácia pomáhajú modelu naučiť sa robustnejšie príznaky.

Čo zmeníme:

- necháme **základnú architektúru** (2×Conv + Dense 128), aby sme videli čisto efekt regularizácie,
- v trénovacom generátore pridáme:
    - rotation_range=15
    - width_shift_range=0.1
    - height_shift_range=0.1
- do Dense vrstvy pridáme L2 regularizáciu a Dropout(0.5),
- znížime počet epoch (napr. na 20)

Kód:

```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1
)
```

```
cnn = Sequential()
cnn.add(Conv2D(32, 3, activation='relu', input_shape=(64, 64, 3)))
cnn.add(MaxPooling2D(2, 2))

cnn.add(Conv2D(32, 3, activation='relu'))
cnn.add(MaxPooling2D(2, 2))

cnn.add(Flatten())
cnn.add(Dense(128, activation='relu',kernel_regularizer=regularizers.l2(1e-4)))
cnn.add(tf.keras.layers.Dropout(0.5))
cnn.add(Dense(num_classes, activation='softmax'))

cnn.compile(
    optimizer='adam',
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

cnn.summary()
```

```
hist = cnn.fit(
    x=training_set,
    validation_data=test_set,
    epochs=20
)
```

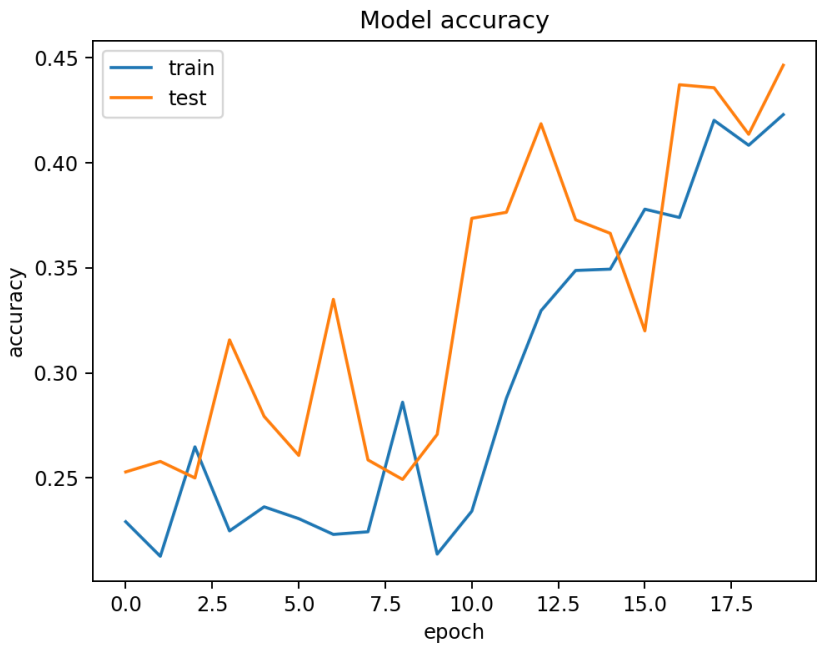
Architektúra:

Model: "sequential"

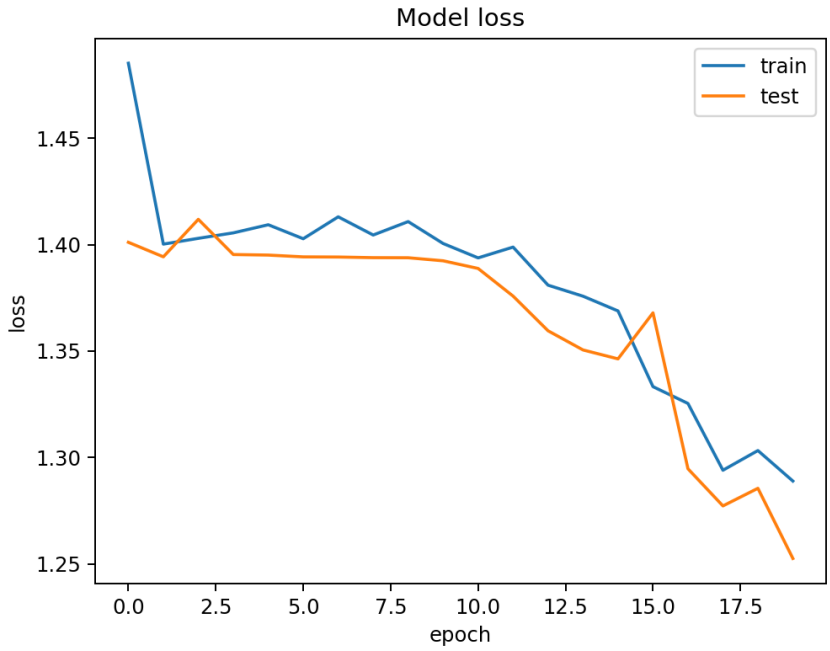
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 62, 62, 32)	896
max_pooling2d (MaxPooling2D)	(None, 31, 31, 32)	0
conv2d_1 (Conv2D)	(None, 29, 29, 32)	9,248
max_pooling2d_1 (MaxPooling2D)	(None, 14, 14, 32)	0
flatten (Flatten)	(None, 6272)	0
dense (Dense)	(None, 128)	802,944
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 4)	516

Total params: 813,604 (3.10 MB)
Trainable params: 813,604 (3.10 MB)
Non-trainable params: 0 (0.00 B)

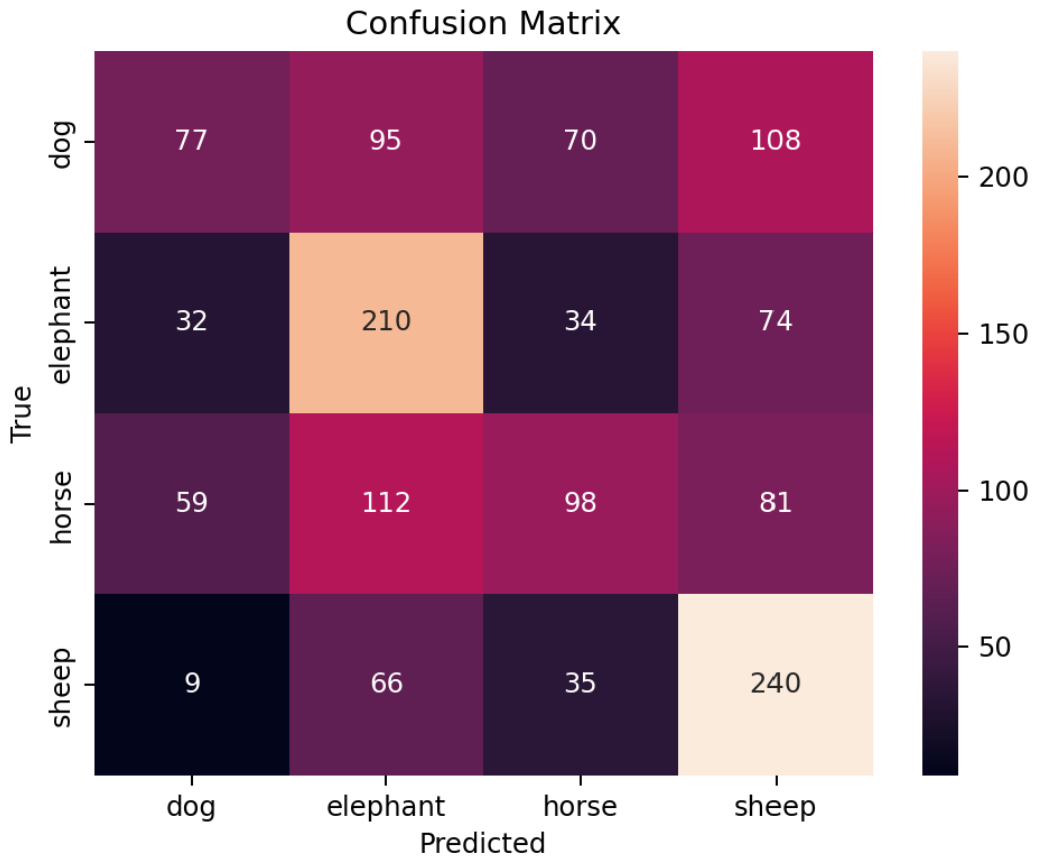
Presnosť:



Loss:



Skóre presnosti, konfúzna matica, klasifikačný report:




```
test loss, test acc: [1.2525451183319092, 0.4464285671710968]
Accuracy_score: 0.44642857142857145
Confusion matrix:
 [[ 77  95  70 108]
 [ 32 210  34  74]
 [ 59 112  98  81]
 [  9  66  35 240]]
Classification report:
              precision    recall  f1-score   support

         dog       0.44      0.22      0.29       350
    elephant       0.43      0.60      0.50       350
       horse       0.41      0.28      0.33       350
       sheep       0.48      0.69      0.56       350

    accuracy                           0.45      1400
   macro avg       0.44      0.45      0.42      1400
weighted avg       0.44      0.45      0.42      1400
```

Po natrénovaní počas 20 epoch dosiahla sieť test accuracy 45 %, čo je najmenej spomedzi všetkých experimentov. Z konfúznej matice vyplýva, že hoci sa mierne zlepšila klasifikácia tried elephant (recall = 0.60) a sheep (recall = 0.69), výkon tried dog a horse výrazne klesol (recall 0.22, resp. 0.28). Model často zamieňa tieto triedy medzi sebou a s ostatnými kategóriami.

Výsledky naznačujú, že kombinácia silnej augmentácie a regularizácie spôsobila, že sieť je príliš „opatrná“ a nedokáže sa dostatočne prispôsobiť trénovacím dátam.

Experiment 3 – Väčšie rozlíšenie obrázka (128×128) + jednoduchšia hlava

Obrázky 64×64 nemusia stačiť na jemné rozdiely (napr. textúra srsti, roh ovce vs. uši psa/koňa).
Ak zväčšíš vstup na 128×128, sieť dostane viac informácie. Aby model extrémne nenarástol, zvolíme jednoduchší „vrch“ siete.

Kód:

```
img_size = (128, 128)

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

test_datagen = ImageDataGenerator(rescale=1./255)

training_set = train_datagen.flow_from_directory(
    'zad2/dataset/training_set',
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False
)

test_set = test_datagen.flow_from_directory(
    'zad2/dataset/test_set',
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False
)
```

```python
cnn = Sequential()
cnn.add(Conv2D(32, 3, activation='relu', input_shape=(128, 128, 3)))
cnn.add(MaxPooling2D(2, 2))

cnn.add(Conv2D(64, 3, activation='relu'))
cnn.add(MaxPooling2D(2, 2))

cnn.add(Conv2D(64, 3, activation='relu'))
cnn.add(MaxPooling2D(2, 2))

cnn.add(Flatten())
cnn.add(Dense(64, activation='relu'))
cnn.add(Dense(num_classes, activation='softmax'))

cnn.compile(
    optimizer='adam',
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

cnn.summary()
```

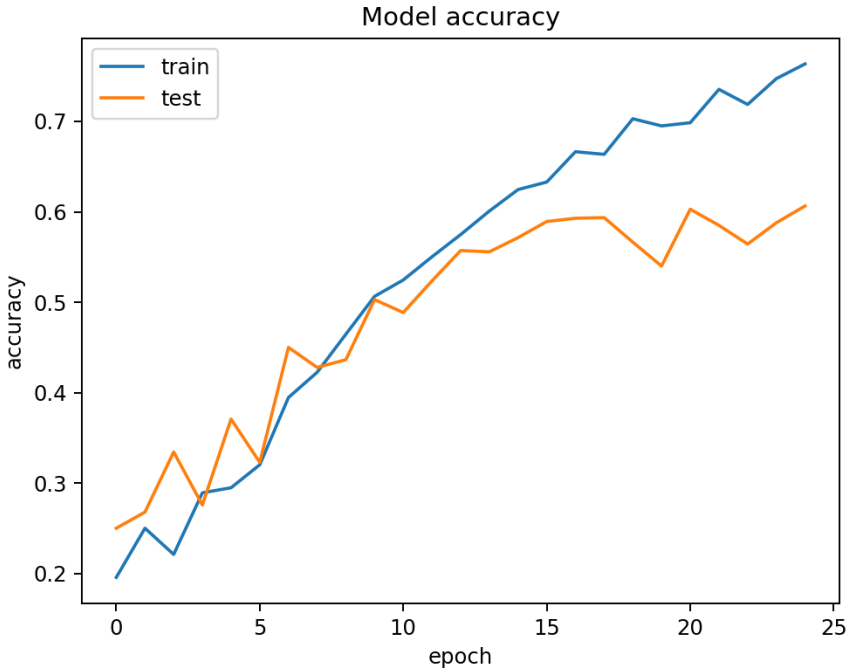
Architektúra:

Model: "sequential"

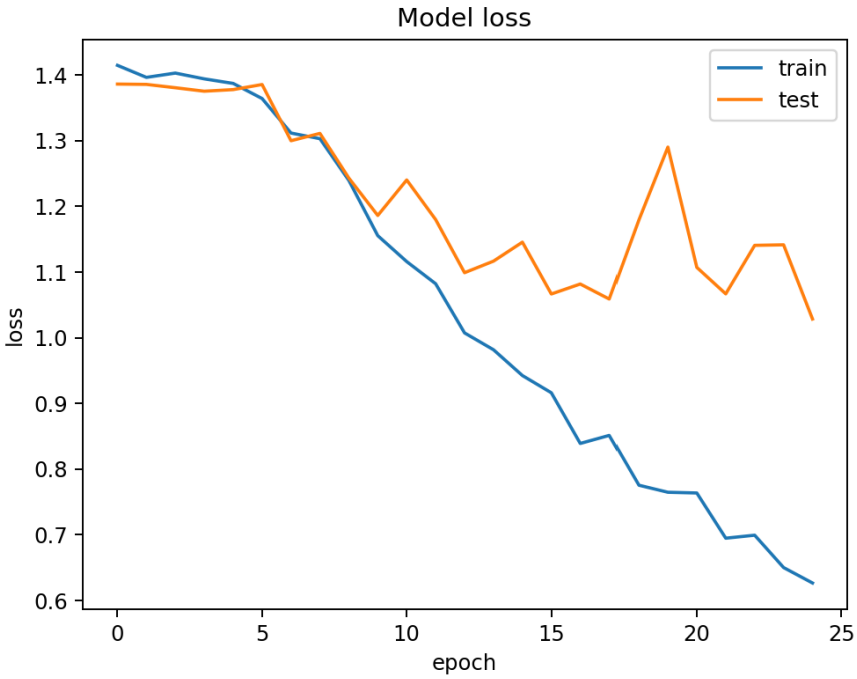
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 126, 126, 32)	896
max_pooling2d (MaxPooling2D)	(None, 63, 63, 32)	0
conv2d_1 (Conv2D)	(None, 61, 61, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 30, 30, 64)	0
conv2d_2 (Conv2D)	(None, 28, 28, 64)	36,928
max_pooling2d_2 (MaxPooling2D)	(None, 14, 14, 64)	0
flatten (Flatten)	(None, 12544)	0
dense (Dense)	(None, 64)	802,880
dense_1 (Dense)	(None, 4)	260

Total params: 859,460 (3.28 MB)
Trainable params: 859,460 (3.28 MB)
Non-trainable params: 0 (0.00 B)

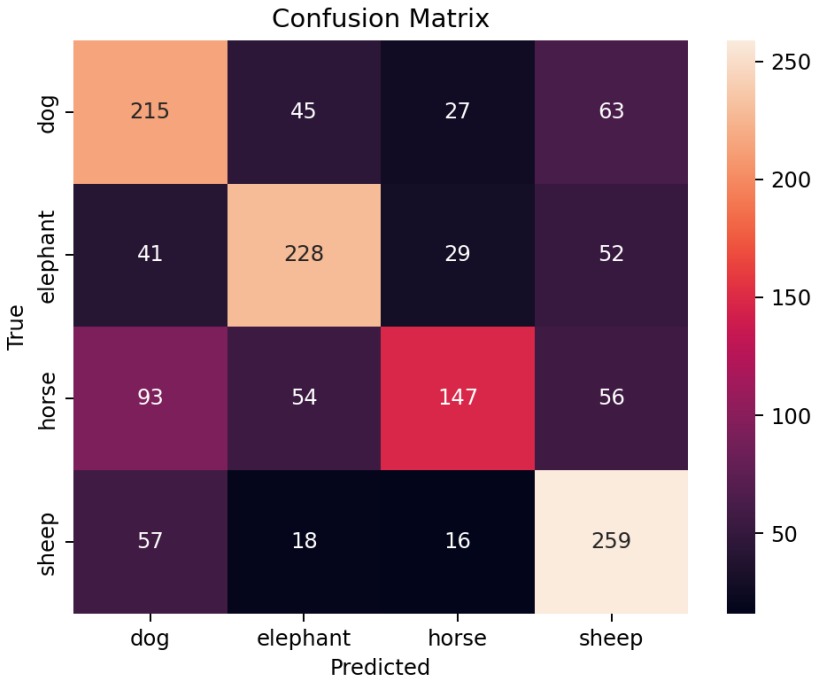
Presnosť:



Loss:



Skóre presnosti, konfúzna matica, klasifikačný report:




```
test loss, test acc: [1.028511643409729, 0.6064285635948181]
Accuracy_score: 0.6064285714285714
Confusion matrix:
 [[215  45  27  63]
 [ 41 228  29  52]
 [ 93  54 147  56]
 [ 57  18  16 259]]
Classification report:
              precision    recall  f1-score   support

         dog       0.53      0.61      0.57       350
    elephant       0.66      0.65      0.66       350
       horse       0.67      0.42      0.52       350
       sheep       0.60      0.74      0.66       350

    accuracy                           0.61      1400
   macro avg       0.62      0.61      0.60      1400
weighted avg       0.62      0.61      0.60      1400
```

V experimente 3 bolo zvýšené rozlíšenie vstupných obrázkov zo 64×64 na 128×128 pixelov a mierne bola upravená kapacita konvolučnej časti siete. Tento experiment dosiahol najvyššiu test accuracy zo všetkých vykonaných experimentov (61 %).

Z konfúznej matice vyplýva, že sieť výrazne zlepšila klasifikáciu triedy sheep (recall = 0.74) a stabilizovala výkon v triedach elephant a dog. V porovnaní s predchádzajúcimi experimentmi sa mierne zlepšila aj trieda horse (f1-score = 0.52), hoci táto kategória naďalej zostáva najnáročnejšou.

Výsledky naznačujú, že vyššie rozlíšenie obrázkov poskytlo modelu viac detailov o tvare tela a štruktúre objektov, čo viedlo k presnejšiemu rozpoznávaniu. Tento experiment tak potvrdil predpoklad, že slabý výkon v triede horse bol spôsobený najmä stratou vizuálnych detailov pri nízkom rozlíšení.

Záver

V práci bola navrhnutá a natrénovaná konvolučná neurónová sieť na klasifikáciu štyroch tried zvierat (dog, elephant, horse, sheep). Základný model dosiahol presnosť 59 %, pričom z konfúznej matice bolo zrejmé, že sieť mala problém najmä s triedami *dog* a *horse*. Z tohto dôvodu boli vykonané tri experimenty, v ktorých sa menila architektúra siete, miera regularizácie a rozlíšenie vstupných obrázkov.

Výsledky ukázali, že rozhodujúci vplyv na kvalitu klasifikácie má najmä informatívnosť vstupných dát a množstvo zachytených detailov v obraze. Zvýšenie rozlíšenia malo väčší prínos ako samotné zväčšovanie architektúry. Do budúcna by bolo vhodné otestovať modernejšie CNN architektúry a transfer learning.