Krzysztof Bohdanowicz

nr albumu 253933

Zadanie Projektowe 2

Przetwarzanie obrazów

https://github.com/kbohdanowicz/cats_and_dogs_image_classification

1. Baza danych

Baza danych zawiera 25000 zdjęć psów i kotów. Została podzielona na treningowe oraz testowe zestawy danych w proporcji 4:1.

2. Przetwarzanie danych

Dla sieci neuronowej CNN zdjęcia są bezpośrednio pobierane z folderu. Dodatkowo została użyta metoda powiększania zestawu danych poprzez klasę *ImageDataGenerator* z odpowiednio dobranymi parametrami.

```
train_data_gen = ImageDataGenerator(rescale=1.0 / 255.0,
                                    width_shift_range=0.1,
                                    height_shift_range=0.1,
                                    horizontal_flip=True)

test_data_gen = ImageDataGenerator(rescale=1.0 / 255)
```

Zdjęcia zostały zmniejszone do rozmiaru 200 x 200 (zmienna *IMAGE_SIZE* = 200).

```
train_set = train_data_gen.flow_from_directory(f'{DATASET_DIRECTORY}train/',
                                               target_size=(IMAGE_SIZE, IMAGE_SIZE),
                                               batch_size=_batch_size,
                                               class_mode='binary')

test_set = test_data_gen.flow_from_directory(f'{DATASET_DIRECTORY}test/',
                                             target_size=(IMAGE_SIZE, IMAGE_SIZE),
                                             batch_size=_batch_size,
                                             class_mode='binary')
```

Dla klasyfikatora KNN, zdjęcia są przerabiane na wektor kolorowych pikseli, histogram kolorów oraz listę etykiet.

```
raw_images = []
histogram = []
labels = []

image_paths = list(Path(f'{DATASET_DIRECTORY}').rglob("*.jpg"))

for (i, image_path) in enumerate(image_paths):
    image_path = str(image_path)
    image = cv2.imread(image_path)
    label = image_path.split(os.path.sep)[-1].split(".")[0]

    pixels = image_to_pixels(image)
    histogram = image_to_color_histogram(image)

    raw_images.append(pixels)
    histogram.append(histogram)
    labels.append(label)
```

Testy wydajności zostały wykonane na komputerze o następującej specyfikacji:

- System: Windows 10 Home 64-bit
- Procesor: AMD RYZEN 2700X 3700 Mhz
- Karta graficzna: GeForce GTX 1060 3GB

Do pracy sieci neuronowej użyto karty graficznej co kilkukrotnie zwiększyło wydajność sieci.

3. Klasyfikator KNN

Zostały przetestowane cztery wersje klasyfikatora:

- Podstawowa z obrazkami
- Podstawowa z histogramem
- Optymalna z obrazkami
- Optymalna z histogramem

Optymalne parametry zostały znalezione za pomocą klasy *RandomizedSearchCV.*

```
model = KNeighborsClassifier(n_jobs=-1)
params = {'n_neighbors': np.arange(1, 31, 3), 'metric': ['euclidean', 'cityblock']}
grid = RandomizedSearchCV(model, params)

start = time()
grid.fit(train, train_labels)
accuracy = grid.score(test, test_labels)

print('[INFO]: randomized search took {:.2f} seconds'.format(time() - start))
print('[INFO]: grid search accuracy: {:.2f}%'.format(accuracy * 100))
print('[INFO]: randomized search best parameters: {}'.format(grid.best_params_))
```

Parametry podstawowej wersji:

- liczba sąsiadów = 1
- euklidesowa metoda mierzenia odległości.

Parametry optymalnej wersji:

- liczba sąsiadów = 28
- taksówkowa metoda mierzenia odległości.

W obu przypadkach (obrazki i histogram) optymalne parametry były takie same.

4. Konwolucyjna sieć neuronowa

Zostały przetestowane cztery wersje sieci CNN:

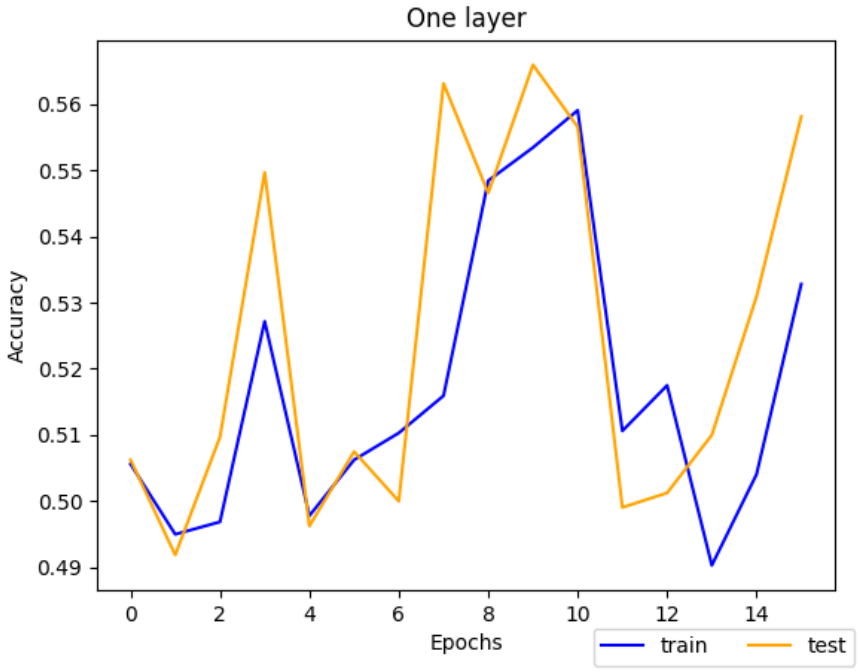
- Jedno warstwowa
- Trzy warstwowa
- Trzy warstwowa z dropoutem
- VGG16

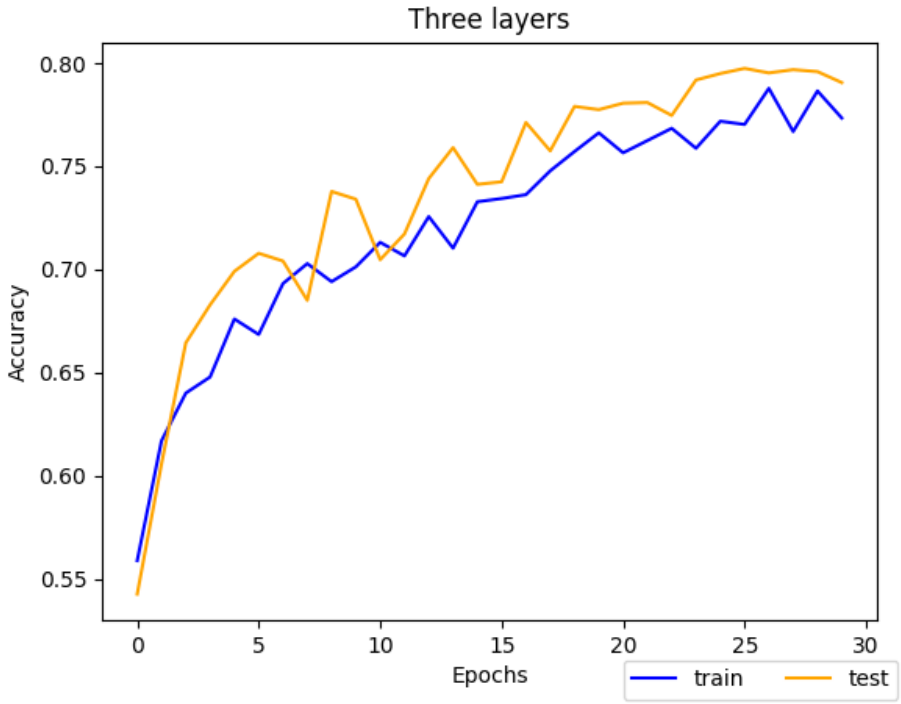
Dla sieci zastosowano dwie funkcje callback:

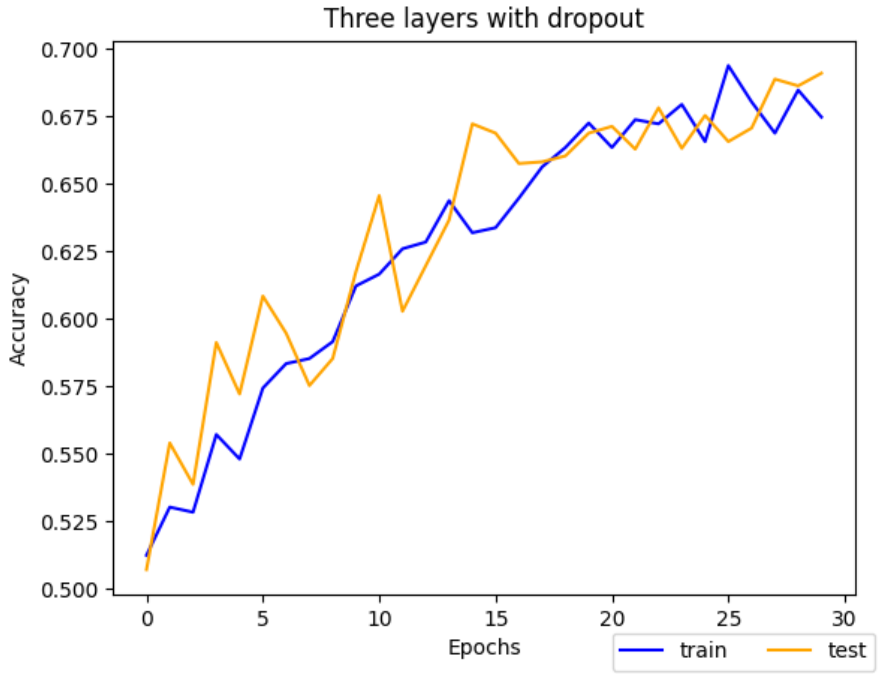
- *ReduceLROnPlateau* (zmniejszenie tempa nauczania przy braku dalszych postępów przy ostatnich 2 epokach)
- *EarlyStopping* (wczesne zatrzymywanie sieci przy braku dalszych postępów przy ostatnich 5 epokach)

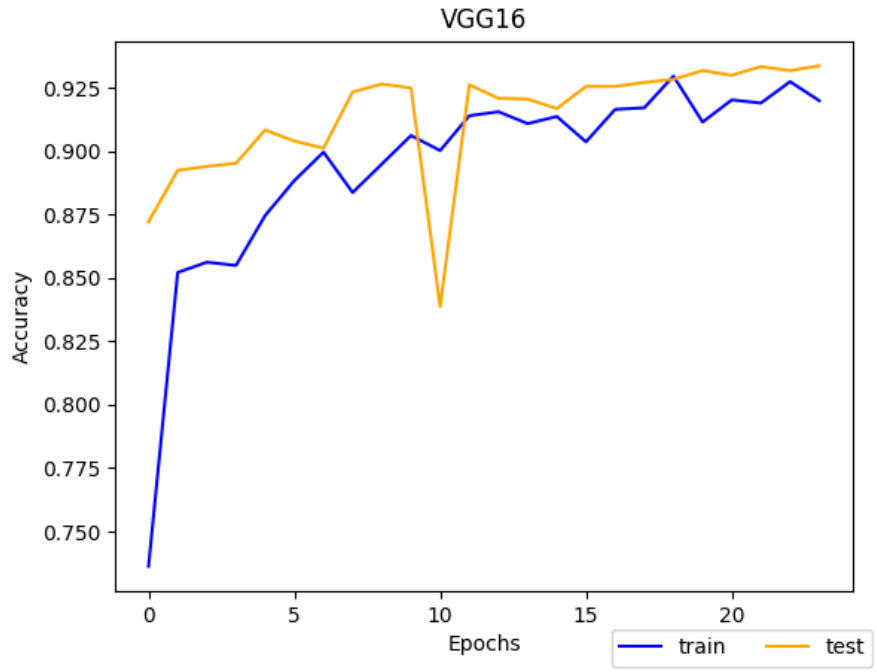
```
learning_rate_reduction = ReduceLROnPlateau(monitor='loss',
                                            mode="min",
                                            patience=2,
                                            verbose=1,
                                            factor=0.5,
                                            min_lr=0.00001)

early_stopping = EarlyStopping(monitor="loss",
                               mode="min",
                               patience=5,
                               restore_best_weights=True,
                               verbose=1)
```

5. Dokładność sieci neuronowej

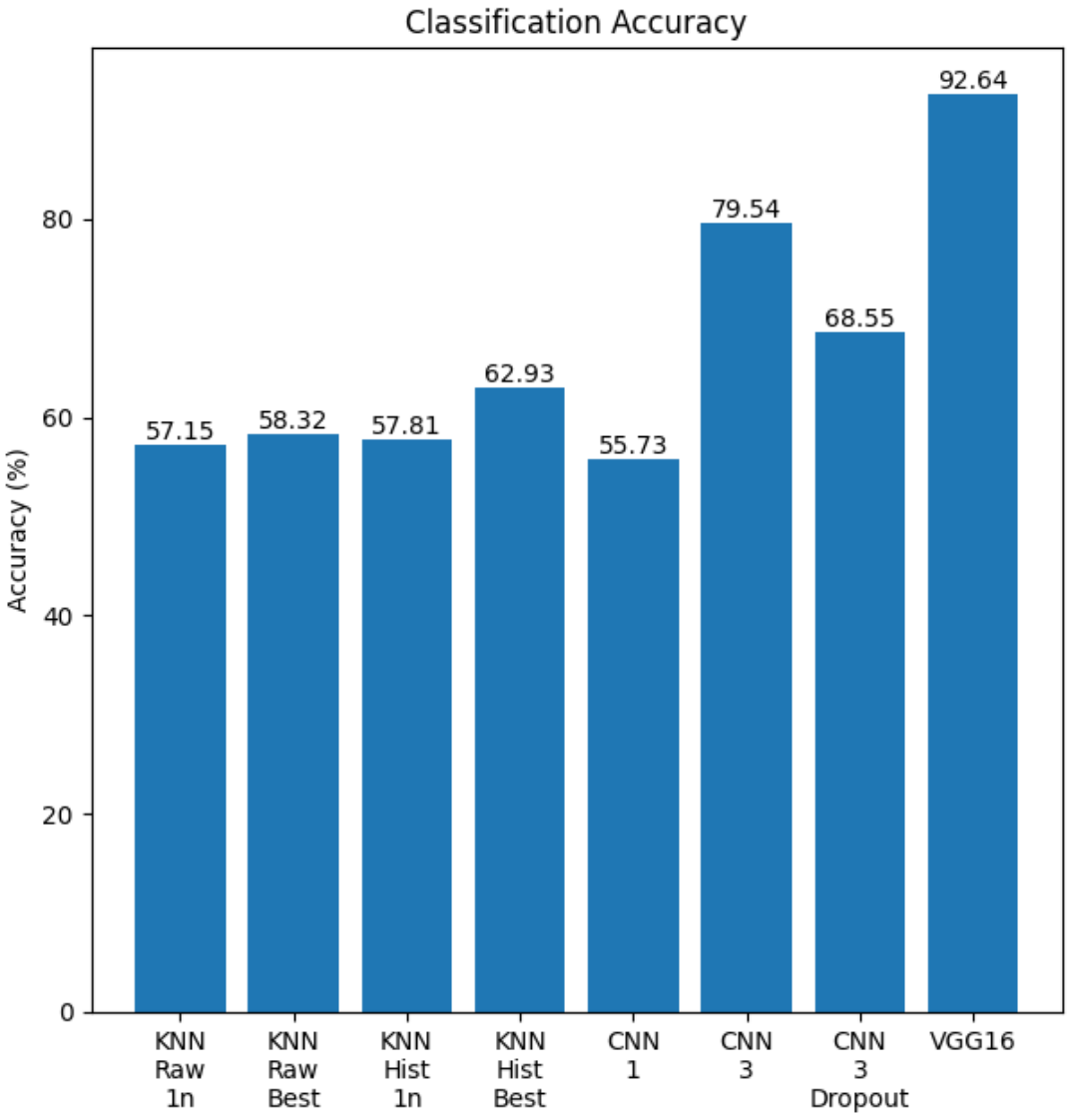




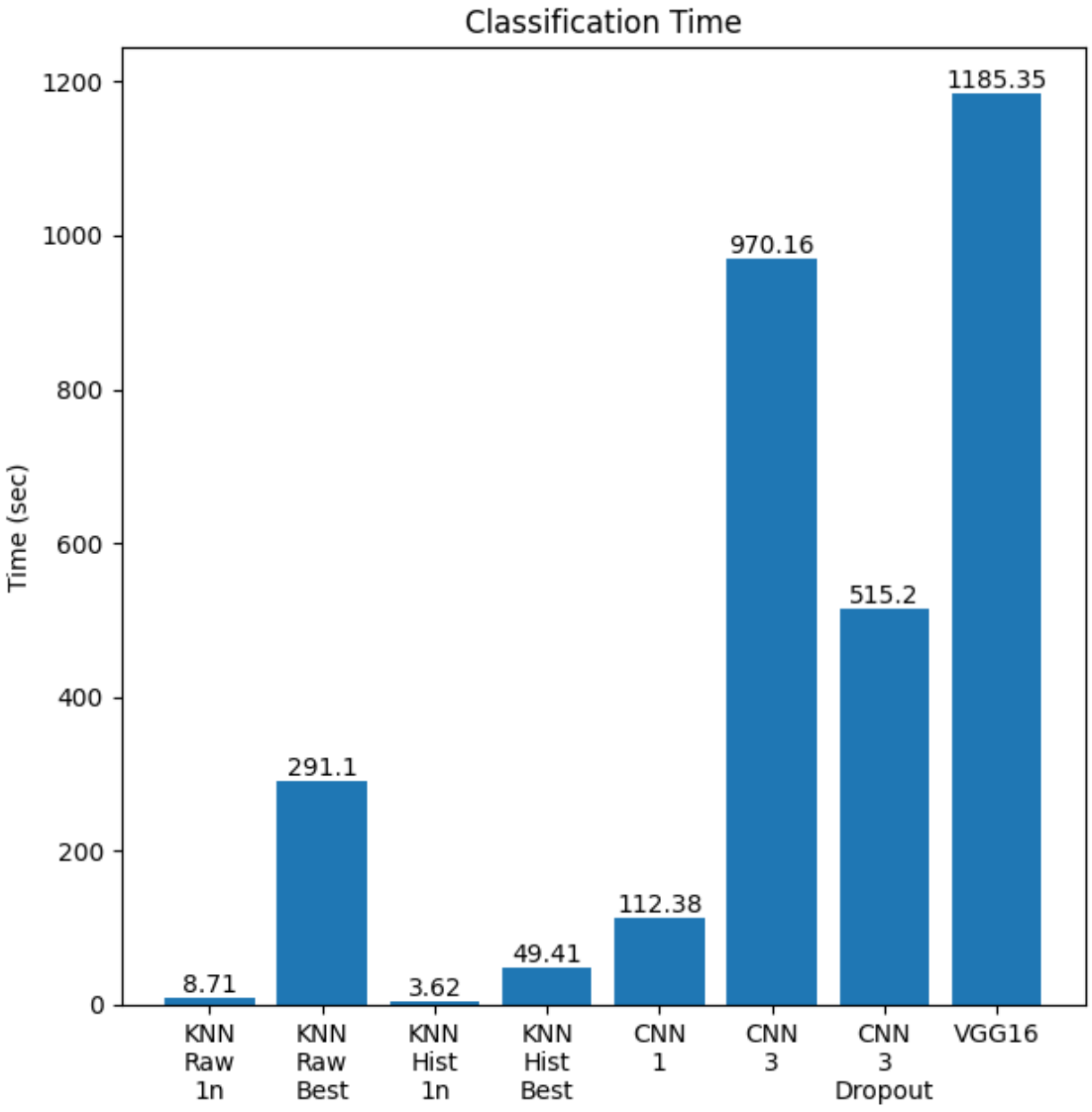

Three layers with dropout
0.700
0.675
0.650
0.625
0.600
0.575
0.550
0.525
0.500
Accuracy
0
5
10
15
20
25
30
Epochs
train
test


VGG16
0.925
0.900
0.875
0.850
0.825
0.800
0.775
0.750
Accuracy
0
5
10
15
20
Epochs
train
test

6. Dokładność klasyfikatorów

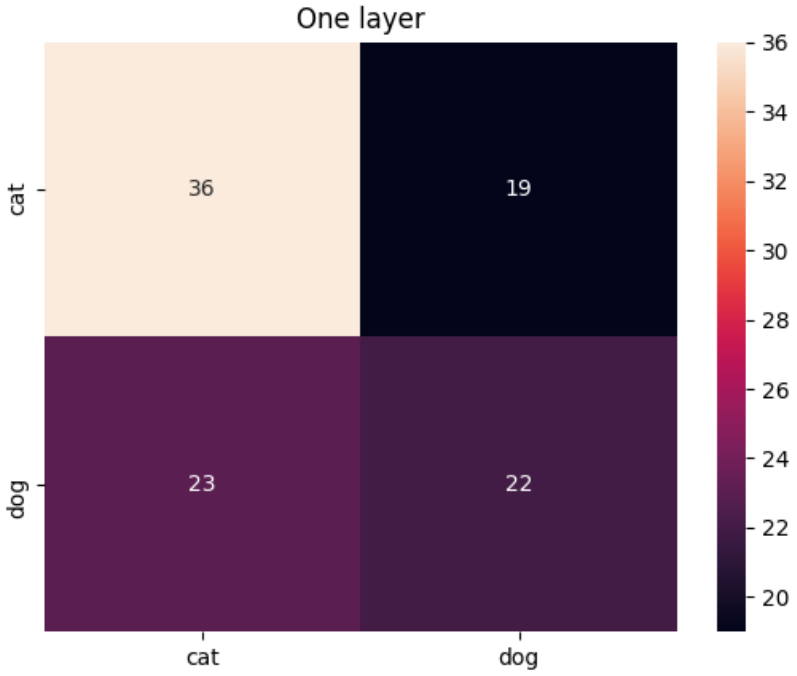


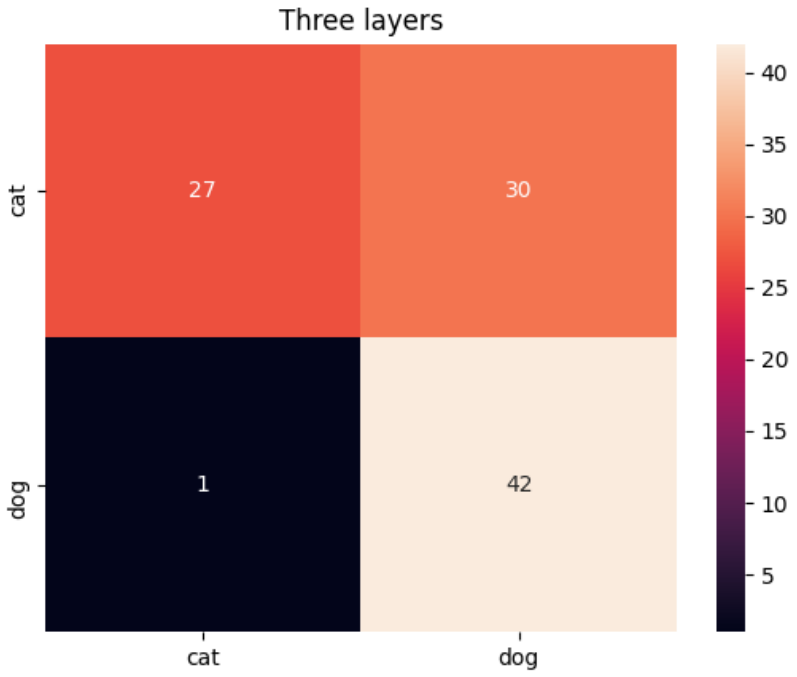
7. Czas działania klasyfikatorów

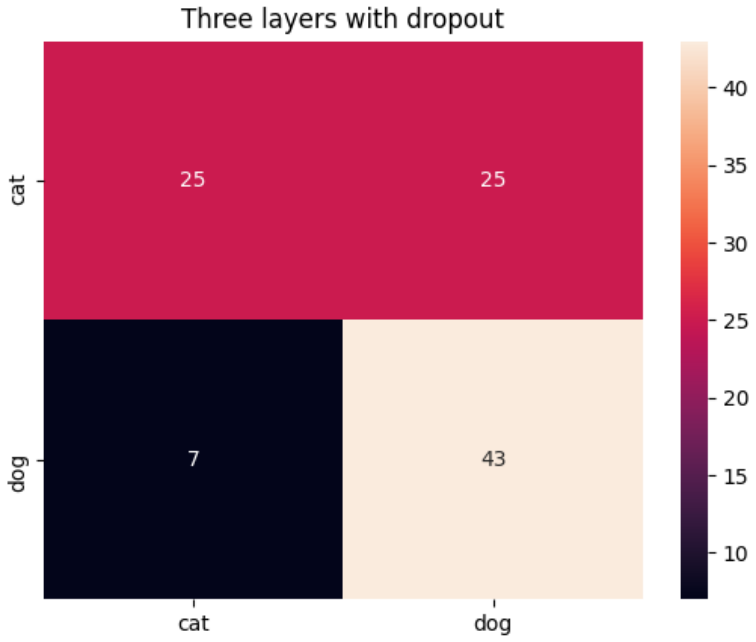


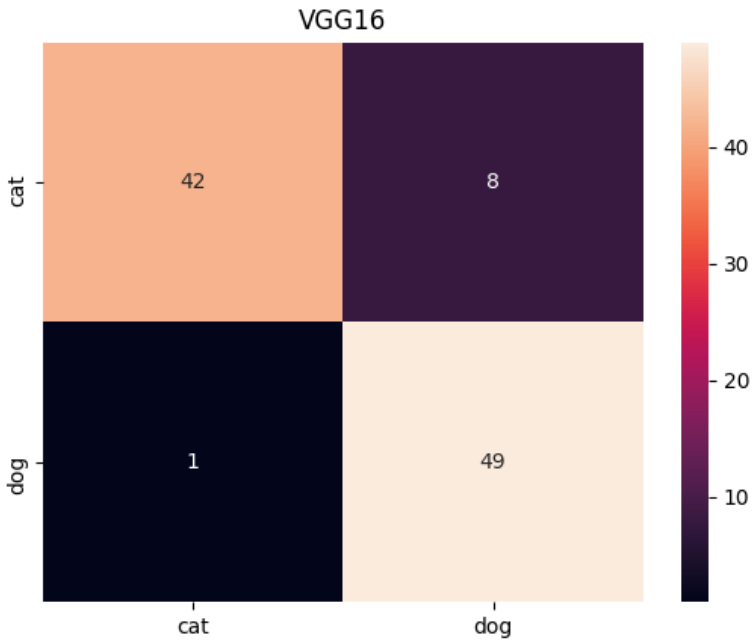
8. Macierze błędu

Ilość zdjęć = 100






Three layers with dropout
25
25
7
43
cat
dog
cat
dog
40
35
30
25
20
15
10


VGG16
42
8
1
49
cat
dog
cat
dog
40
30
20
10