Sztuczna inteligencja w systemach informatycznych - projekt

Zastosowanie sieci konwolucyjnych do rozpoznawania zwierząt

Skład zespołu:

- Dawid Białka
- Michał Czarnecki
- Michał Sokół
- Piotr Świderski
- Jakub Wydra

Opis datasetu

Dataset to zbiór 28 tysięcy obrazów zwierząt. Zwierzęta są podzielone na 10 kategorii:

- Psy
- Konie
- Słonie
- Motyle
- Kury
- Koty
- Krowy
- Owce
- Pająki
- Wiewiórki

Wszystkie zdjęcia zostały pobrane ze zdjęć google, oraz zostały sprawdzone przez człowieka.
Ilość zdjęć dla każdej kategorii jest między 2, a 5 tysięcy obrazów.
Rozmiar całego datasetu to 616 MB.

Lokalizacja datasetu:
https://www.kaggle.com/datasets/alessiocorrado99/animals10?resource=download

Plan pracy

1. Wybranie odpowiedniego modelu sieci
2. Edycja modelu, aby pasował do danych
3. Obróbka danych (augmentacja, grayscale itd.)
4. Wytrenowanie modelu na nowych danych
5. Korekta danych zaburzających uczenie
6. Ponowne wytrenowanie modelu
7. Analiza wyników

Eksperymenty

Wybranie odpowiedniego modelu sieci

Do analizy dostępnych modeli wykorzystaliśmy zestawienie przygotowanie przez keras.io
https://keras.io/api/applications/

Model	Size (MB)	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth	Time (ms) per inference step (CPU)	Time (ms) per inference step (GPU)
Xception	88	79.0%	94.5%	22.9M	81	109.4	8.1
EfficientNetB0	29	77.1%	93.3%	5.3M	132	46.0	4.9

Xception - model cechujący się wysoką dokładnością i średnim rozmiarem
EfficientNetB0 - model o stosunkowo niewielkich rozmiarach, który wyróżnia się na tle innych, modeli o małym rozmiarze, wysoką dokładnością

Edycja modelu, aby pasował do danych

Wykorzystane modele - tak Xception jak i EfficientNetB0 otrzymały dodatkowe warstwy dense

```
def addTopModel(bottom_model, num_classes):

    """creates the top or head of the model that will be
    placed ontop of the bottom layers"""

    top_model = bottom_model.output
    top_model = GlobalAveragePooling2D()(top_model)
    top_model = Dense(1024, activation='relu')(top_model)

    top_model = Dense(1024, activation='relu')(top_model)

    top_model = Dense(512, activation='relu')(top_model)

    top_model = Dense(num_classes, activation="softmax")(top_model)

    return top_model
```

Obróbka danych (augmentacja, grayscale itd.)

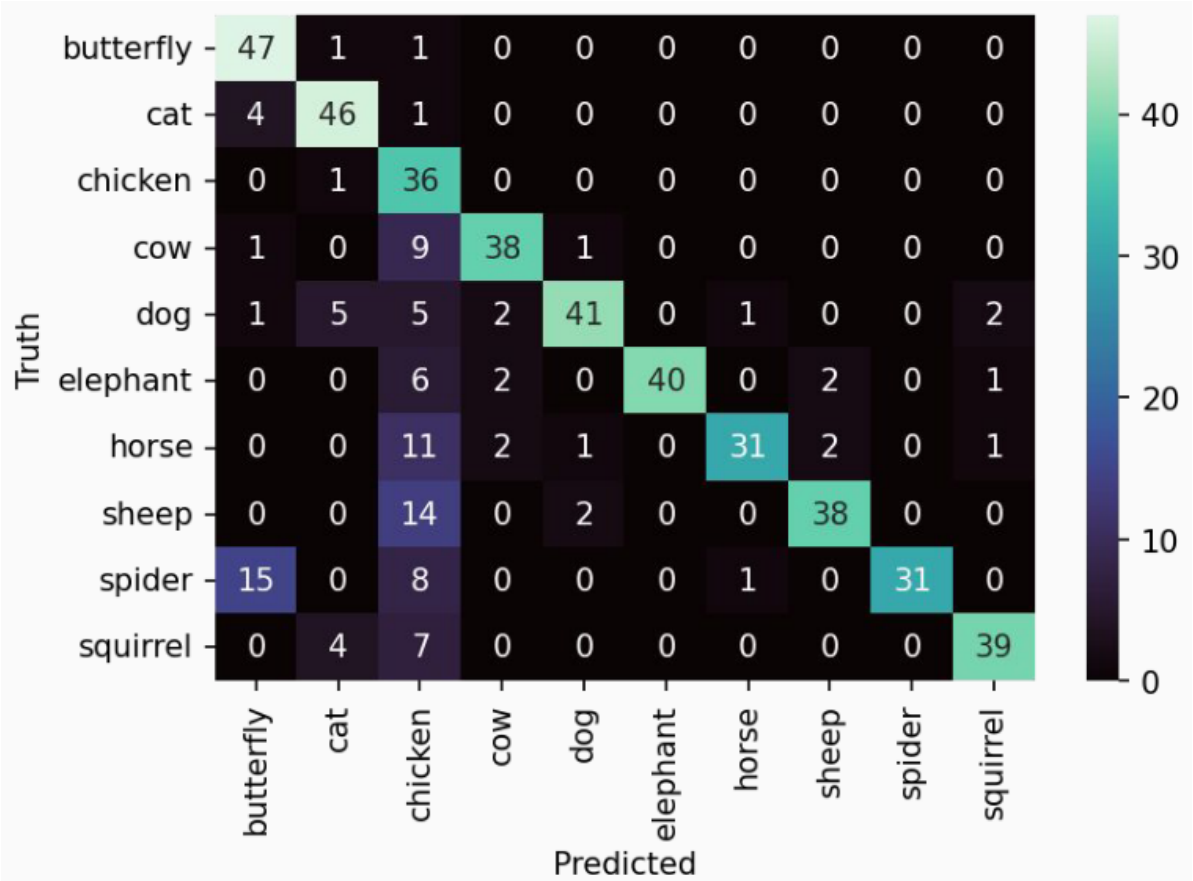
Proces trenowania na pełnym zbiorze danych był zbyt długotrwały na lokalnych komputerach, a narzędzia dostępne online (takie jak Google Colab) przerywały trening w połowie, po wyczerpaniu dostępnych zasobów.
Z tego względu zmniejszyliśmy dataset do 10 kategorii po 500 egzemplarzy z każdej, a następnie zredukowaliśmy rozmiary obrazków do 300x300 pikseli (stanowiących wejście dla rozpatrywanych modeli)

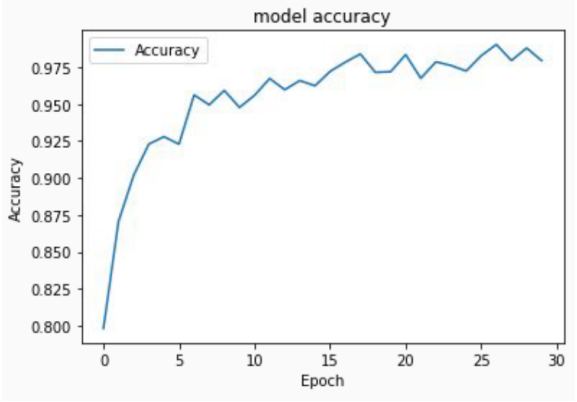
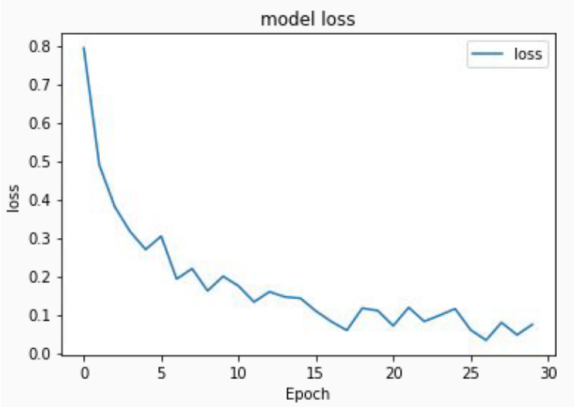
```
def create_data():
    for category in CATEGORIES:

        path = os.path.join(DATADIR, category)
        class_num = CATEGORIES.index(category)
        i = 0
        for img in tqdm(os.listdir(path)):
            img_array = cv2.imread(os.path.join(path, img))
            # Xception and EfficientnetB0 input image shape
            img_array = cv2.resize(img_array, (299,299))
            data.append([img_array, class_num])
            i += 1
            if i > 500:
              break
```

Wytrenowanie modelu na nowych danych

Xception

Confusion matrix

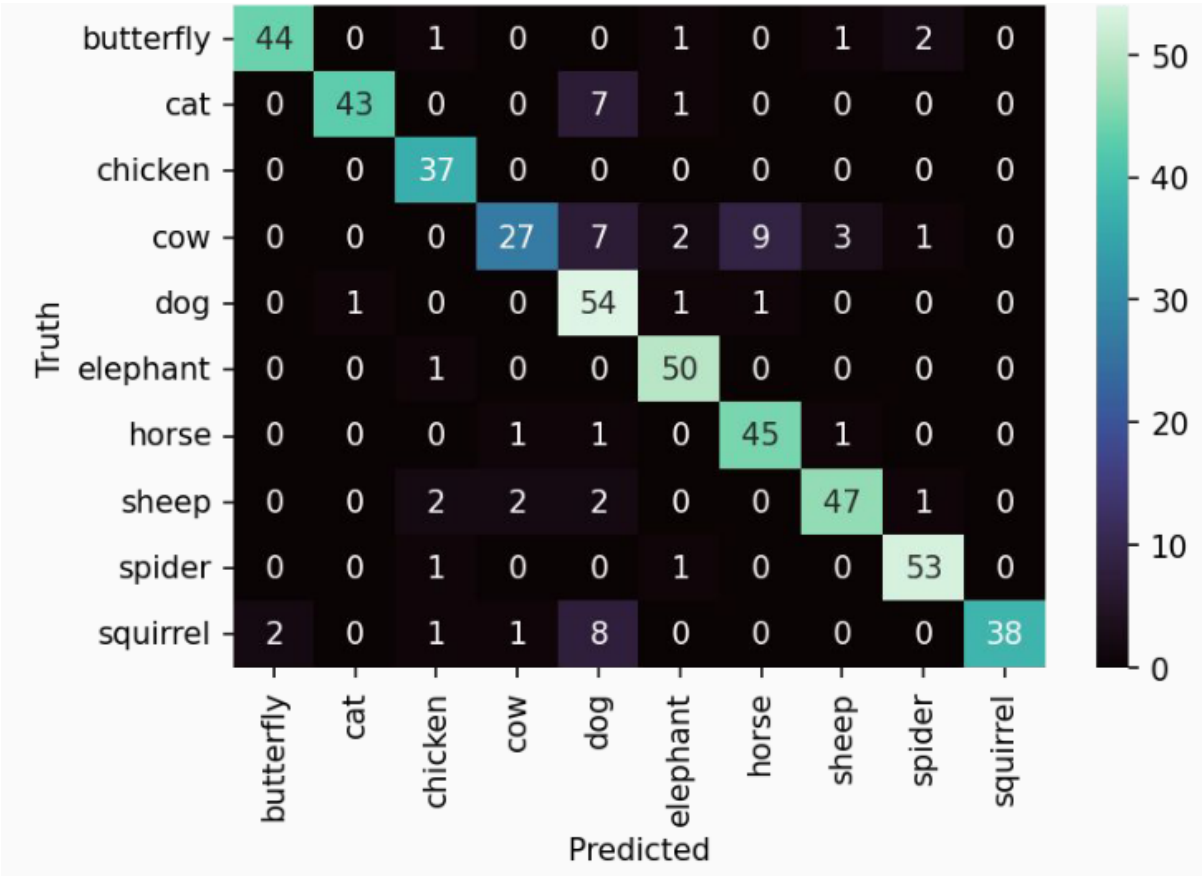


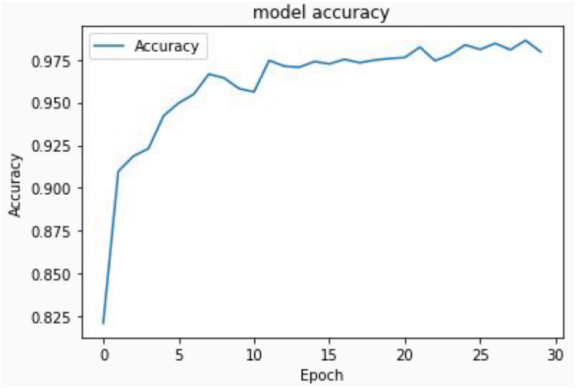
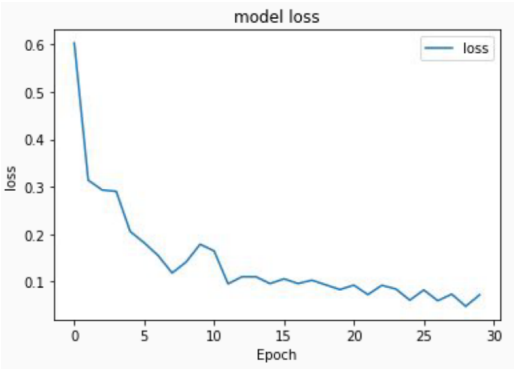
Model accuracy	Model loss
	

Dokładność na całym datasecie: 71.1% acc

EfficientNetB0

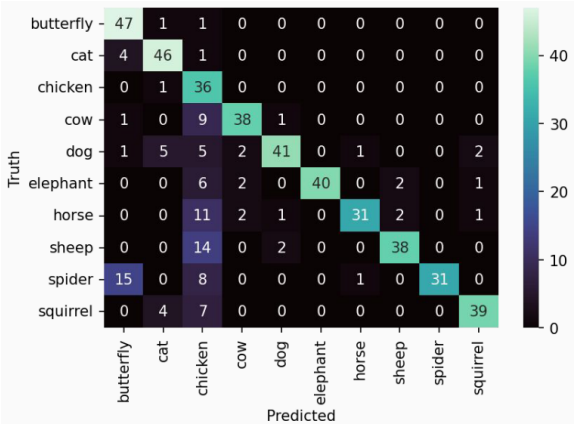
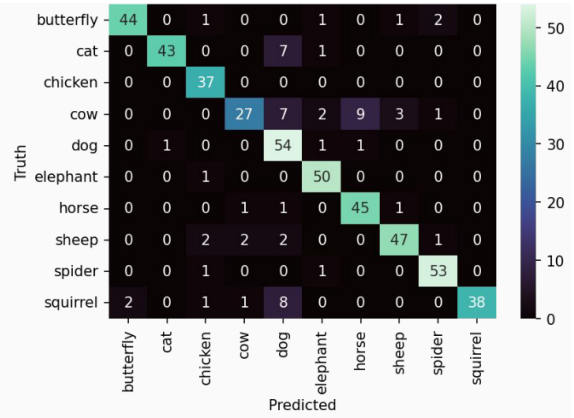
Confusion matrix



Model accuracy	Model loss
	

Dokładność na całym datasecie: 82.4% acc

Porównanie confusion matrix dla obu modeli

Xception	EfficientNetB0
	

Przykłady błędnie sklasyfikowanych obrazków

	
Poprawna klasa: słoń Predykcja: **owca**	Poprawna klasa: słoń Predykcja: **motyl**

Korekta danych zaburzających uczenie

W datasecie zauważyliśmy że pewna ilość obrazków w każdej kategorii nie ukazuje w pełni danego zwierzęcia, co może przekładać się na jakość modelu - jeśli takie obrazy były użyte jako wyznacznik do trenowania.

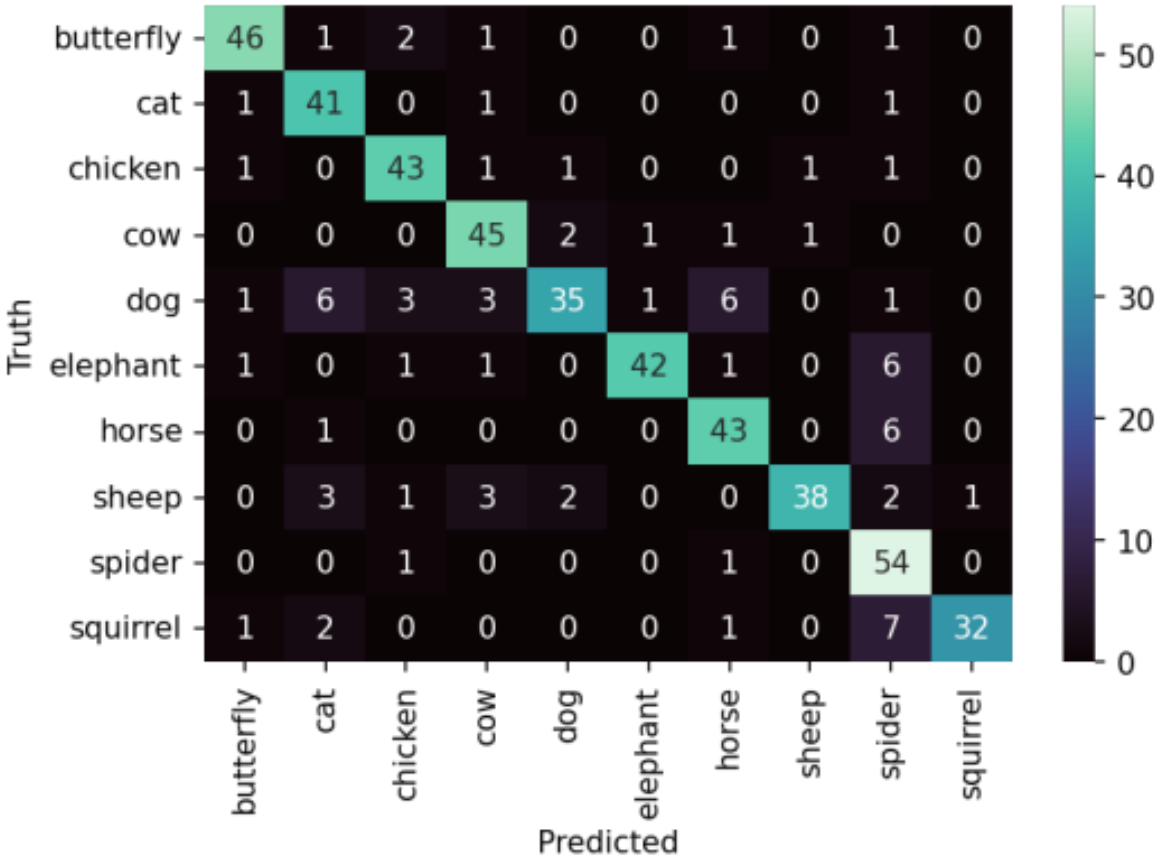
Do odseparowania ich wykorzystaliśmy już wytrenowane modele.
Każdy obrazek z każdej kategorii oceniany był przez oba modele. Jeśli obie predykcje były poprawne co do kategorii to zachowywaliśmy zdjęcie do ponownego treningu; jeśli nie to to zdjęcie było odrzucane.

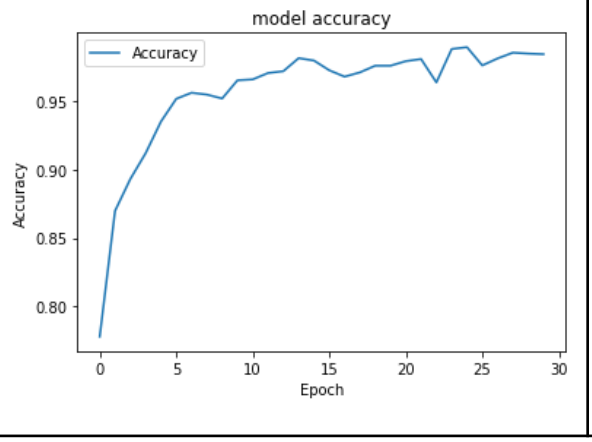
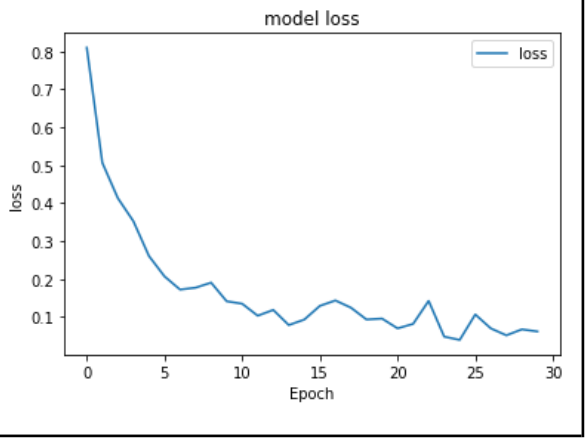
Proces zakończyliśmy wtedy, kiedy zebraliśmy po 500 obrazków z każdej kategorii.

Ponowne wytrenowanie modelu

Xception

Confusion matrix

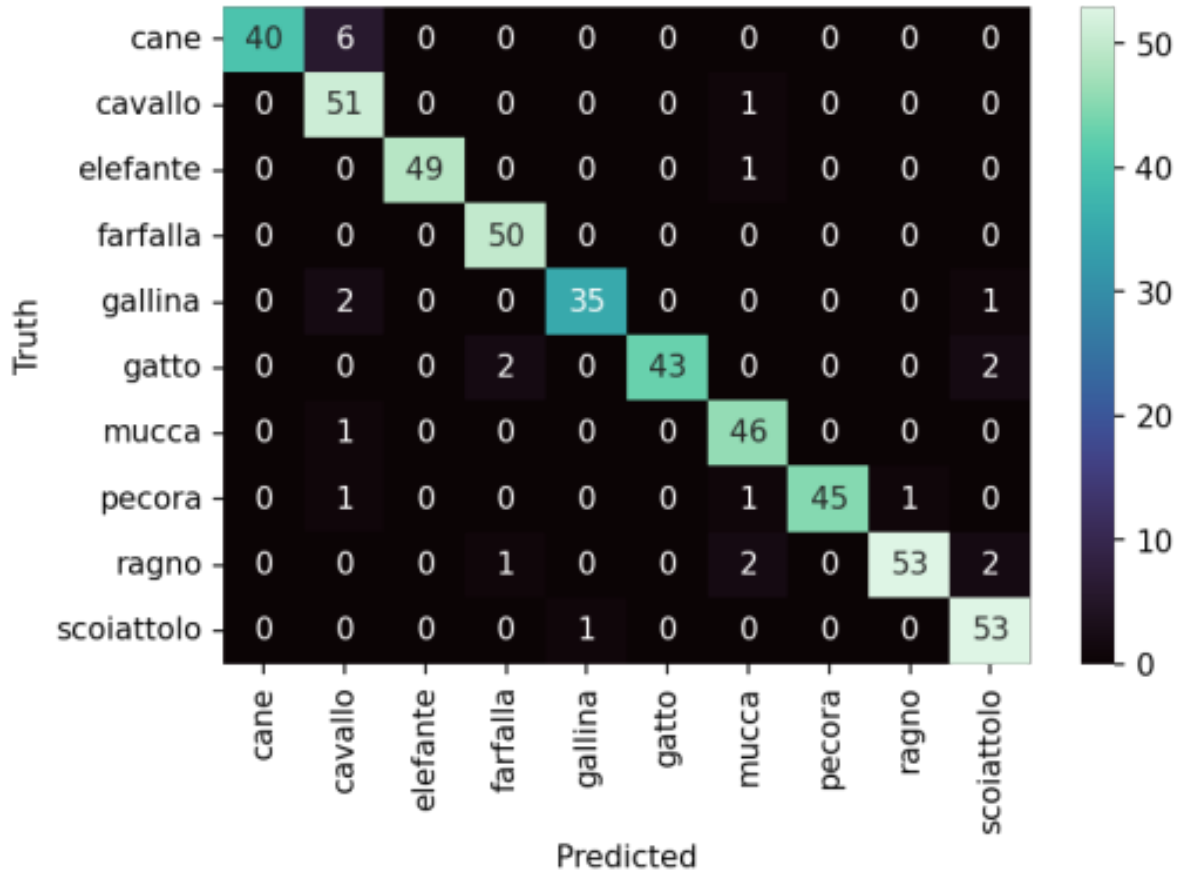


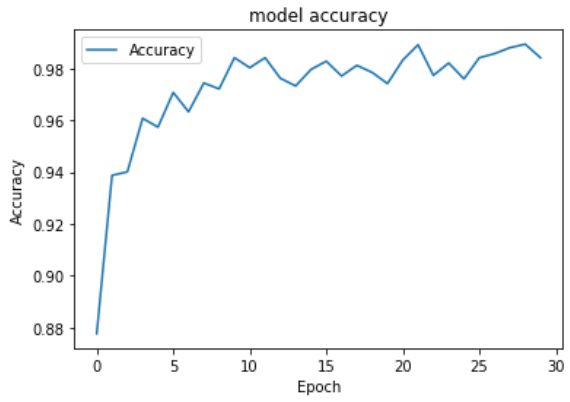
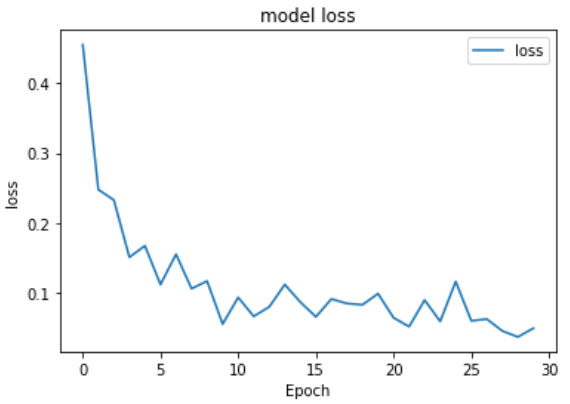
Model accuracy	Model loss
	

Dokładność na całym datasecie: 81.2% acc

EfficientNetB0

Confusion matrix



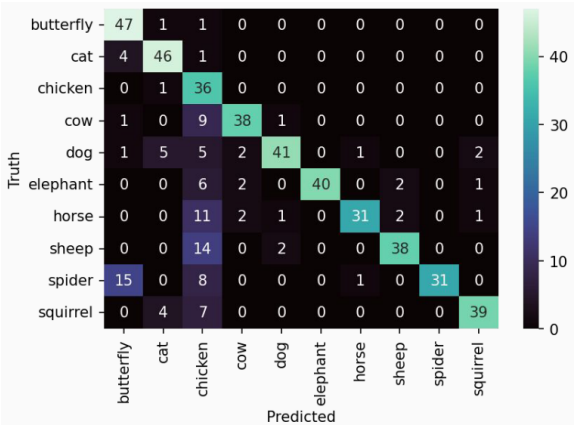
Model accuracy	Model loss
	

Dokładność na całym datasecie: 89.4% acc

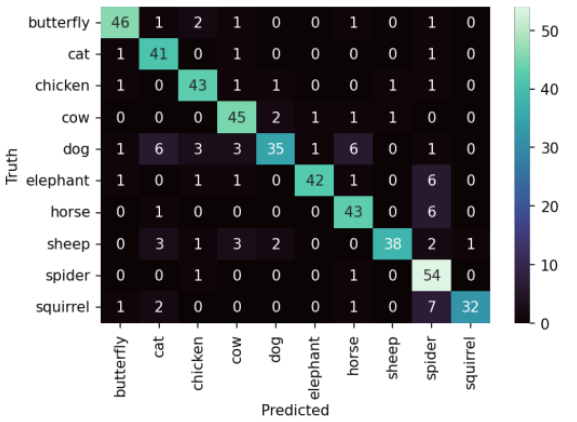
Analiza wyników

Porównanie confussion matrix

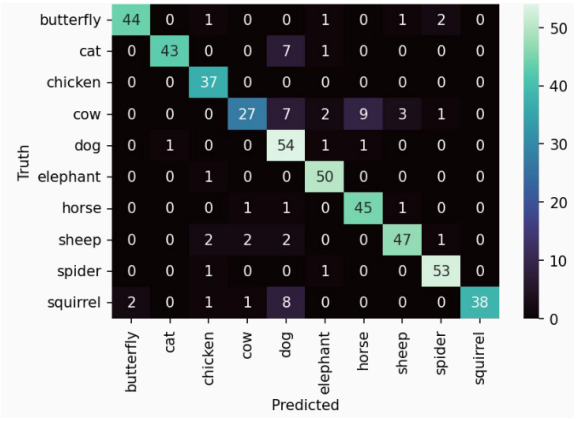
Xception - base dataset



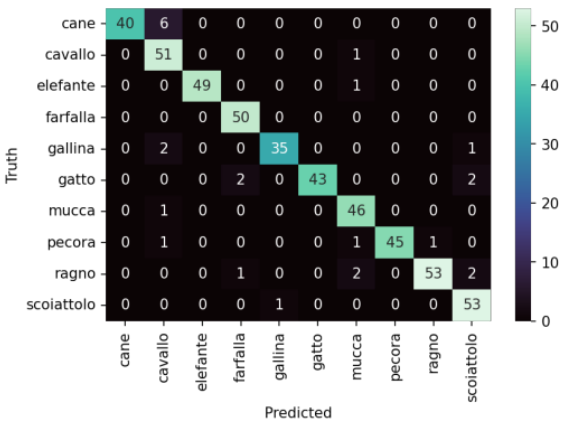
Xception - reduced dataset



EfficientNetB0 - base dataset



EfficientNetB0 - reduced dataset



Porównanie dokładności

Xception - base dataset: 71,08% acc
EfficientNetB0 - base dataset: 82,37% acc

Xception - reduced dataset: **81,2% acc**
EfficientNetB0 - reduced dataset: **89,4%**

Wnioski

Po selekcji zdjęć które były poprawnie identyfikowane przez obie sieci i ponownym treningu modeli zaobserwowaliśmy gwałtowny skok w poprawnych predykcjach, przy tym samym zbiorze walidacyjnym - całym datasecie.

Predykcje modelu Xception poprawiły się o **14%**, a predykcje EfficientNetB0 o **8.5%**.

W naszym projekcie zastosowaliśmy technikę podobną do ensemble learning, z tym wyjątkiem że nasze modele dalej posiadają bazowy rozmiar. Łączone predykcje zastosowane zostały jedynie w preprocessingu, a tak przygotowane zdjęcia poprawiły ustalenie wag przy trenowaniu.

Rozszerzenia na przyszłość

Nowa metoda znajdowania outlayerów

Sieci neuronowe starają się jak najlepiej dopasować wagi do danego datasetu, jednocześnie starając się generalizować część parametrów.
Stosując naszą metodę preprocessingu można to wykorzystać poprzez trenowanie sieci na datasecie, odrzucenie źle sklasyfikowanych przykładów i ponowny trening.

Zwiększenie liczby modeli biorących udział w redukcji datasetu

W naszym projekcie wykorzystaliśmy jedynie dwa modele.
Zwiększenie ich liczby może równocześnie zwiększyć dokładność selekcji przykładów do zredukowanego datasetu.

Bibliografia:

1. https://keras.io/api/applications/
2. https://www.tensorflow.org/resources/models-datasets
3. http://advances.utc.sk/index.php/AEEE/article/view/2202/1271
4. https://www.researchgate.net/publication/324960511_Wild_Animal_Detection_Using_Deep_Convolutional_Neural_Network