TASK 1 – CLASSIFICATION

Cel: Klasyfikacja znaków amerykańskiego języka migowego

Dane: 1200 zdjęć w 4 katalogach

Etykiety: “0”, “A”, “B”, “C”

Wykonał: Cezary Karczewski

Wczytywanie danych oraz dodawanie etykiet:

Dane zostały wczytane z katalogu plików przy pomocy bibliotek *pathlib* oraz funkcji *image_dataset_from_director.*

Normalizacja danych:

Przeskalowanie zdjęcia z zakresu [0, 255](RGB) do [0, 1] z wykorzystaniem funkcji *map* oraz *lambdy*.

Augmentacja danych:

Stworzenie powłoki, która później zostanie wykorzystana w modelu. Wykorzystanie funckji *RandomFlip* z parametrem “*horizontal*” oraz ustawieniem rozmiaru zdjęcia 300x300 i 3 kanałami.

Podział danych:

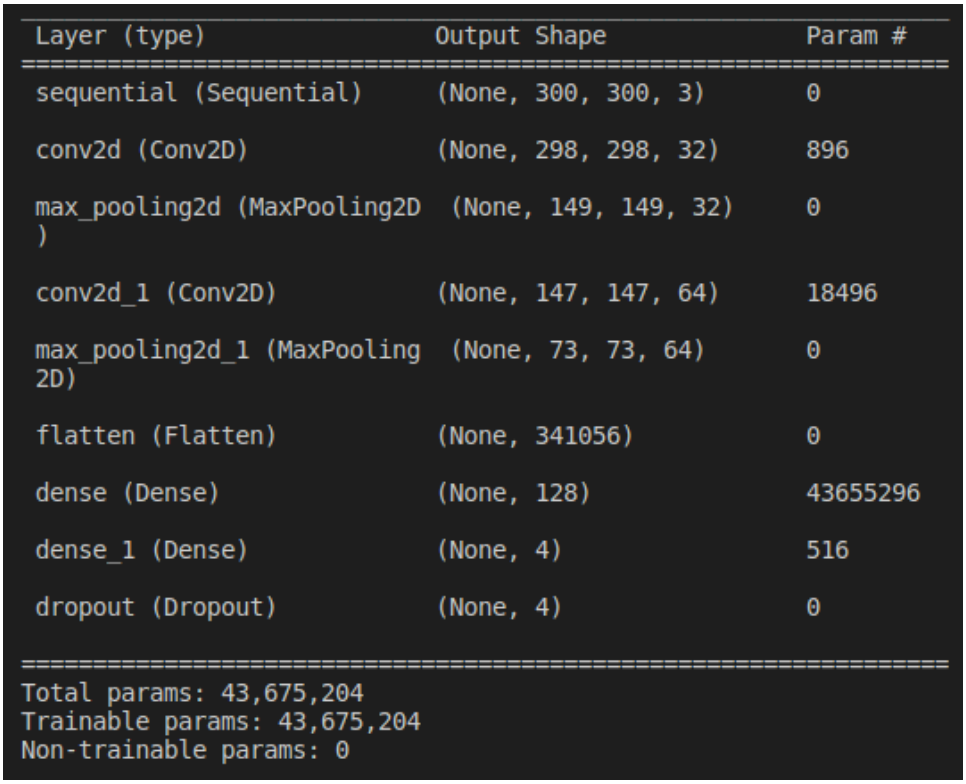
Podział na dane treningowe, walidacyjne i testowe w proporcji 8:1:1 przy pomocy własnej funkcji, która przed podziałem dokonuje operacji *shuffle*.

Model:

Wykorzystanie klasy *Swquential*. Model składa się łącznie z 9 warstw:

- data augmentation
- Conv2D z parametrami (filters 32, kernel size 3, activation “relu”)
- Conv2D z parametrami (filters 64, kernel size 3, activation “relu”)
- 2x MaxPooling2D z ustawionym domyślnie parametrem pool_size (2, 2)
- Dense z parametrami (units 128, activation=”relu”)
- Dense z parametrami (units 4)
- Dropout z parametrami (rate 0.2)

Podsumowanie modelu:



Layer (type)	Output Shape	Param #
sequential (Sequential)	(None, 300, 300, 3)	0
conv2d (Conv2D)	(None, 298, 298, 32)	896
max_pooling2d (MaxPooling2D)	(None, 149, 149, 32)	0
conv2d_1 (Conv2D)	(None, 147, 147, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 73, 73, 64)	0
flatten (Flatten)	(None, 341056)	0
dense (Dense)	(None, 128)	43655296
dense_1 (Dense)	(None, 4)	516
dropout (Dropout)	(None, 4)	0

Total params: 43,675,204
Trainable params: 43,675,204
Non-trainable params: 0

Model został skompilowany przy pomocy funckji *compile*. Wykorzystane parametry: (optimizer *RMPprop* z learning_rate ustawionym na 0.0001, loss *SparseCategoricalCrossentropy* z from_logits ustawionym na True, metrics "*accuracy*")

Przygotowanie danych przed uczeniem poprzez użycie metod *Dataset.cache* oraz *Dataset.prefetch* w celu skrócenia czasu trenowania modelu.

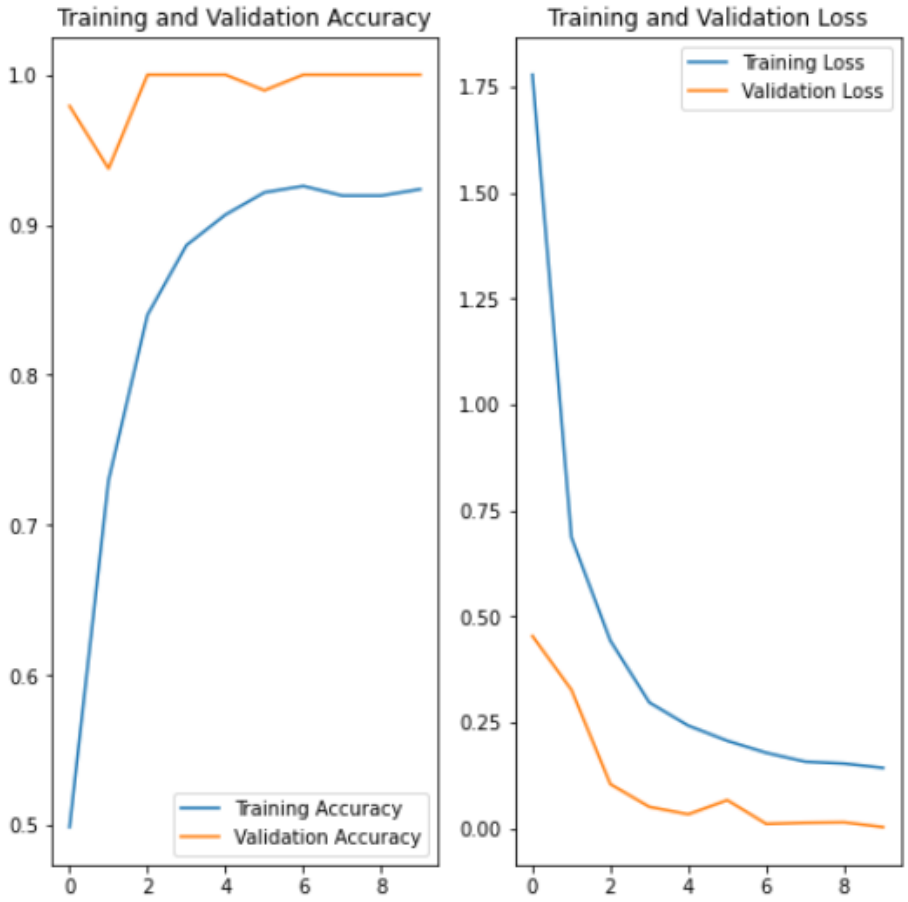
Wytrenowanie modelu przy użyciu danych treningowych oraz walidacyjnych w czasie 10 epokach.

Raport z trenowania modelu:

Epoch 1/10
30/30 [==============================] - 48s 2s/step - loss: 1.7778 - accuracy: 0.4989 - val_loss: 0.4542 - val_accuracy: 0.9792
Epoch 2/10
30/30 [==============================] - 51s 2s/step - loss: 0.6868 - accuracy: 0.7299 - val_loss: 0.3273 - val_accuracy: 0.9375
Epoch 3/10
30/30 [==============================] - 49s 2s/step - loss: 0.4432 - accuracy: 0.8400 - val_loss: 0.1049 - val_accuracy: 1.0000
Epoch 4/10
30/30 [==============================] - 51s 2s/step - loss: 0.2975 - accuracy: 0.8865 - val_loss: 0.0514 - val_accuracy: 1.0000

Epoch 5/10
30/30 [==============================] - 51s 2s/step - loss: 0.2429 - accuracy: 0.9068 - val_loss: 0.0338 - val_accuracy: 1.0000
Epoch 6/10
30/30 [==============================] - 47s 2s/step - loss: 0.2072 - accuracy: 0.9216 - val_loss: 0.0673 - val_accuracy: 0.9896
Epoch 7/10
30/30 [==============================] - 48s 2s/step - loss: 0.1788 - accuracy: 0.9258 - val_loss: 0.0113 - val_accuracy: 1.0000
Epoch 8/10
30/30 [==============================] - 46s 2s/step - loss: 0.1579 - accuracy: 0.9195 - val_loss: 0.0140 - val_accuracy: 1.0000
Epoch 9/10
30/30 [==============================] - 46s 2s/step - loss: 0.1535 - accuracy: 0.9195 - val_loss: 0.0150 - val_accuracy: 1.0000
Epoch 10/10
30/30 [==============================] - 50s 2s/step - loss: 0.1433 - accuracy: 0.9237 - val_loss: 0.0036 - val_accuracy: 1.0000

Wykresy krzywe uczenia “Accuracy” i “Loss”:



Ewaluacja oraz testowanie modelu

Testowanie modelu na zbiorze testowym oraz wyliczenie Precision, Recall, Accuracy oraz f1-score. Do wyliczenia poszczególnych wartości wykorzystałem gotowe funkcje. f1-score wyliczyłem według wzoru.

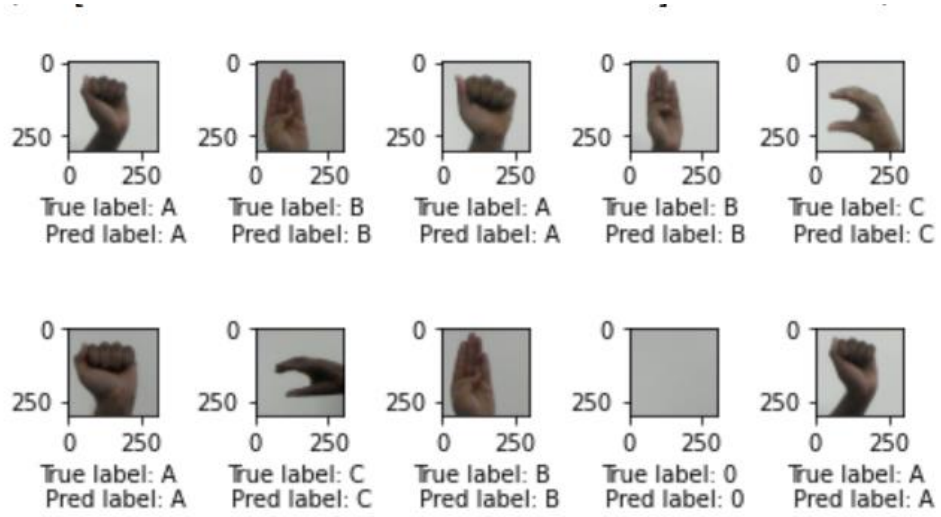
Precision: 1.0

Recall: 1.0,

Accuracy: 0.9937

f1-score: 1.0

Prezentacja 10 przykładowych wyników przewidywań wytrenowanego modeli:



Zapis modelu:

Model zostaje zapisany w jako models/imageclassifier.h5.

Opis wyników:

Na pierwszy rzut oka patrząc na wyniki miar jakości można dojść do wniosku, że otrzymaliśmy perfekcyjnie wytrenowany model, ponieważ wszystkie wartości osiągnęła wartość 1.0. Analizując krzywe uczenia zbioru treningowego również widzimy, że wartość Accuracy przy pierwszej epoce wynosiła 0.5 oraz z każdą następną epoką rosła w okolicy 6 epoki osiągnęła stagnację zatrzymując się na poziomie około 90%. Wykres straty danych treningowych również wygląda poprawnie. Najistotniejszy w przypadku tego modelu jest wykres danych walidacyjnych, który pokazuję nam, że już przy pierwszej epoce osiągnął wartość 0.97 w następnych epokach był niewielki spadek, następnie znów wzrost i od 3 epoki utrzymuje stałą wartość w okolicy 1. Można z tego wywnioskować, że nasz model jest zbyt dopasowany “overfitting”, jest to spowodowane zbyt mało różnorodną bazą danych. Wszystkie zdjęcia są do siebie bardzo podobne oraz posiadają jednolite tło, co sprawia, że model jest podatny overfiting. Jednym z rozwiązań, które można zastosować w przypadku tego modelu byłoby urozmaicenie zbioru danych poprzez dodanie zdjęć liter języka migowego w różnym otoczeniu. Można wprowadzić wykonać dodatkowe, bardziej urozmaicone metody augmentacji jak Affine Transformation lub Rotation co również urozmaici oraz poszerzy zbiór danych.