Rozponawanie sportów

Mikołaj Malek

8 maja 2024

1 Wstep

Projekt skupia sie na porównywaniu wyników dla różnych modeli które maja na celu rozpoznawanie typów sportów. Chciałbym pokazać który z modeli wykaże sie najwieksza skutecznościa przy podobnych ustawieniach i ilości danych dla tych modeli. Pokaże również czy kolor zdjeć ma wpływ na wyniki modeli.

2 Dane

2.1 Baza Danych

Baza danych została pobrana ze strony Kaggle i zawiera ponad 10 000 obiektów ale musiała zostać przeze mnie naprawiona ponieważ zawierała troche błedów. Baza danych była podzielona na foldery o nazwie sportu które zawierały ok 1000 zdjeć danego sportu.

2.2 Preprocessing

Na poczatku musiałem pousuwać wszystkie zbedne foldery, a nastepnie usunać wszystkie piki które nie były zdjeciami a było ich sporo wykorzystałem do tego własnorecznie napisana funkcje. Po usunieciu wszytskich błednych danych ustawiłem rozdzielczość zdjeć na 128x72 aby przyśpieszyć proces uczenia modeli. Podzieliłem baze na zbiór treningowy i testowy w stosunku 65/35%, a batch size ustawiłem na 64.

```
train_df, validate_df = train_test_split(df, test_size=0.35, random_state=42)
train_df = train_df.reset_index(drop=True)
validate_df = validate_df.reset_index(drop=True)
```

3 Model CNN

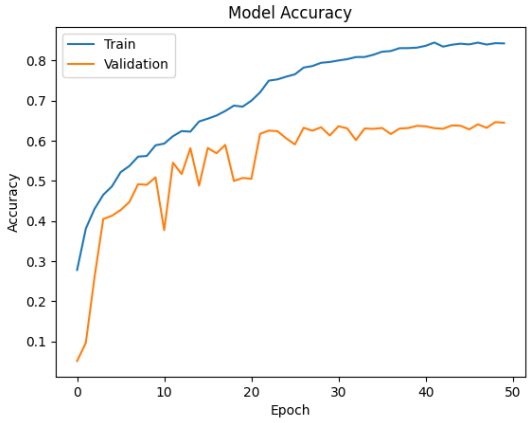
Pierwszym z modeli którym którego użyjemy jest mój własny model wytrenowany przy użyciu klasyfikatora CNN

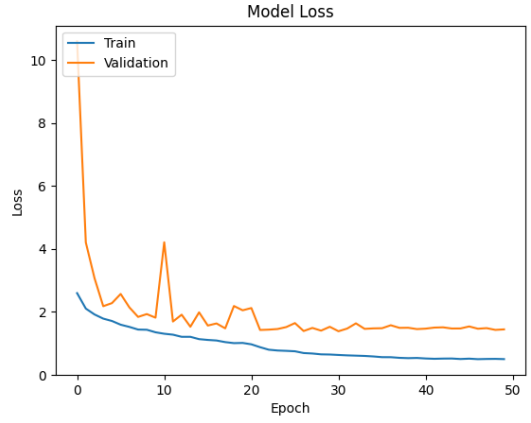
3.1 Wyniki dla zdjeć RGB

Do utworzenia tego modelu edytowałem kod z zajeć i dostosowałem do moich danych. Na poczatku dodaje warstwe konwolucyjna z 32 filtrami o rozmiarze 3x3 i aktywacja ReLU. Nastepnie dodajemy warstwe normalizacji abtcha i warstwe poolingu maksymalnego. Powatrzamy ten proces kilka razy zwiekszajac liczbe filtrów. Po warstwach konwolucyjnych dane sa spłaszczane i przekazywane do warstw gestych z 512 neuronami, aktywacja ReLU oraz warstawmi normalizacji batcha i Dropout. Na końcu dodajemy warstwe wyjściowa z funkcja aktywacji softmax. Model jest kompilowany używajac categoricalcrossentropy jako funkcji straty, optymaliztora Adam i metryki accuracy.

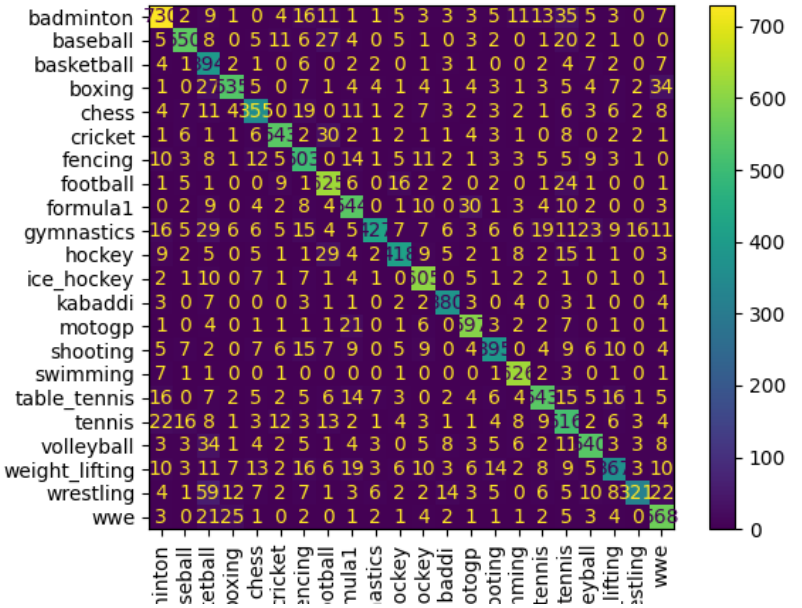
```python
model = Sequential()

model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(IMAGE_WIDTH, IMAGE_HEIGHT, IMAGE_CHANNELS)))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.2))

model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.2))

model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.2))

model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.2))
model.add(Dense(22, activation='softmax'))

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

model.summary()
```

Wynik dla tego modelu prezentuja sie nastepujaco:





Skuteczność dla danych treningowych wyniosła ok. 80% a dla danych testowych ok. 64%. Czas treningu takiefo modelu to ok 1100 sekund przy 50 epokach czyli 22 sekundy na epoke.



Matriks konfuzji pokazuje nam że model najczesciej mylił baseball z wrestlingiem oraz tenis z badmintonem. Widać również na nim ze nasz model poradził sobie w miare dobrze ztymi danymi.

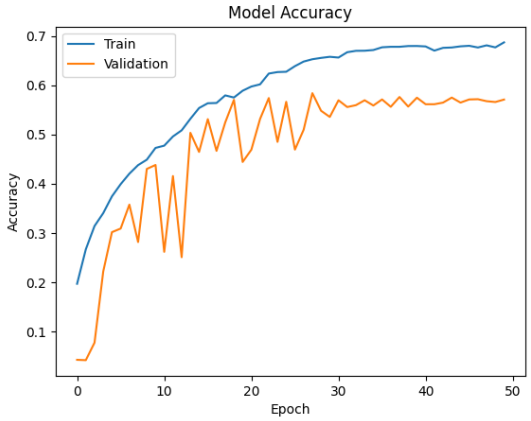
Tutaj przykładowa predykcja dla tego modelu:

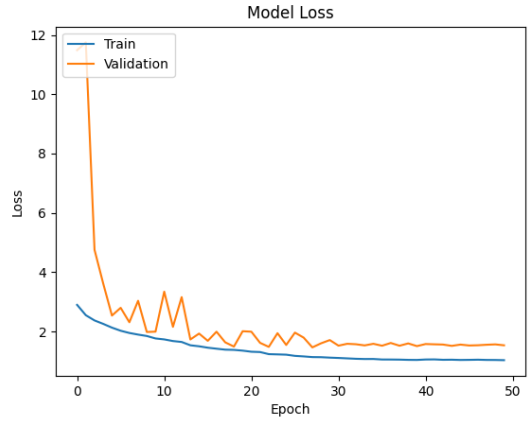
3.2 Wyniki dla zdjeć w skali szarości

Napisałem funkcje która zmienia zdjecia na skale szarości a nastepnie wybrałem te zdjecia i użyłem ich do nauki modelu bazujacego na tym samym klasyfikatorze lecz z jednym kanałem zamiast 3 kanałów.

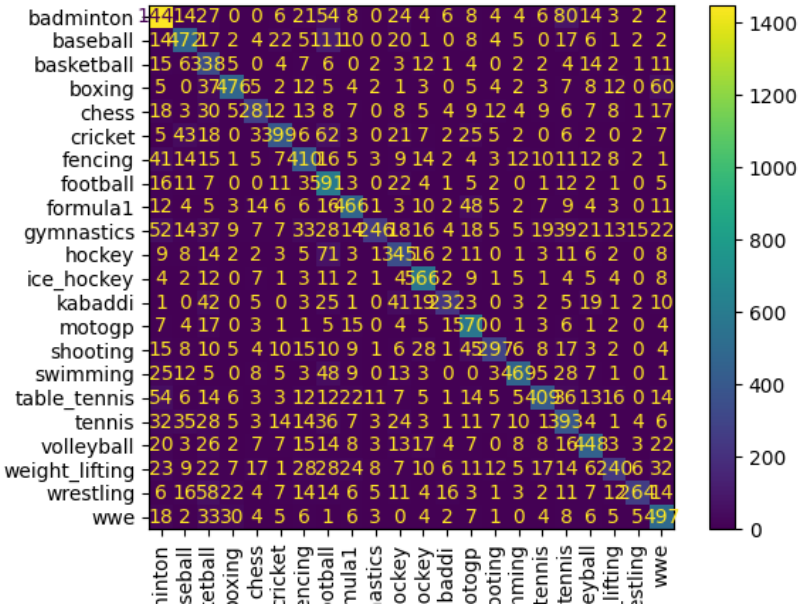
```python
def convert_images_to_grayscale(folder_path):

    items = os.listdir(folder_path)
    for item in items:
            item_path = os.path.join(folder_path, item)

            if os.path.isdir(item_path):
                convert_images_to_grayscale(item_path)
            elif os.path.isdir(folder_path):
                if item.endswith(".jpg") or item.endswith(".png"):
                    img = cv2.imread(item_path)


                    gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

                    output_path = os.path.join(folder_path, f"gray_{item}")
                    print(output_path)
                    cv2.imwrite(output_path, gray_img)
```

Wyniki tego modelu nie sa aż tak satysakcjonujace jak poprzednich modeli a prezentuja sie nastepujaco:



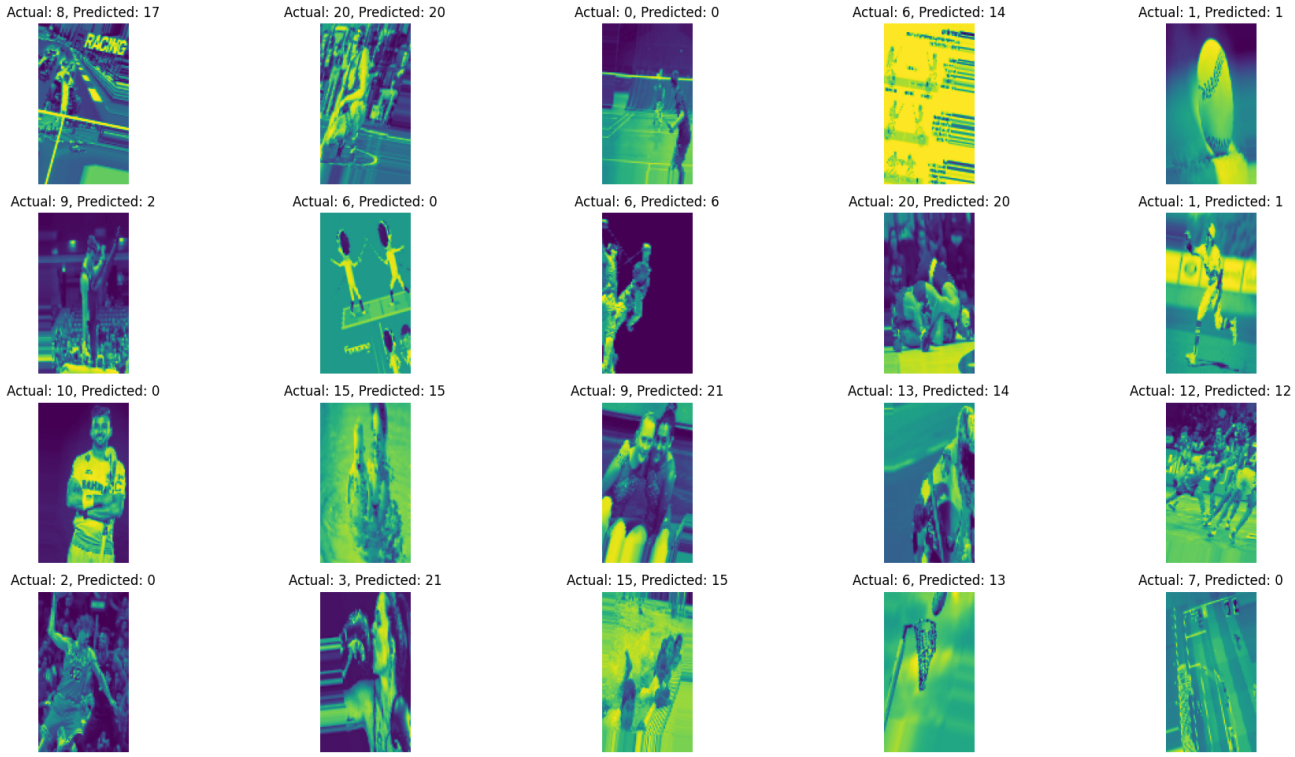


Skuteczność dla danych treningowych wyniosła ok. 70% a dla danych testowych ok. 55%. Czas treningu takiefo modelu to ok 750 sekund przy 50 epokach czyli 15 sekundy na epoke.



Matriks konfuzji pokazuje nam że model radzi sobie w miare z danymi i nie popełnia zbyt dużo błedów. Najcześćiej myli inne sporty z badmintonem oraz inne sporty z hokejem na lodzie.

Tutaj przykładowa predykcja dla tego modelu:



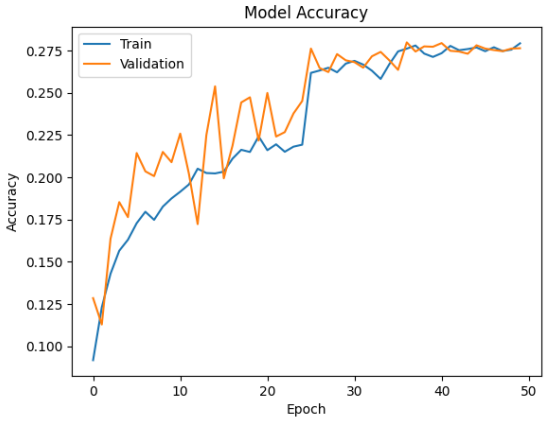
4 Pre-trained ResNet50

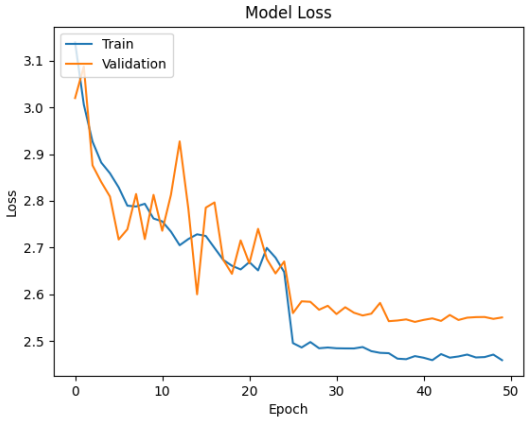
Użyłem tutaj gotowego modelu ResNet50 aby douczyć go na moich danych i porównać wynik z wcześniejszymi modelami.

Po implementacji modelu zamrażamy wagi dla każdej z warstw nastepnie dodajemy 3 kolejne warstwy do modelu GlobalAveragePooling2D, Flatten przekształca dane na wektor oraz warstwa Dense z 22 neuronami i funkcja aktywacji softmax generuje prawdopodobieństwa dla każdej klasy.

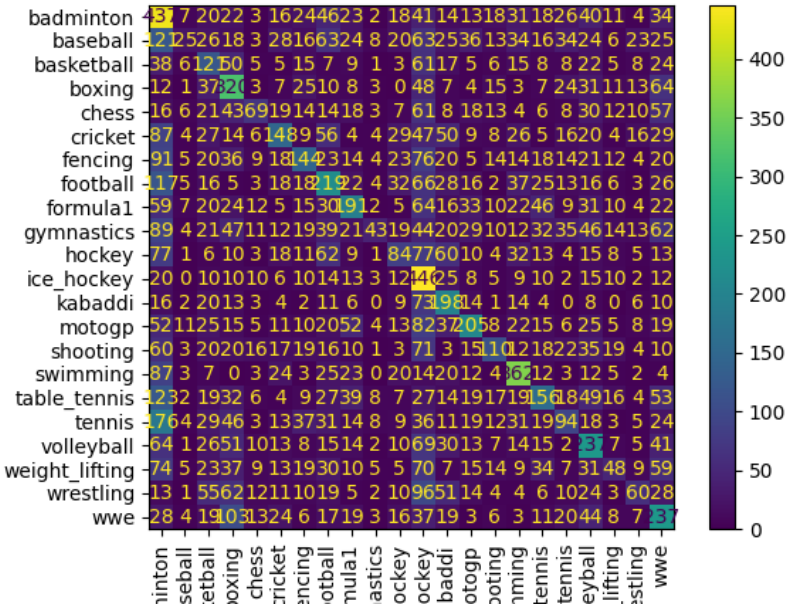
```
base_model = ResNet50(weights='imagenet', include_top=False, input_shape=(IMAGE_WIDTH, IMAGE_HEIGHT, IMAGE_CHANNELS))

for layer in base_model.layers:
    layer.trainable = False

global_average_layer = GlobalAveragePooling2D()(base_model.output)
PlusFlatten = Flatten()(global_average_layer)
predictionLayer = Dense(22, activation='softmax')(PlusFlatten)

model = Model(inputs=base_model.input, outputs=predictionLayer)


model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.01), loss='categorical_crossentropy', metrics=['accuracy'])
```

Wyniki tego modelu nie sa aż tak satysakcjonujace jak poprzednich modeli a prezentuja sie nastepujaco:





Skuteczność dla danych treningowych wyniosła ok. 27,5% a dla danych testowych ok. 27,5%. Czas treningu takiefo modelu to ok 1350 sekund przy 50 epokach czyli 27 sekundy na epoke.



Matriks konfuzji pokazuje nam że model słabo radzi sobie z naszymi danymi i jest słabo wyuczony ma dużo pomyłek i mała skuteczność

Tutaj przykładowa predykcja dla tego modelu:

5 Podsumowanie

Najlepszym klasyfikatorem był klasyfikator CNN dla zdjeć RGB. Dobrze sobie radził z danymi i stosunkowo szybko. Nastepny był klasyfikator CNN dla zdjeć w skali szarości. Radził sobie troche gorzej

z danymi lecz był najszybszym ze wszytskich klasyfikatorów wiec jeśli zależy nam na czasie a nie na sktuteczności można rozważyć wybór takiej opcji. Najgorszym ze wszystkich był pre-trained ResNet50 jego wynik bardzo odbiegały od dwóch poprzednich klasyfikatorów oraz zajmował najwiecej czasu ze wszytskich.

Literatura

StackOverflow
Kaggle
ChatGpt
Materiały z zajeć