Report

Abdurakhim Bakytzhan

Astana IT University

IT-2106

Advanced Programming

Yeleu Sultanmurat

Astana, 2023

Github link: https://github.com/aBacoding/PlayersClassification.git

YouTube video link: https://youtu.be/YqCU-kOthqo

Introduction

- **Problem:** The main problem was creating a web application about the CS:GO gaming news website with built-in image classification for face recognition of 5-star players: device, electronic, NiKo, s1mple, ZywOo.

- **Literature review with links:**

  1. https://www.youtube.com/watch?v=uqomO_BZ44g

  In this video, the author makes a classification between emotions (happy and sad). By this video I understood how to work with my own created dataset and make predictions.

  2. https://www.youtube.com/watch?v=m1dQ38qDABw

  3. https://www.youtube.com/watch?v=kwKfWBb6frs

  Thanks to these videos, I understood how to collect images for a dataset and how to clean them.

  4. https://www.tensorflow.org/tutorials/images/cnn?hl=ru

  In the official Tensorflow website above, I learned how to use CNN (Convolutional layers) correctly in the model.

  5. https://medium.com/ymedialabs-innovation/data-augmentation-techniques-in-cnn-using-tensorflow-371ae43d5be9

  This article above contains all the important information about data augmentation usage. After reading them, I realized how data augmentation can affect photos in the training set

- **Current work (description of the work):**

  My current work consisted of:

  1) Creating the frontend part of the project (HTML/CSS/Bootstrap/JS)

  2) Creating the backend part of the project (Node.js/Express.js)

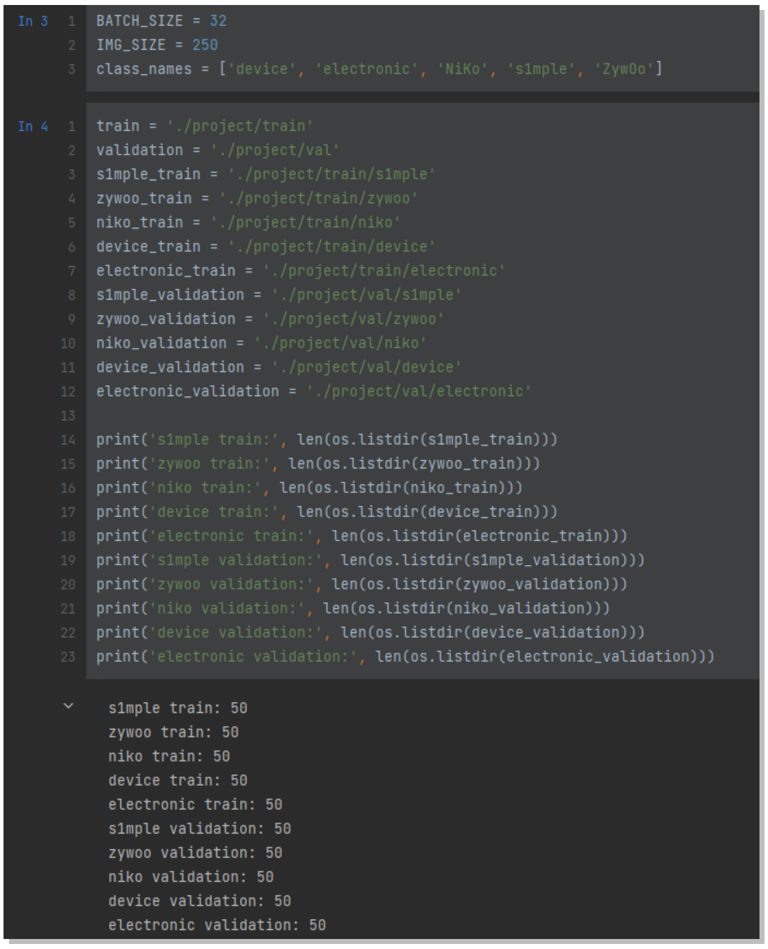
3) Creating the main part of a machine learning project (create, train and save a model using Tensorflow, then convert this my_model.h5 file into model.json to use Tensorflow.js on the website)

Data and Methods

- **Information about the data (probably analysis of the data with some visualization):**

  Link to my dataset for preview: https://drive.google.com/drive/folders/1Bq9DUWtPcT2VREyb-Nj7CRe2lDByjrGy?usp=share_link

  To train the model, I created my dataset with an image of 5 players (100 images for each, 50 for training and 50 for validation) (pic. 1). Thanks to the website hltv.org I downloaded most of the images and cleaned them (cropped the faces from the image, tried to take only high-quality images), and thanks to google images, I supplemented my dataset with different photos so that they would not be duplicated.



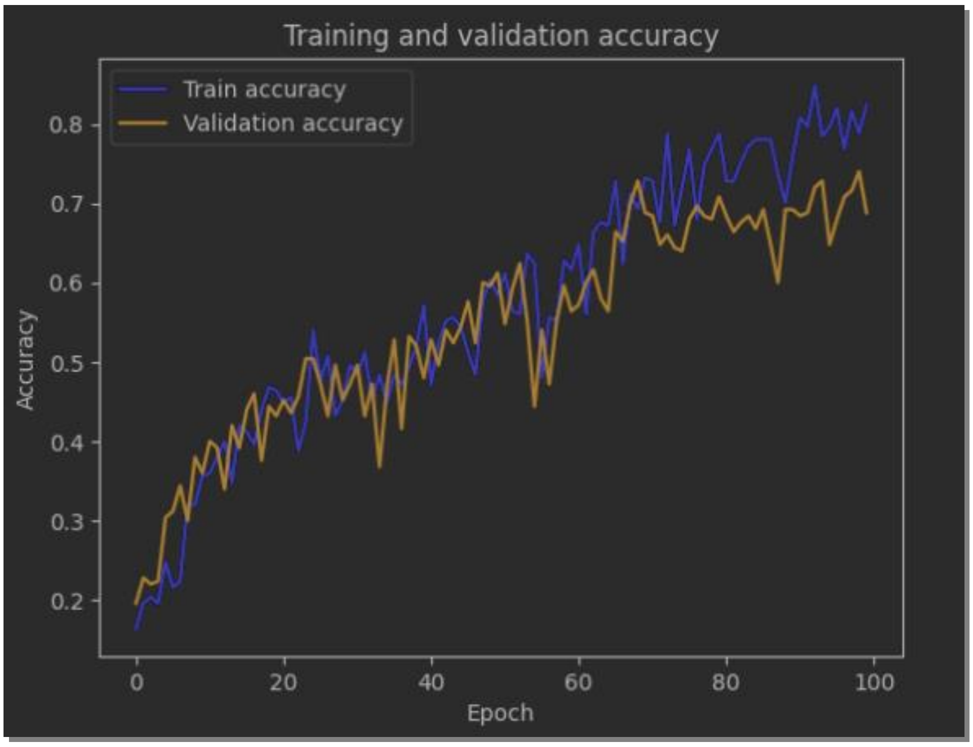
```
In 3
BATCH_SIZE = 32
IMG_SIZE = 250
class_names = ['device', 'electronic', 'NiKo', 's1mple', 'ZywOo']
```

```
In 4
train = './project/train'
validation = './project/val'
s1mple_train = './project/train/s1mple'
zywoo_train = './project/train/zywoo'
niko_train = './project/train/niko'
device_train = './project/train/device'
electronic_train = './project/train/electronic'
s1mple_validation = './project/val/s1mple'
zywoo_validation = './project/val/zywoo'
niko_validation = './project/val/niko'
device_validation = './project/val/device'
electronic_validation = './project/val/electronic'

print('s1mple train:', len(os.listdir(s1mple_train)))
print('zywoo train:', len(os.listdir(zywoo_train)))
print('niko train:', len(os.listdir(niko_train)))
print('device train:', len(os.listdir(device_train)))
print('electronic train:', len(os.listdir(electronic_train)))
print('s1mple validation:', len(os.listdir(s1mple_validation)))
print('zywoo validation:', len(os.listdir(zywoo_validation)))
print('niko validation:', len(os.listdir(niko_validation)))
print('device validation:', len(os.listdir(device_validation)))
print('electronic validation:', len(os.listdir(electronic_validation)))
```

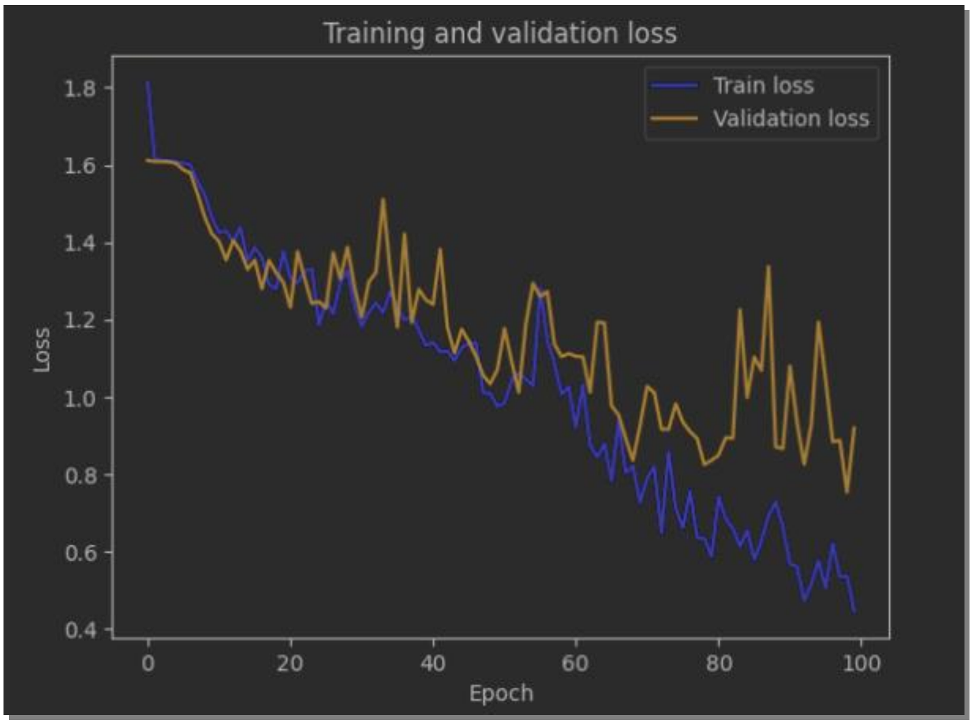
```
s1mple train: 50
zywoo train: 50
niko train: 50
device train: 50
electronic train: 50
s1mple validation: 50
zywoo validation: 50
niko validation: 50
device validation: 50
electronic validation: 50
```

Pic. 1. Setting the image size to 250 and batch size to 32, also creating my own dataset of 500 images

After I trained the model, I displayed accuracy (pic. 2) and loss (pic. 3) graphs for a clear example of how data augmentation helps to prevent overfitting.

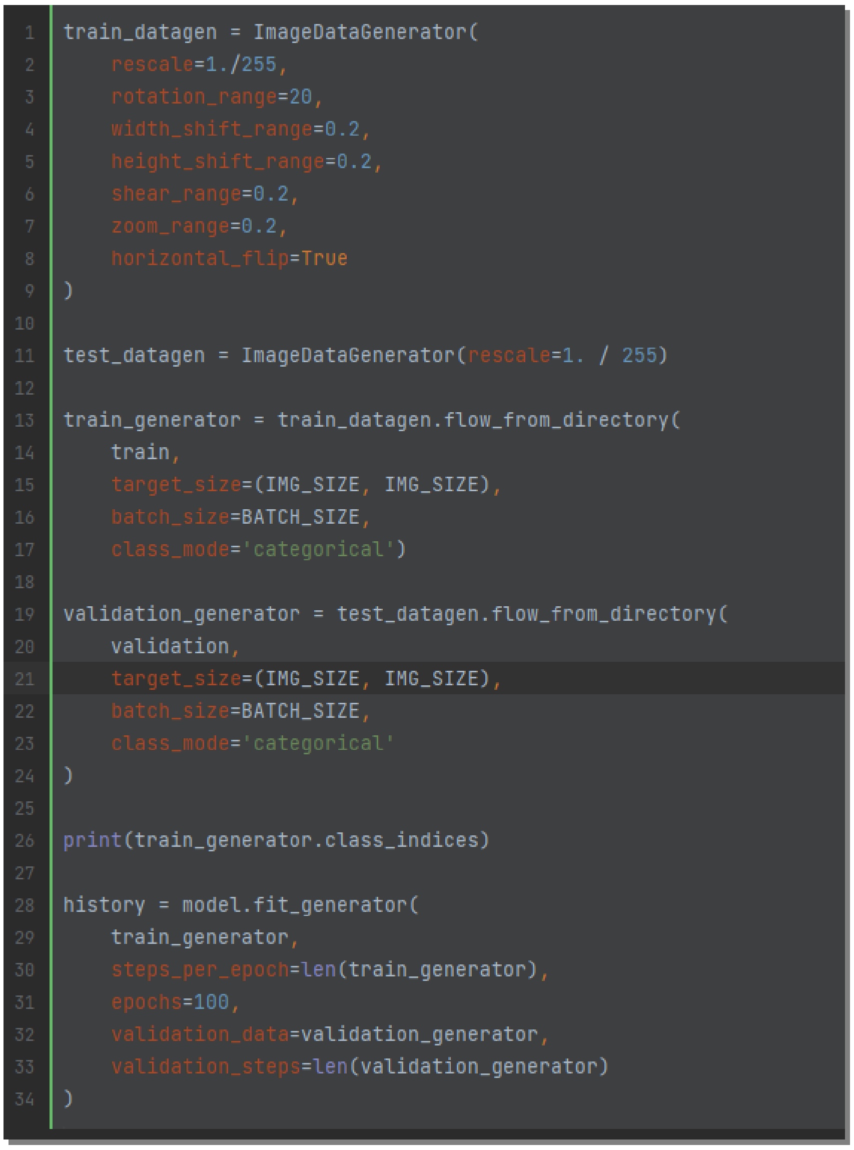


Pic. 2. Train accuracy and validation accuracy of the model



Pic. 3. Train loss and validation loss of the model

I also want to mention that I used data augmentation in order to prevent overfitting, because I really don't have a huge amount of images in the dataset (pic. 4).

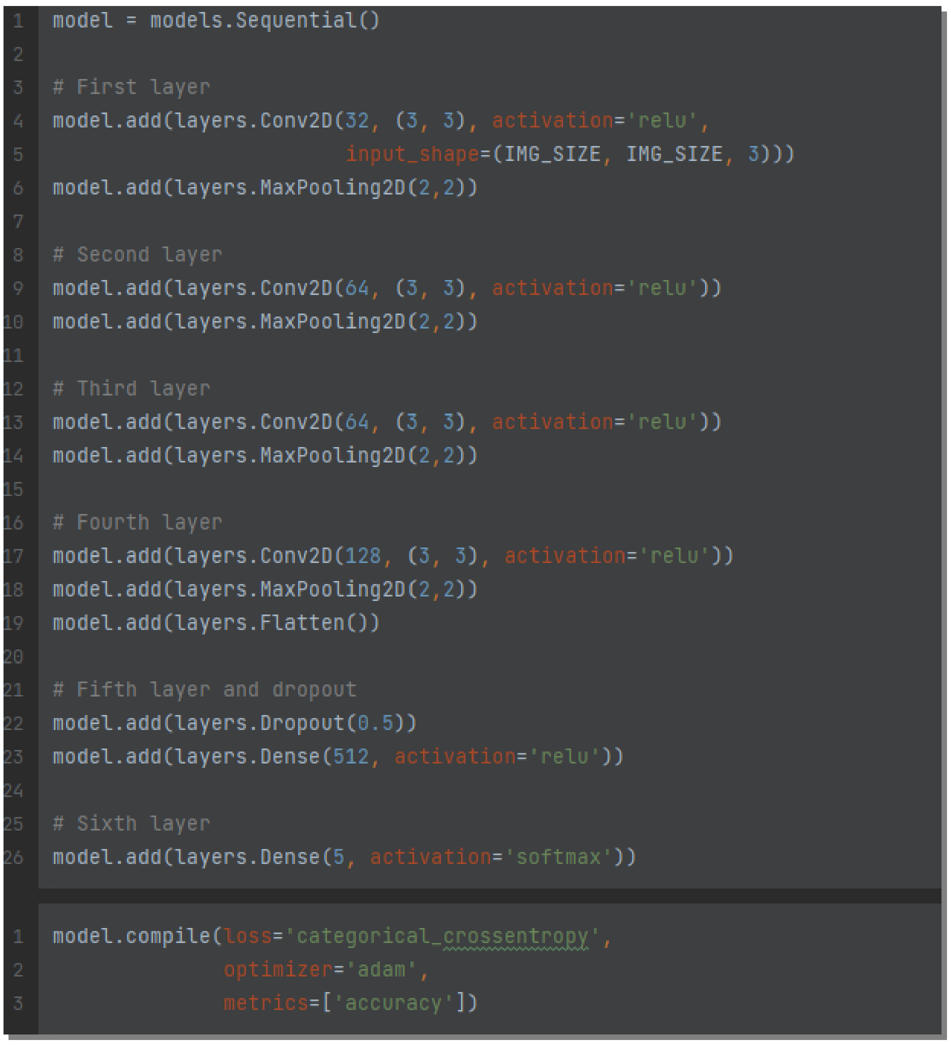


```python
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True
)

test_datagen = ImageDataGenerator(rescale=1. / 255)

train_generator = train_datagen.flow_from_directory(
    train,
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='categorical')

validation_generator = test_datagen.flow_from_directory(
    validation,
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='categorical'
)

print(train_generator.class_indices)

history = model.fit_generator(
    train_generator,
    steps_per_epoch=len(train_generator),
    epochs=100,
    validation_data=validation_generator,
    validation_steps=len(validation_generator)
)
```

Pic. 4. Data augmentation

- **Description of the ML models you used with some theory**

I used a sequential model with 4 Convolutional layers, Flatten layer, dropout to prevent overfitting, fully connected dense layer and output dense layer (pic. 5).



```python
model = models.Sequential()

# First layer
model.add(layers.Conv2D(32, (3, 3), activation='relu',
                        input_shape=(IMG_SIZE, IMG_SIZE, 3)))
model.add(layers.MaxPooling2D(2,2))

# Second layer
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D(2,2))

# Third layer
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D(2,2))

# Fourth layer
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D(2,2))
model.add(layers.Flatten())

# Fifth layer and dropout
model.add(layers.Dropout(0.5))
model.add(layers.Dense(512, activation='relu'))

# Sixth layer
model.add(layers.Dense(5, activation='softmax'))
```

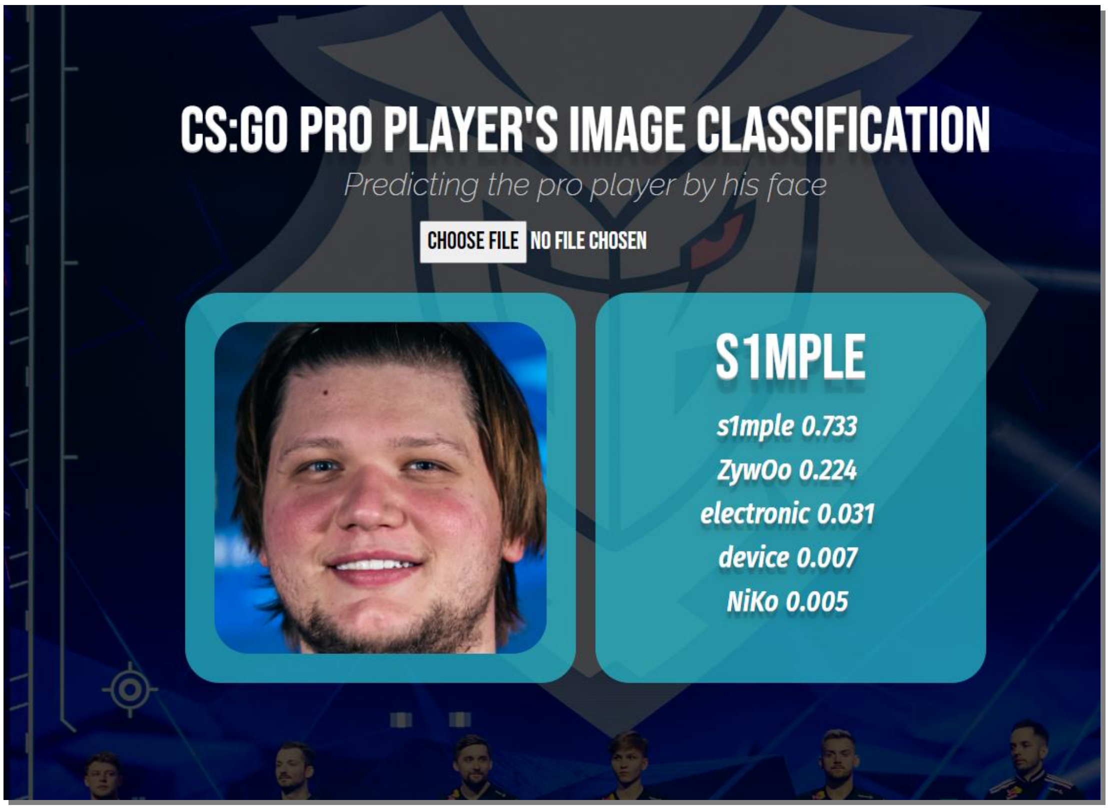
```python
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

Pic. 5. Sequential model in my project

As I mentioned in my project idea proposal, I will take the one image of the professional CS:GO player and will recognize the image by using CNN architecture. For do that, the image will be the input layer, then pass by the convolutional and pooling layers. This stage is called – Feature Extraction. Finally, it will pass from the fully connected layers and output the results of predictions. This stage is called – Classification. The all hidden layers are "ReLU" activation function and the output layer is "softmax" instead of "sigmoid" because there are more than 2 classes. Also I mentioned about dropout technique that randomly drops neurons for prevent overfitting.

Results

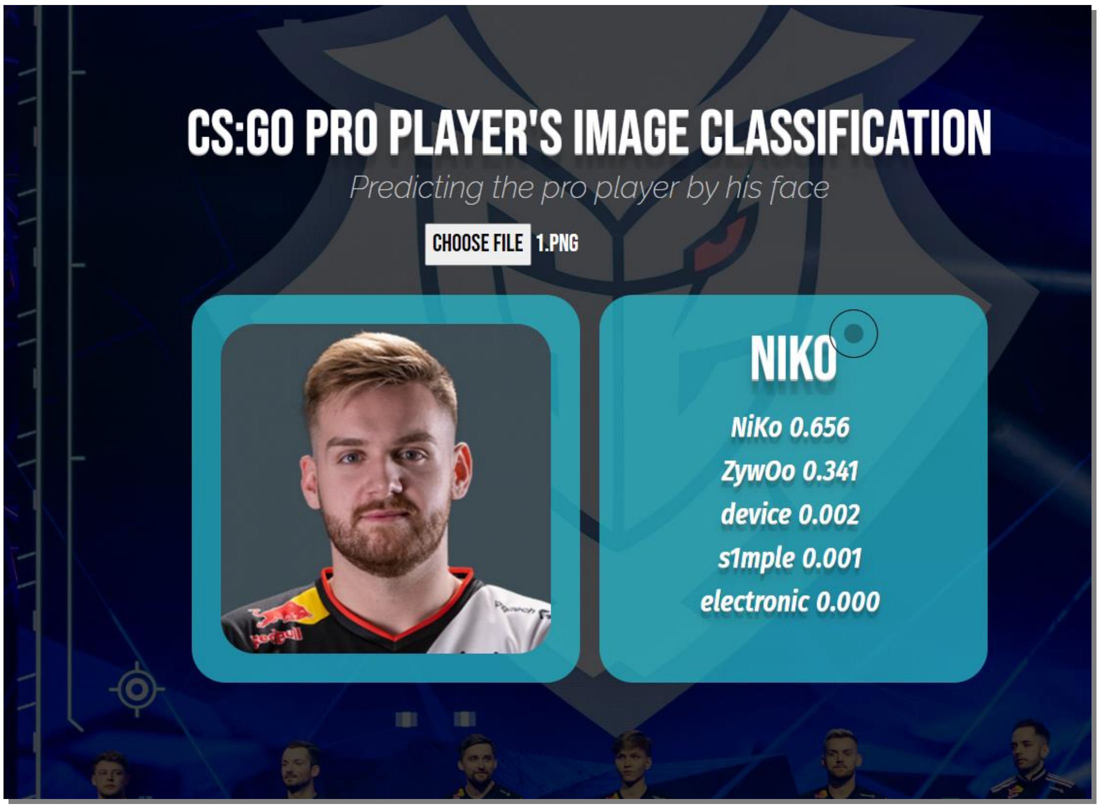
As a result I would like to show how the model works on my website that I developed for this project. The classes have adopted indexes: {0: 'device', 1: 'electronic', 2: 'NiKo', 3: 's1mple', 4: 'ZywOo'}. Referring to this information, the prediction of the "s1mple" (pic. 6), the prediction of the "NiKo" (pic. 7), the prediction of the “device” (pic. 8), the prediction of the “ZywOo” (pic. 9), the prediction of the “electronic” (pic. 10).


CS:GO PRO PLAYER'S IMAGE CLASSIFICATION
Predicting the pro player by his face
CHOOSE FILE
NO FILE CHOSEN

S1MPLE

s1mple 0.733
ZywOo 0.224
electronic 0.031
device 0.007
NiKo 0.005

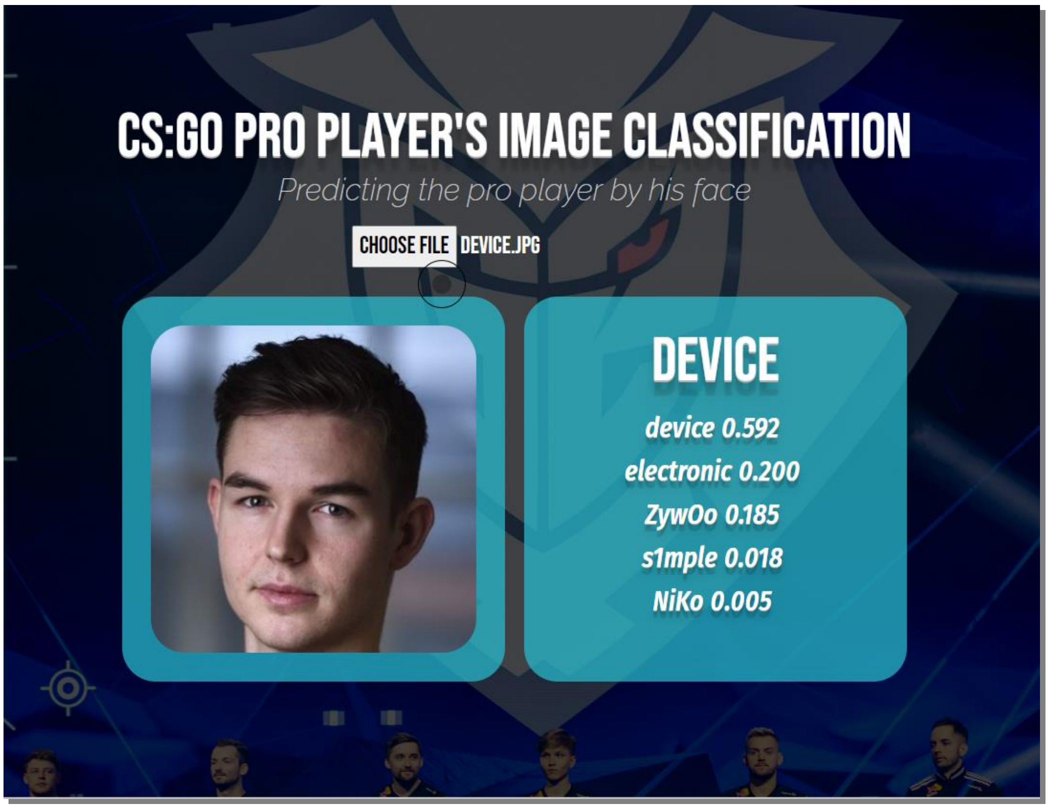
Pic. 6. Classification by the “s1mple” face


CS:GO PRO PLAYER'S IMAGE CLASSIFICATION
Predicting the pro player by his face
CHOOSE FILE
1.PNG

NIKO

NiKo 0.656
ZywOo 0.341
device 0.002
s1mple 0.001
electronic 0.000

Pic. 7. Classification by the “NiKo” face

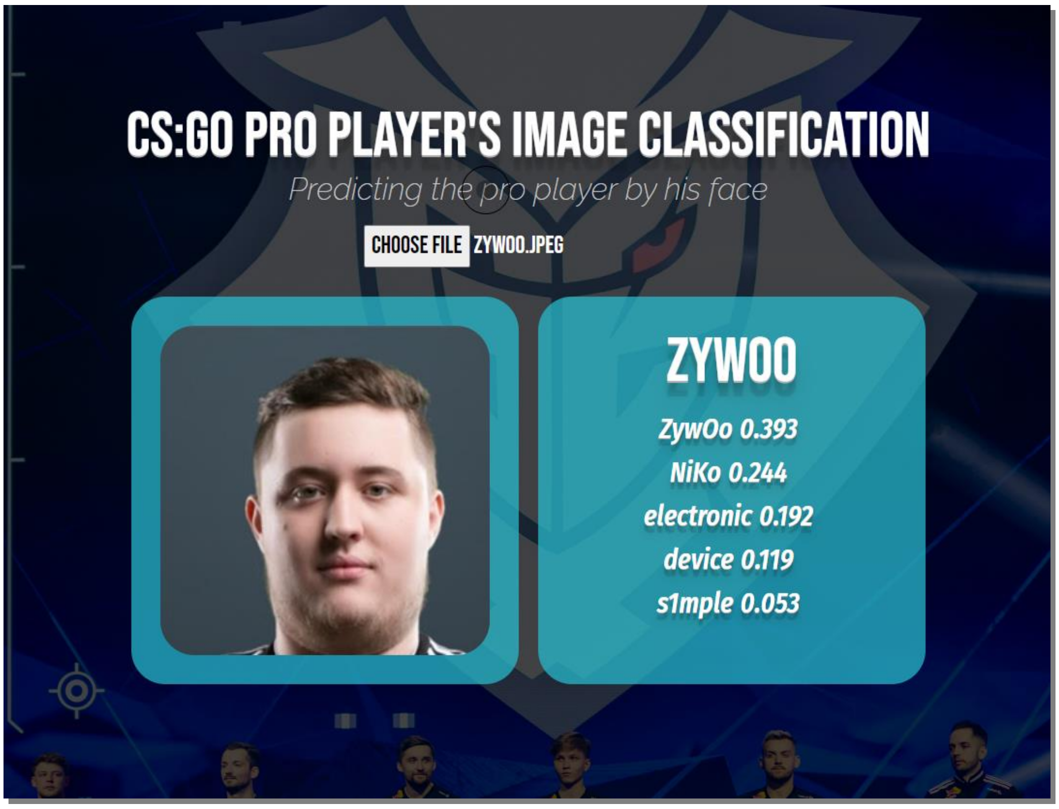


CS:GO PRO PLAYER'S IMAGE CLASSIFICATION
Predicting the pro player by his face
CHOOSE FILE
DEVICE.JPG

DEVICE

device 0.592
electronic 0.200
ZywOo 0.185
s1mple 0.018
NiKo 0.005

Pic. 8. Classification by the "device" face

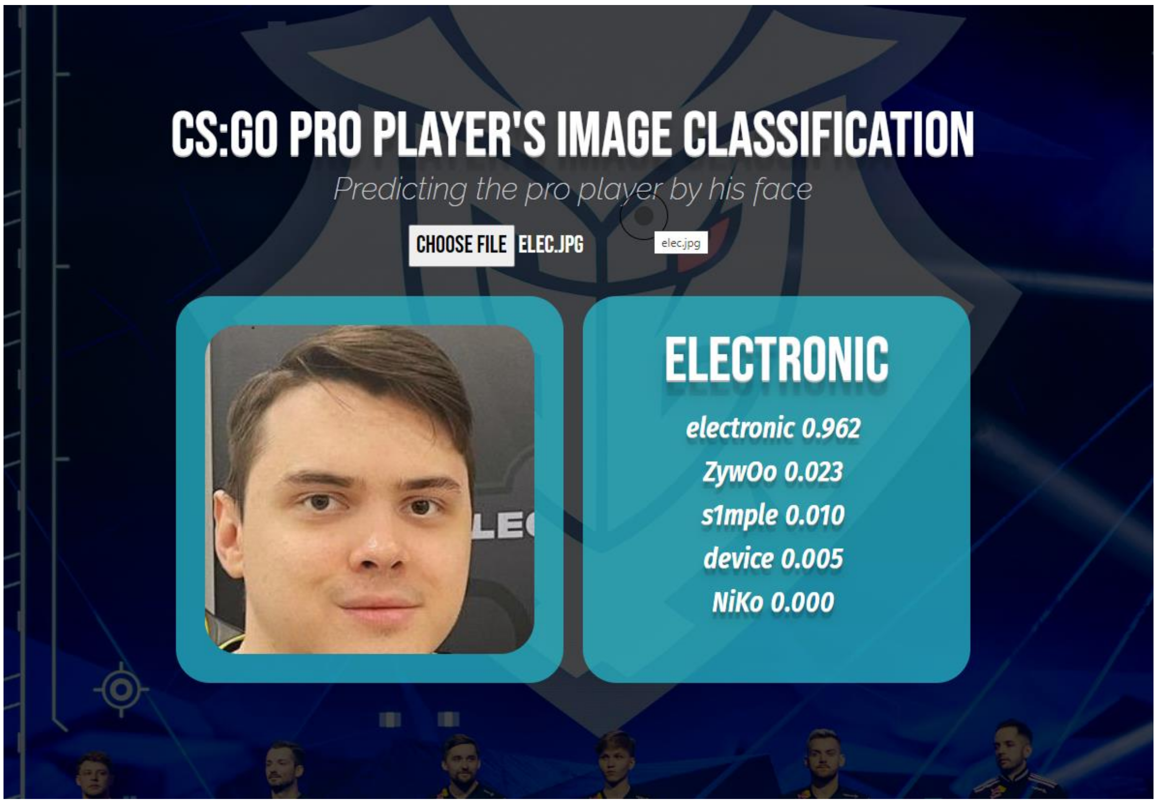


CS:GO PRO PLAYER'S IMAGE CLASSIFICATION
Predicting the pro player by his face
CHOOSE FILE
ZYWOO.JPEG

ZYWOO

ZywOo 0.393
NiKo 0.244
electronic 0.192
device 0.119
s1mple 0.053

Pic. 9. Classification by the "ZywOo" face


CS:GO PRO PLAYER'S IMAGE CLASSIFICATION
Predicting the pro player by his face
CHOOSE FILE
ELEC.JPG
elec.jpg
ELECTRONIC

```
electronic 0.962
ZywOo 0.023
s1mple 0.010
device 0.005
NiKo 0.000
```

Pic. 10. Classification by the "electronic" face

Discussion

In general, I am very pleased with the result for the first time working with machine learning. It was a huge and at the same time interesting experience that will give me the confidence to continue working on the project. In the future, I want to increase the number of players to 10, then to 20 and gradually expand the amount of data for the development of the project as one of my successful ones in the portfolio.

Sources

YouTube videos:

1. https://www.youtube.com/watch?v=uqomO_BZ44g
2. https://www.youtube.com/watch?v=m1dQ38qDABw
3. https://www.youtube.com/watch?v=kwKfWBb6frs

Official Tensorflow website:

4. https://www.tensorflow.org/tutorials/images/cnn?hl=ru
5. https://www.tensorflow.org/js/tutorials/conversion/import_keras?hl=ru

Medium forum:

6. https://medium.com/ymedialabs-innovation/data-augmentation-techniques-in-cnn-using-tensorflow-371ae43d5be9

HLTV (Half-life TV) official website:

7. https://www.hltv.org/galleries

Google Images:

8. https://images.google.com/

Starter code from the Github:

9. https://github.com/tensorflow/tfjs-examples/tree/master/mobilenet

10. https://github.com/fchollet/deep-learning-with-python-notebooks/blob/master/first_edition/5.2-using-convnets-with-small-datasets.ipynb

Starter code from the Google Drive:

11. https://drive.google.com/drive/folders/1g8R9Ca4vnQuiEZrO4j6W7wnDkpKn0Srx