FACE RECOGNITION AND CLASSIFICATION IN SKYRIM

Jovan Svorcan, Marko Mijatović

[1]Department of SIIT, Research Mentors: Dragan Vidaković, Milica Škipina

Faculty of Technical Science

Introduction

The main function of this project is to be able to predict the race of a video game character based on a picture of said character. Unlike in real life, this particular game has 10 different races its characters can have.

A list of representatives of each race in the game

Reason for changing the method given in our issue

We were originally going to use to detect and label faces using haar cascade detection but ended up using Labellmg to custom detect faces as we were having problems finding faces of some of the more fantasy-based races in the game.

Data acquirement

We collected all the pictures used for training and testing ourselves. The labels are created in xml formats, which are then all converted into one csv file that the rest the program works with.

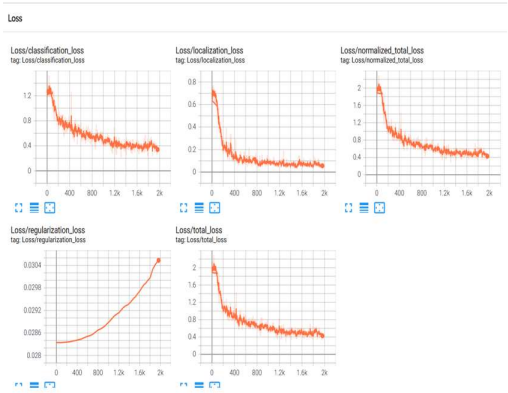
Methods

We installed Tensorflow Object Detection API and Labellmg for custom detection. After getting all labels from the train set and assigning them id numbers, the model training is then tracked on localhost:6006. We continue to train the model until it reaches a satisfying loss. The existing model we changed to fit our project is 'efficientdet_d0_coco17_tpu-32'. Finally we evaluate the trained model on test images.

Haar cascade detection (left) vs Labellmg detection (right)

Results

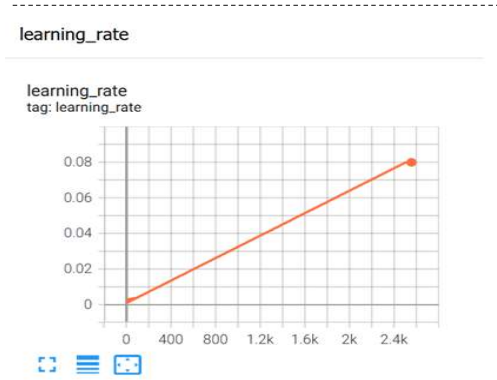
We trained our model in about 2000 steps and it took approximately 10 hours to finish. There were a lot of false-positives, which we solved by only taking the ones with the highest probability. Face detection was successful, but the classification did not show the best results due to similarities between some of races in the game.



Graphs tracked on localhost:6006

Conclusions

The main reason for the performance issues of the classification part was our lack of experience, since this was our first time dealing with such a problem. The learning rate should not have been linear, but rather should have started decreasing over time. Also, the images we chose weren't the best representatives of the game's races. We realized our mistakes too late since the model had already been exported to a graph and all of its parameters were frozen by that point.



Linear learning rate

Source:

TensorFlow Object Detection with Tensorflow 2: Creating a custom model: https://gilberttanner.com/blog/tensorflow-object-detection-with-tensorflow-2-creating-a-custom-model?fbclid=IwAR0zfqXknSKwU4WAnw5LUgtKgvecDwU8e2QSCMHkYfeT6STJ9UIDMDh5RYc