

**CSCI 4734** | Machine Learning

2019 Spring

Aytan Alakbarova, Aytan Babayeva, Nijat Mursali

Final Report

Baku 2019

**Motivation**

Object recognition is defined as computer vision technique for identifying objects in images or videos. It is used as a key output for deep or machine learning algorithms. When human beings look at a picture or watch a video, they can readily detect people or objects or other visual details. The aim of object recognition to implement this ability to a computer in a way that it can digitally understand image and its details.

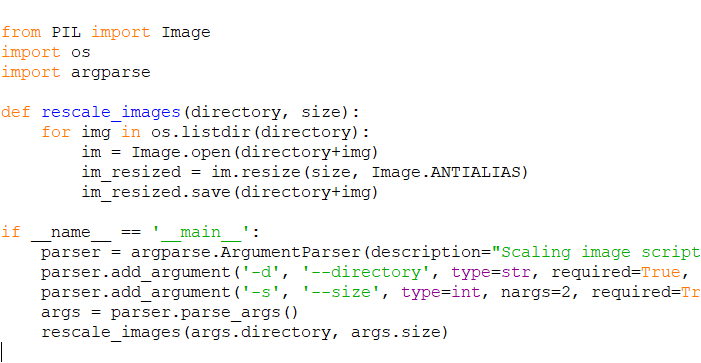
To implement object recognition utilizing a standard machine learning approach, beginning with a collection of images (or video) and choosing the applicable features in each image is required. For instance, a feature extraction algorithm might select edge or boundary features which may be helpful to differentiate between classes in data. Then in turn, these features are attached to a machine learning model, which will classify these features into their different categories, and then utilize this information when analyzing and classifying new objects.

Our aim of choosing this application is to implement this widely used algorithm for local community purposes such that spotting violent behavior in a traffic or crowded place, ensuring proper quality control of parts in manufacturing and etc. The main feature of our application will be to fit it to our local circumstances and conditions such that applying Azerbaijani Language image detection would be helpful for local people.

**Application of Machine Learning Methods**

**Resizing the images**

The first step we have done implementing this algorithm was resizing the images into the form that could take less space. For this purpose, we have used our own script that was written on Python programming language. What the script did was that it took the directory with images and scaled them into 800x600 scale which was less than the original size of the images.



*Figure. Scaling images*

**Deep Learning with Tensorflow**

**Neural Network**

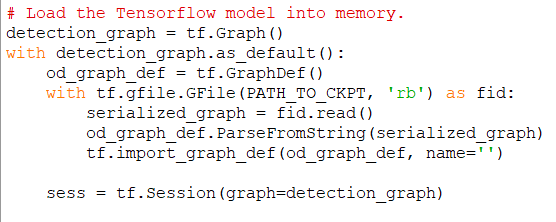
The fundamental objective of using neural networks for our dataset was diving the data into 23 categories which showed individual objects such as phones, projector, cameras and etc. For our dataset, we have used more than 90 images with different perspectives and angles. We have divided the images into two categories: testing and training. It is required to have 80% of the images in training folder and other 20% in the testing folder. All the sample images we have taken were labelled into classes which are shown in Table.

|  |  |
| --- | --- |
| Label | Class |
| 1 | Nijat |
| 2 | Qələm |
| 3 | Günəbaxan tumu |
| 4 | Lays |
| 5 | Iphone 5S |
| 6 | Pişik |
| 7 | Dəftər |

*Table. Labelled images*

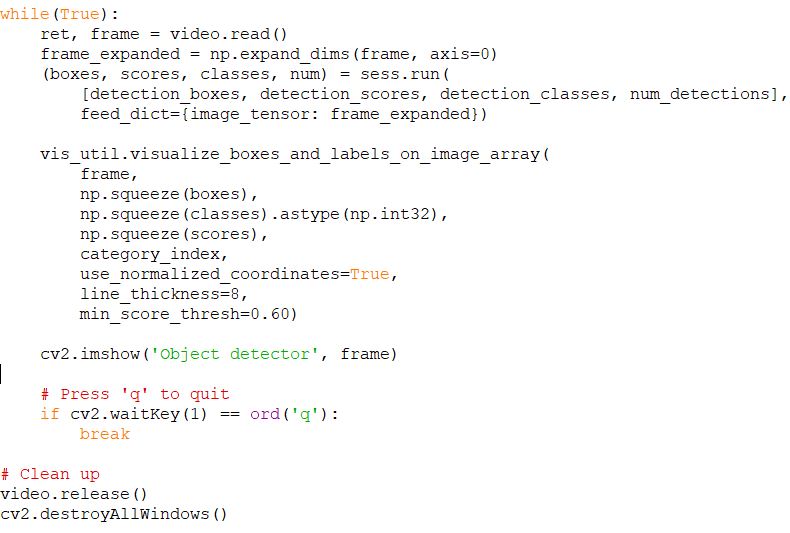
All the images that were shown in the label was mapped into one single label. The next step for our dataset was building the neural network which required training the model. When we have compiled the models we had to take three things into the account: loss function, optimizer and metrics which loss function was the most essential one. When we have trained the neural network models we had to specify train images and labels which we have defined previously. Then we had to ask the model to make predictions for the test case which we have initialized and added in the previous steps.

For running the neural network with our own model, we had to initialize a session with the help of ***Session()*** in order to pass the graph that we have defined for detection.



**OpenCV**

We have used OpenCV (stands for Open Source Vision Library which used for computer vision and machine learning) in order to get the vision for real-time object detection for our dataset. By the help of this library, we could manage to identify the objects, camera movements and getting track of moving objects in the camera. The main idea of using OpenCV was that getting the frame that we have initialized for our dataset by simply typing ***video.read()*** and returning the ***frame*** value for the OpenCV. Then we had to import the OpenCV and put the ***frame*** value into that as shown in Figure.



**References**

**Tensorflow by Google**

* <https://www.tensorflow.org/>

**A Beginner's Guide to Neural Networks and Deep Learning**

* <https://skymind.ai/wiki/neural-network>