SignMeUp

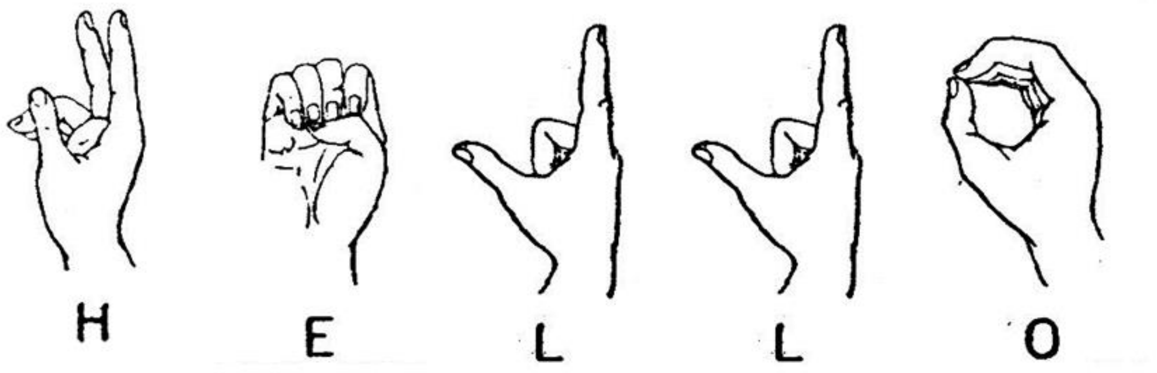


Table Of Contents

Introduction 3
Our project- Visions & Goals 4
Methodology 4
Data Collection 4
Dataset Overview 5
Data Processing/Annotation/Normalization 5
Experiments – Setup, Configuration 6
Algorithms, Object Detection architectures/systems 8
    What is Object Detection? 8
    Why is it useful for us? 9
Results & Quantitative Analysis 9
Qualitative & Error Analysis 10
Discussion, Comments/Notes and Future Work 12
Members/Roles 12
Time Plan 13

Introduction

In the line of this course, we were taught how to conceive the meaning of big data and have a bigger understanding about the numerous ways that we can use them and retrieve information from them. For example, a piece of text can be used for social media, documentation, text recognition, language recognition, word prediction, Question Answering Systems, text summarization, machine translation, sentiment analysis and others. After opening our horizons to the multiple use of big data we were introduced to ways of exploiting them. These ways were machine learning algorithms that would get as input a text, image or video and get trained to predict, translate and understand in the best rate possible the wanted result. The use of these algorithms is thoroughly explained bellow. The whole project is executed in anaconda notebook environment, in python language.

Our project- Visions & Goals

For this project, our team wanted to focus on a social matter that is close to our hearts. Living in our era, it is difficult for our generation to imagine that even with the technological advancements of 2021, there are still minority individuals that are not able to execute everyday activities which for others are granted and not even get a second thought. For this reason, we concentrated on hearing disabled people, and we looked for a way to help them communicate with a more user-friendly way, so we can make our society more open to them. Hearing disabled individuals are commonly secluded from everyday activities due to lack of communication means. Covid intensified this problem since the use of masks made lip reading impossible. Our vision is to try minimizing this gap and help people who do not understand sign language to interpret it and communicate with those individuals. Our plan is to use images of hand signs as input to train an algorithm to recognize those signs and translate them. Now may this project has seen more progress in other countries, not enough efforts are made in our our country and we owe it to our community to try and help.

Methodology

The first thing that we had to find was data. For our project, we chose to create our own dataset by taking pictures of us showing the signing letters of the Greek alphabet. This choice was made both because we wanted to participate in this cause the most way possible but also by necessity since there is a lack of data in the specific area that we examined. After creating our dataset, we had to clean our data and recreate some of the photos in a way that the boundaries were clear for the object detection. Moving forward we had to overcome some resources difficulties by using Google Colab Cloud. This need was unmerged by the fact that photos as opposed to text, occupy more memory and need more computational power.

We created the photos by using video capture in an algorithm, where we would use our computer cameras to capture the frames of us signing the letters. In order to train our algorithm, we had to use an object detection method, so that the computer could isolate our hand gestures. Object detection refers to the capability of computer and software systems to locate objects in an image/scene and identify each object. Object detection has been widely used for face detection, vehicle detection, pedestrian counting, web images, security systems and driverless cars.

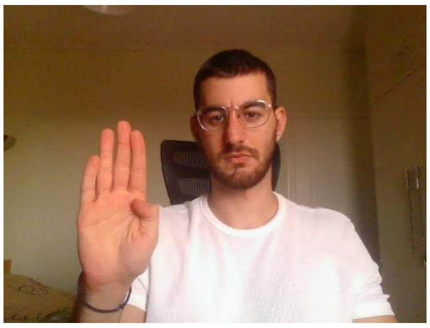
After completing all the procedural steps such as installing and importing the right packages like tensorflow, we set up the paths and created a label map. Continuing, we activated the object detection for our photos and installed the object detection API. Then, we created the TF records and downloaded the TF Models Pretrained Models from Tensorflow Model Zoo. We copied our model to the variable `‘CUSTOM_MODEL_NAME’` to our training folder and we updated the config for transfer learning. After that we trained our model and we exported it to our workspace folder. Moving forward, we loaded the train model from the checkpoint and we started detection in real time and we concluded by inferencing our trained models.

Data Collection

Our dataset was created completely from scratch. Our software (PhotoShooting.ipynb) which is handling the camera, captures a stream of 7 photos for each given letter locally, in which we ensured that we moved the sign properly in order to cover more angles and coordinates. The photos are stored in a separate folder along with the according letter name.

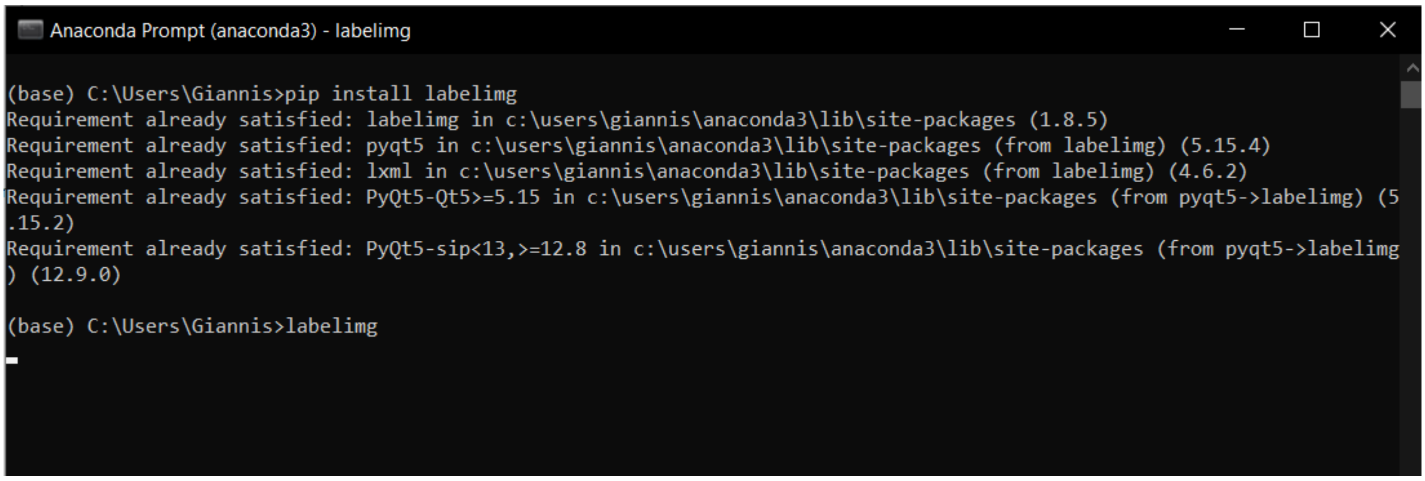
Dataset Overview

The photos used in the dataset will be letters from the Greek sign language that can be combined to form words. Each folder represents a letter and consists of 14 jpg images, implemented from different "actors", and their according annotations.

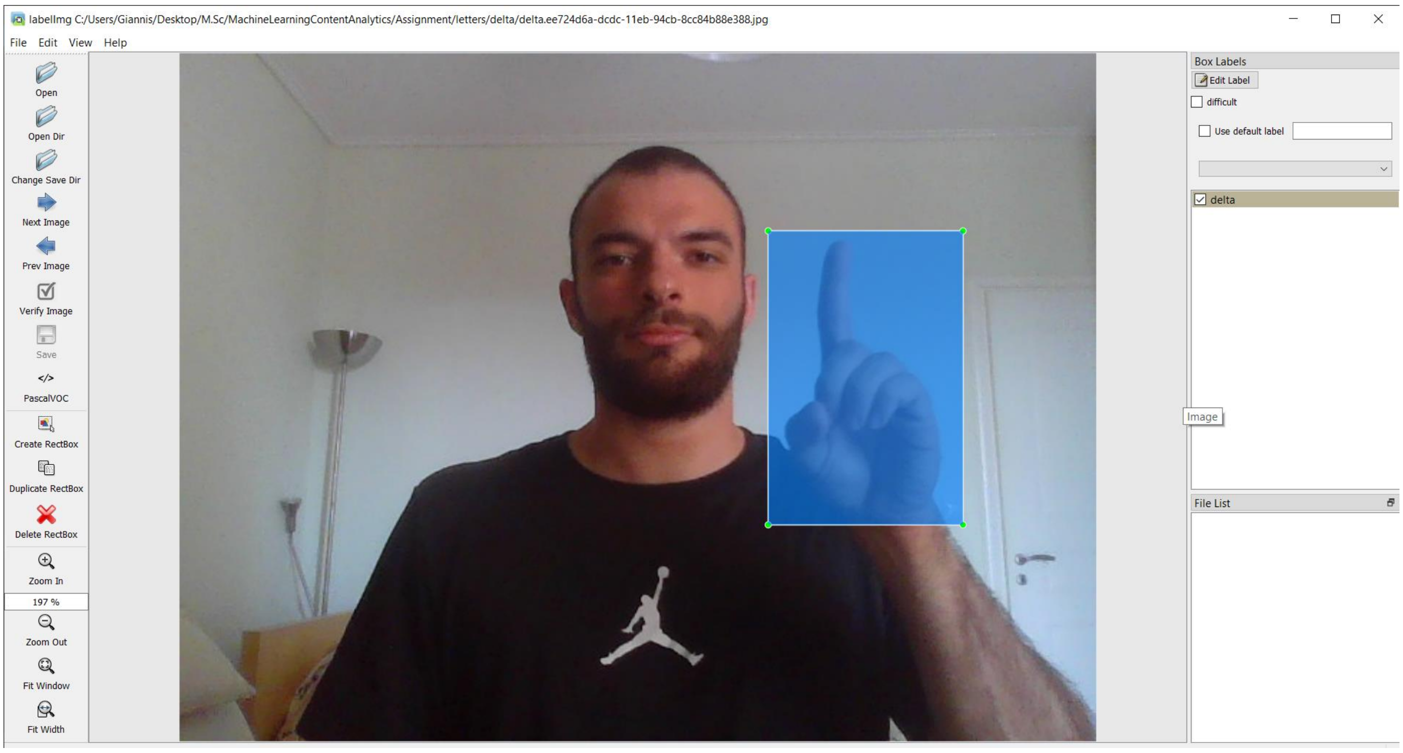


Data Processing/Annotation/Normalization

Right after capturing the photos that we needed to create our dataset as described above, we had to come up with a kind of data cleansing. Some of our photos were 'corrupted', so we had to replace them with new photographs of the corresponding Greek sign language letter respectively. After cleaning our dataset, we had to annotate the photos, in order to create the proper custom boundaries for our object detection project. To annotate the images we will be using the labelImg package using the command of our local computer again. The output of the annotations was saved in the same folder where the according photo was stored respectively.

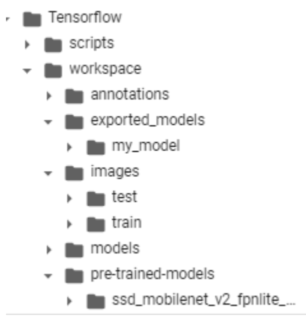


```
Anaconda Prompt (anaconda3) - labelimg

(base) C:\Users\Giannis>pip install labelimg
Requirement already satisfied: labelimg in c:\users\giannis\anaconda3\lib\site-packages (1.8.5)
Requirement already satisfied: pyqt5 in c:\users\giannis\anaconda3\lib\site-packages (from labelimg) (5.15.4)
Requirement already satisfied: lxml in c:\users\giannis\anaconda3\lib\site-packages (from labelimg) (4.6.2)
Requirement already satisfied: PyQt5-Qt5>=5.15 in c:\users\giannis\anaconda3\lib\site-packages (from pyqt5->labelimg) (5
.15.2)
Requirement already satisfied: PyQt5-sip<13,>=12.8 in c:\users\giannis\anaconda3\lib\site-packages (from pyqt5->labelimg
) (12.9.0)

(base) C:\Users\Giannis>labelimg
```



Experiments – Setup, Configuration

So firstly, we had to setup the environment to establish a custom object detection application. Initially we though that it would be easy to train our model locally, but that seemed to be difficult, as we did not have the required resources to train our model, not even a smaller sample of our data. Thus, we had to use a cloud service. With the use of Google Colab Cloud resources, i.e. greater GPU runtime, we managed to build our ambitious project. Our first move was to prepare the workspace in Google Colab and our primary step was to create a folder named TensorFlow. Forward, we had to create a new folder under TensorFlow and call it workspace. It is within the workspace that we will store all our training set-ups,



```
Tensorflow
  scripts
  workspace
    annotations
    exported_models
      my_model
    images
      test
      train
    models
    pre-trained-models
      ssd_mobilenet_v2_fpnlite_...
```

which will contain all files related to our model training. It is advisable to create a separate training folder each time we wish to train a different model. The typical structure for training folders is shown below.

Here's an explanation for each of the folders/filer shown in the above tree:

→ `annotations`: This folder will be used to store all `*.csv` files and the respective TensorFlow `*.record` files, which contain the list of annotations for our dataset images.

→ `images`: This folder contains a copy of all the images in our dataset, as well as the respective `*.xml` files produced for each one, once `labelImg` is used to annotate objects.

  - `images/train`: This folder contains a copy of all images, and the respective `*.xml` files, which will be used to train our model.
  - `images/test`: This folder contains a copy of all images, and the respective `*.xml` files, which will be used to test our model.

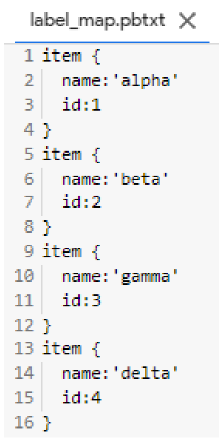
→ `models`: This folder will contain a sub-folder for each of training job. Each subfolder will contain the training pipeline configuration file `*.config`, as well as all files generated during the training and evaluation of our model.

→ `pre-trained-models`: This folder will contain the downloaded pre-trained models, which shall be used as a starting checkpoint for our training jobs.

Once we have collected all the images, as mentioned before, to be used to test your model, we had to place them inside the folder workspace/images. Then using Anaconda/Command Prompt window we activated labelImg . Starting labelImg, we pointed it to the appropriate image folder for each distinct image sign respectively.

After executing python labelImg.py a File Explorer Dialog windows opens. After selecting "Select Folder" button, we started annotating the images start annotating your images. Once we finished annotating images, it is a conventional to use part of it for training, and the rest is used for evaluation purposes. Once we have split the dataset, we had to upload all training images, together with their corresponding `*.xml` files, and place them inside the `workspace/images/train` folder that we have previously created in Colab . Similarly, we uploaded all testing images, with their `*.xml` files, and paste them inside `workspace /images/test`.

The next step of our project had to do with the creation of the Label Map. TensorFlow requires a label map, which namely maps each of the used labels to an integer value. This label map is used both by the training and detection processes. An example of that label map for the first 4 letters of our alphabet is shown below:

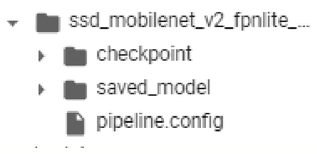


```
item {
  name:'alpha'
  id:1
}
item {
  name:'beta'
  id:2
}
item {
  name:'gamma'
  id:3
}
item {
  name:'delta'
  id:4
}
```

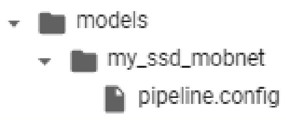
Under the TensorFlow folder, we have created a folder TensorFlow/scripts, which we used to store a useful script. After having generated our annotations and split our dataset into the desired subsamples, it is time to convert our annotations into the so called TFRecord format. To do this we can write a simple script that iterates through all `*.xml` files in the `workspace/images/train` and `workspace /images/test` folders, and generates a `*.record` file for each of the two. We have found that script ready in the TensorFlow official website. After executing the previous steps, 2 new files are created named `test.record` and `train.record`, respectively.

Then we needed to download the pre-trained network for the model we wish to use. This can be done by simply clicking on the name of the desired model in the table found in https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md . Clicking on the name of the model we will use, initiates a download for a `*.tar.gz` file.

Once the `*.tar.gz` file has been downloaded, we had to unzip it . Next, inside `*.tar` folder that you see when the compressed folder is opened, we extracted its contents inside the folder `workspace/pre-trained-models`. Since we downloaded the ssd mobilenet v2 fpnlite 320x320 model, our `workspace` directory should now look as follows:



After downloading and extracting the pre-trained model, let's create a directory for our training object detection project. Under the `workspace/models` create a new directory named `my_ssd_mobnet` and copy the `workspace/pre-trained-models/ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/pipeline.config` file inside the created directory. Our directory now looks like this:



Before we begin training our model, some extra modifications should be made , or just copy the path through to that file , the `TensorFlow/models/research/object_detection/model_main_tf2.py` script and paste it straight into our `workspace` folder. We needed this script in order to initiate the train of our model.

To start the training of our model we had to run the following command inside the `workspace`:
(Note that in Colab , using ! you could execute command line commands .)

```
!python /content/Tensorflow/models/research/object_detection/model_main_tf2.py --
model_dir=/content/Tensorflow/workspace/models/my_ssd_mobnet --
pipeline_config_path=/content/Tensorflow/workspace/models/my_ssd_mobnet/pipeline.co
nfig
```

Algorithms, Object Detection architectures/systems

What is Object Detection?

Object detection is a computer vision technique that allows us to identify and locate objects in an image or video. With this kind of identification and localization, object detection can be used to count objects in a scene and determine and track their precise locations, all while accurately labeling them.

Why is it useful for us?

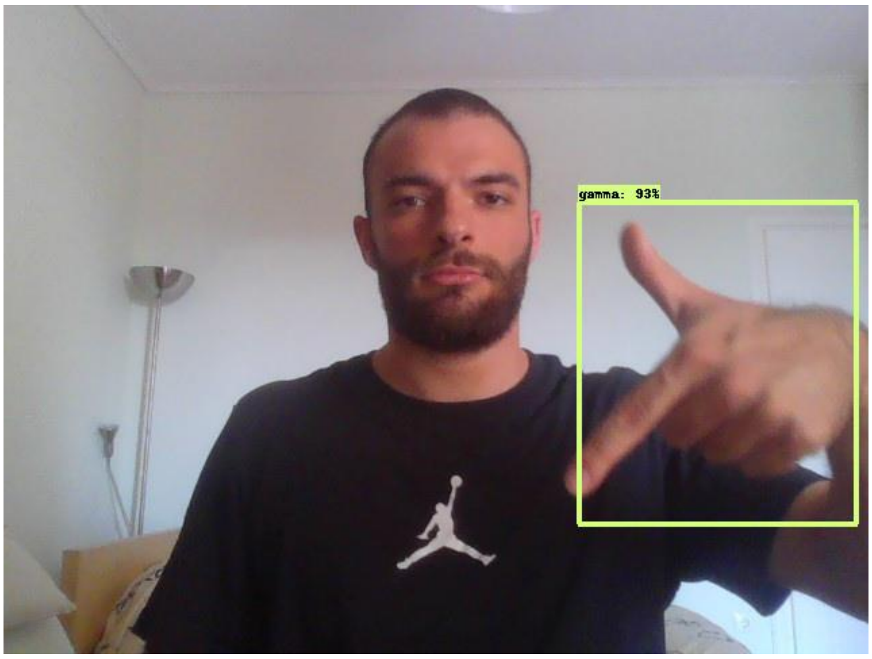
This functionality was, also, necessary for us in order to capture our hand gestures from the photos that were taken from wherever in the space they were and be able to isolate them in order to be trained in translating which letter they showed. For this step in our assignment, we followed instructions for tensorflow object detection API in python programming language. After importing the necessary packages and setting up a useful folder structure, we trained the model on the captures photos and their according annotations.

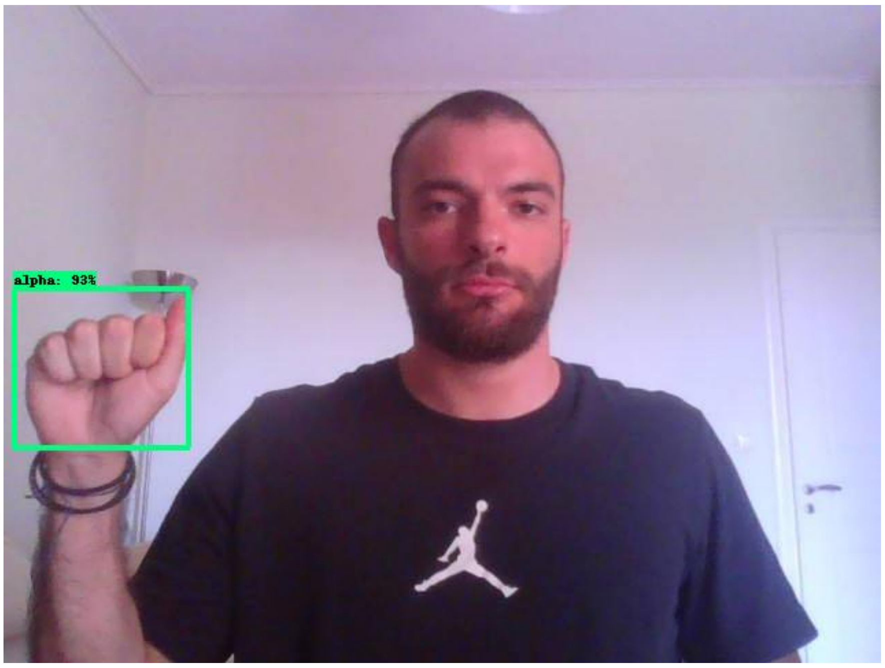
The algorithm that we used to capture our photos worked in a way where we started by labeling each letter so that we knew which photo/hand gesture was what which then were saved in a folder all together. Then we used videocapture and the camera on our computers would open and started capturing frames were our team members were signing the Greek alphabet.

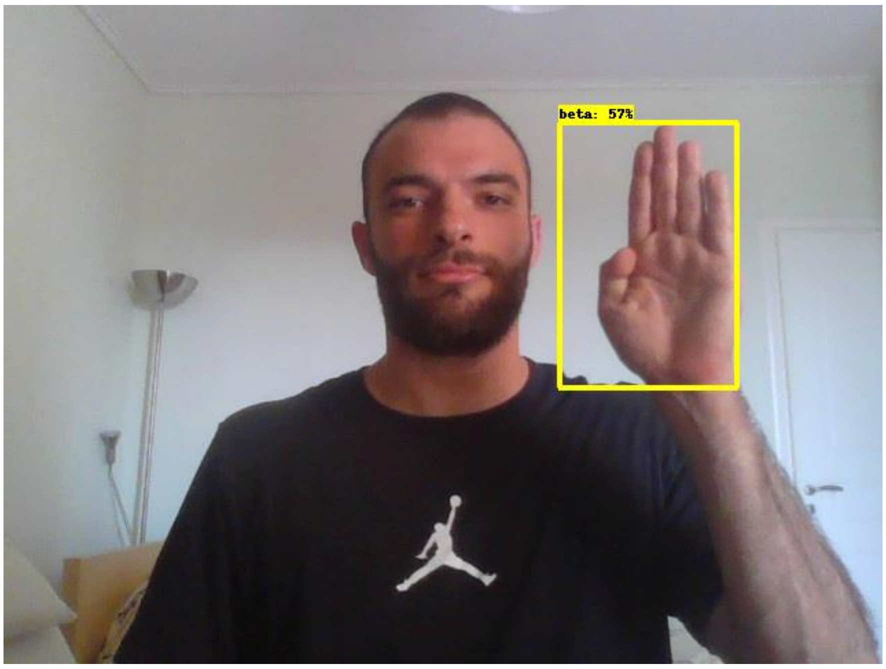
Because of the heavy load of the photos and the intensive needs of resources, the training process lasted many hours. For this reason, we deployed – exported the model in order to use it any time by just inserting a new photo.

Results & Quantitative Analysis

Below, we can see the results of our model in some random chosen photos:



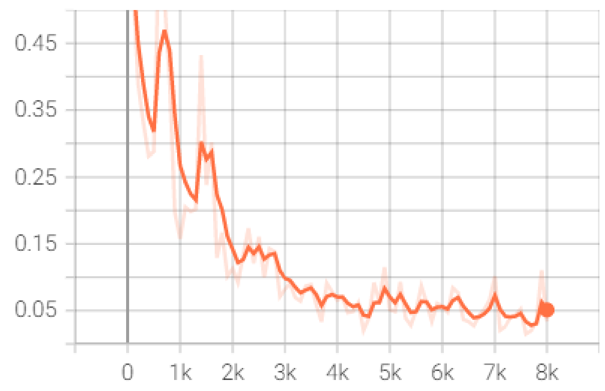




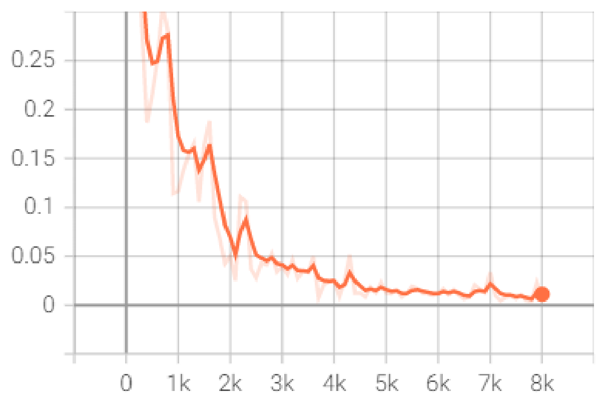
Qualitative & Error Analysis

The visualizations bellow show the evolution of each metric in k steps.

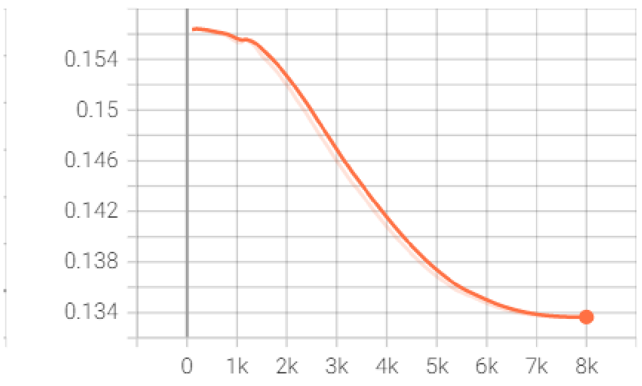
Loss/classification_loss
tag: Loss/classification_loss



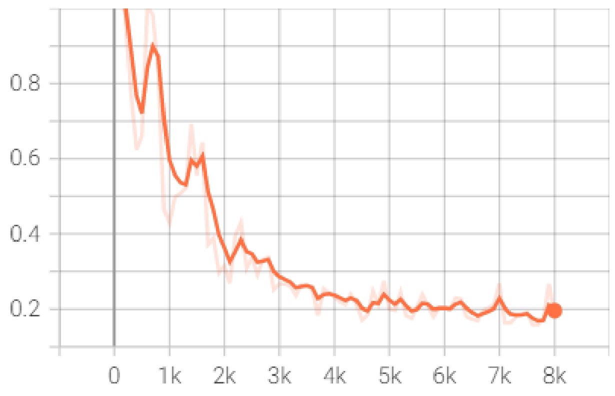
Loss/localization_loss
tag: Loss/localization_loss



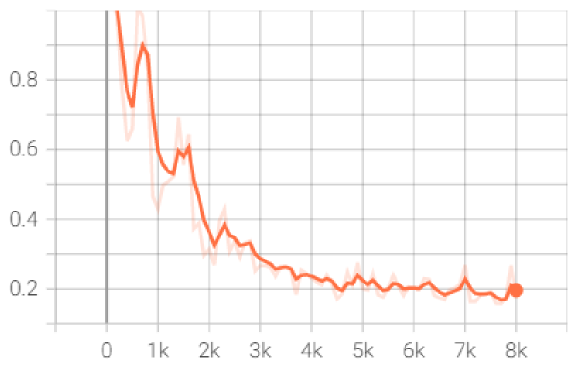
Loss/regularization_loss
tag: Loss/regularization_loss



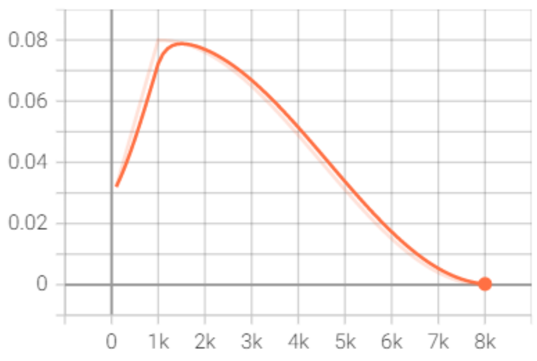
Loss/total_loss
tag: Loss/total_loss

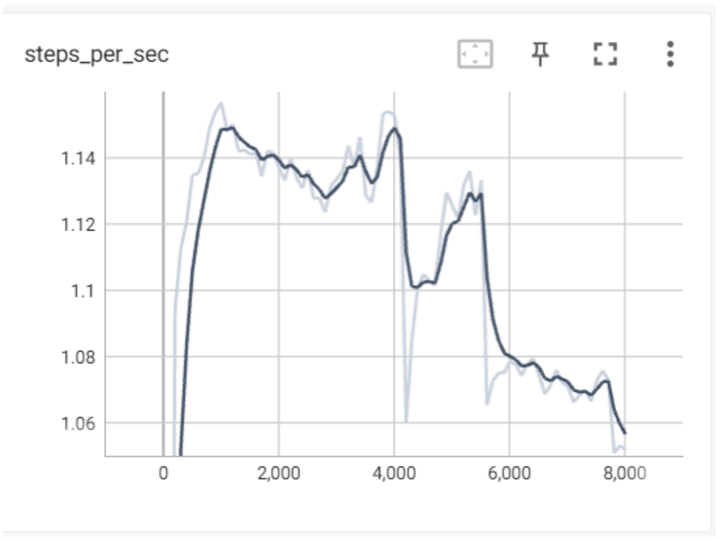


Loss/total_loss
tag: Loss/total_loss



learning_rate
tag: learning_rate





Discussion, Comments/Notes and Future Work

There is no doubt that there is room for improvements to be made. Those improvements could be found both in the execution and in the data that were used. Our initial view was to use all the letters but also words so we could have an enriched dataset. This plan was not fulfilled since we didn't have access to better resources. Moreover, there is the possibility of building a better trained model, in the case where we did not work only with the default Colab GPU runtime power. Additionally, our team would dream to take our first efforts a lot further in the future. We would like to transform our work in a program that capture live streaming video and has the ability to recognize the sign and detect it in the stream video. Using a program like the one mentioned we would be able to build an application that would translate in real time the signing language and it would give voice to all the people that are struggling to be included in our community. Applications similar to this already exist, but not as far as the Greek sign language is concerned.

Members/Roles

Georgios Zygoukis	F2822004
Ioannis Sakkis	F2822015
Maria Sergoulopoulou	F2822016

In order to have a complete overview of the course and of such a project, probably all of the members will be engaged in each and every step of the process. However, given our academic and professional backgrounds, Mrs. Sergoulopoulou will have high responsibility in composing the business report and the final presentation and communicate any business-oriented weaknesses of the project. On the other hand, Mr. Zygoukis and Sakkis will be responsible for the optimization and the composition of the technological needs of the project accordingly.

Time Plan

Throughout the whole process we followed a waterfall way of working. At first, we created the dataset, then we downloaded and implemented all the necessary functionality and transfer- learning and finally training of the model and optimizations of the procedures and presentation.

Bibliography

https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/2.2.0/training.html

https://towardsdatascience.com/custom-object-detection-using-tensorflow-from-scratch-e61da2e10087

https://www.youtube.com/watch?v=pDXdlXlaCco&ab_channel=NicholasRenotte

Appendices