Real time Macedonian sign language recognition

Risto Trajanov, 171523

***Abstract*— There has always been a strugle for people with speech and hearing disabilities to communicate and participate in everyday social activities. The link between deaf and ordenary people is the sign language. But not many ordinary people know sign language and it is difficult for the impaired people to interact with the world. Solutions for speech and hearing impairment have never been more reachable. With the advancment in machine vision and the availability of pre-trained models we have the link between the two worlds on the tip of our fingers. In this work we present a prototype for real-time object detection model that captures the sign language and translates it for the ordinary people.**

I. Introduction

The precise number of hearing and speach impaired people in Macedonia is unknown but 6000 people requested signed news on Macedonian television. This can give us motivation that people are struguling with everyday activities and need a tool that will make their lives easier. There are a few initiative in Macedonia that supports impaired people like Talking Hands[9] and The National association of deaf and Hard of Hearing of the Republic of North Macedonia [6]. Talking hands is a virtual dictionary of almost the whole macedonian sign language. They arranged 100 people of which 80% were deaf and recorded video materials of people showing the alphabet and common words [1]. The assoctiation also has a dictionary of the alphabet but does not have video materials. The purpose of the association is to fight for the rights of the impaired.

As technology advances, more and more solution use the new iprovements to help others. Inspirated from a student from Greece that was developing a model like this we decided to build a real time object detection model for Macedonian sign language. In this work work we propose a pipeline that can be used to train a real time model that can predict Macedonian sign language. In the next few section we will discuss the pipeline on Figure 1.

II. Data

We can state that data is the most difficult problem here because we did not had annotated photos to feed our model. We had the videos from Talking Hands but it was more complicated to extract the photos from there and then annotate them. We decided that we will make the dataset our selves.

We trained our model on two alphabet characters, A and B. For that purpose we took 70 photos of our selves showing each letter. After that we annotated the images using a python tool [5].

Because our model was under performing we augmented every photo in 16 different ways using a python library called imgaug [2]. In Figure are shown some of the augmentation done with the library. Figure 2 and Figure 3 are examples from the augmentation.

III. Results and Discussion

In this study we provide a prototype of real-time object detection using Tensorflow Object Detection Api [7]. All the methods mentioned in this section are explained in the following section Methods.

First of all we developed an image collection script that helped us speed up the data collection process which is a crucial thing because we did not had any annotated data. Using the label img library we annotated the data and produced the xml annotation for the collected images. The amount of images was not enough to fine tune the model so we augmented the data. This was a challenging step because the bounding boxes(xml files) had to be altered also.

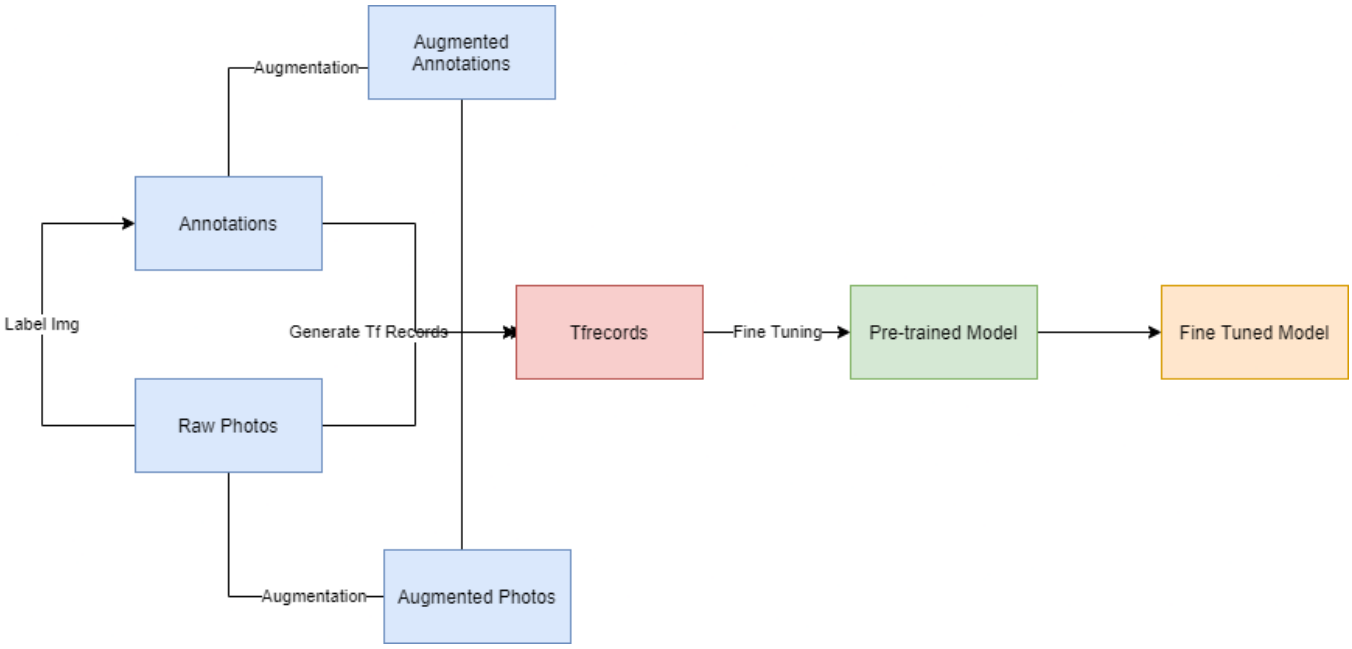


Fig. 1: Proposed pipeline for real time object detection model training.

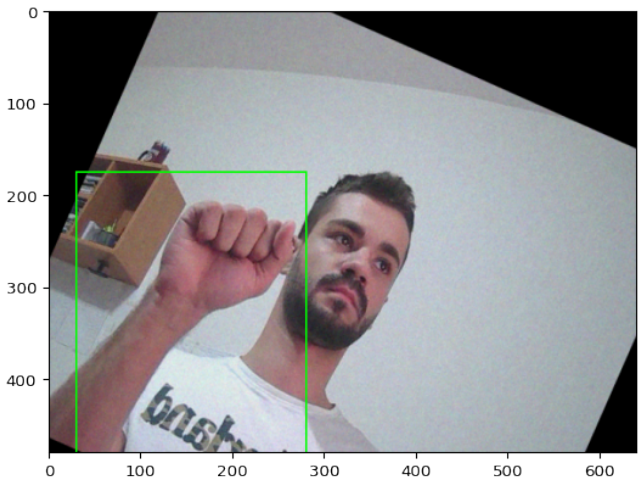


Fig. 2: Augmentation on A

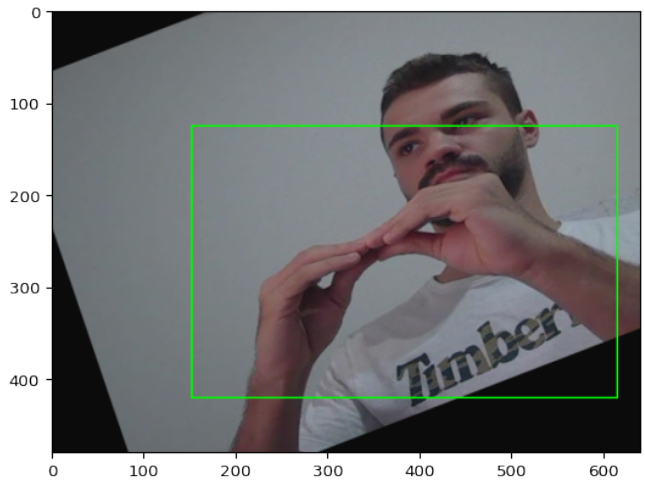


Fig. 3: Augmentation on B

A. *Model*

Moving on to the model we needed to alter our data in order to shape it as the model input requirements. First we created a label map which is a document containing the labels, in our case letters, that were used to tell the fine tuned model what should he predict. The main entry point in the model are TF(tensorflow) records. The building blocks of the records are the images, the annotations(xml's) and the label map. This is all created using scripts provided by tensorflow. And coming to the core of the model we used ssd_mobilenet_v2_320x320_coco17_tpu-8 as a base model for our fine-tuning. The pre-trained model is part of the Tensorflow 2.0 object detection models. This model was chosen because it returns boxes, areas with probabilities that some of the classes is detected. The model is not perfect, there are far better models in the Tensorflow 2 Detection Model Zoo [10] but it is a good starting point for further improvements. The main reason I chose this model is because is fast, it takes only 22 ms for a reply.

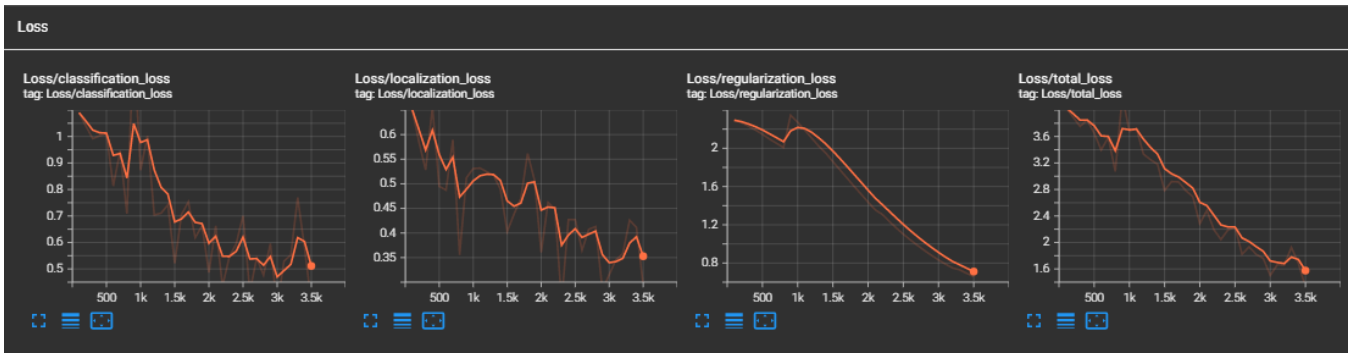


Fig. 4: Training loss presented by tensorboard.

TABLE I: Mean IoU and Class prediction when maximizing IoU

Metric	Score
Mean IoU	0.5828
Class Prediction Accuracy	0.7727

TABLE II: Mean IoU and Class prediction Accuracy when maximizing both parameters

Metric	Score
Mean IoU	0.5188
Class Prediction Accuracy	0.9848

The model was trained on 2332 images and tested on 61.

After experimenting with the training steps with assistance from tensorboard 4. We decided that 3500 steps was the optimal number of steps for the model. When I trained the model with more steps the loss went up.

B. Evaluation

For evaluating the model we used two metrics, Intersection over Union(IoU) and raw class prediction.

The evaluation was done in two scenarios taking the 3 predictions with the highest score. In the first scenario the IoU was maximized as we chose to take the bounding box with the highest IoU. The results are given in the Table I

The second scenario we take the case where there is a class match and the highest IoU, so we maximize both metrics in a manner that there is as little loss as possible. The results are in Table II

Also we provide a few prediction examples in Figure 5

Also we calculated the precision and recall according to this metrics.

- if IoU ≥ 0.5, classify the object detection as True Positive(TP)
- if Iou < 0.5, then it is a wrong detection and classify it as False Positive(FP)
- When a ground truth is present in the image and model failed to detect the object, classify it as False Negative(FN).
- True Negative (TN): TN is every part of the image where we did not predict an object. This metrics is not useful for object detection, hence we ignore TN.

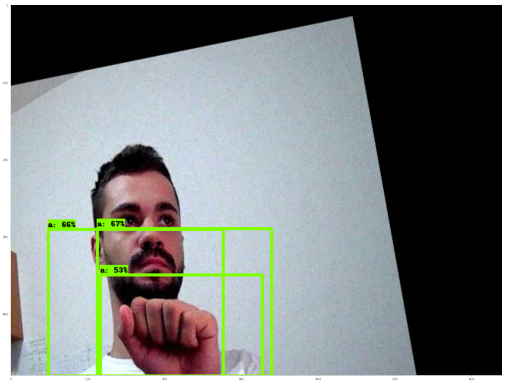
The equations for calculation Precision and Recall are given at 1 and 2 respectfully.

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \tag{1}$$

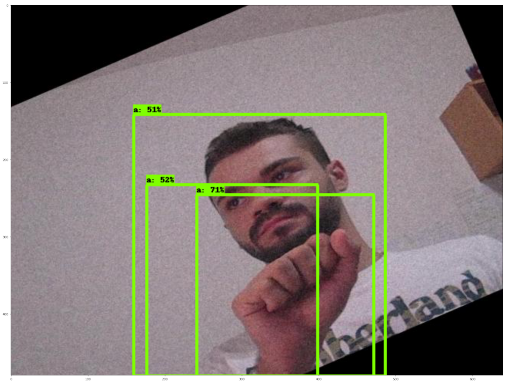
$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \tag{2}$$

The results for the scenario where we maximize IoU is given at Table III. And for the second scenario, where we find the maximum IoU based on the predicted class, results are given at Table IV.

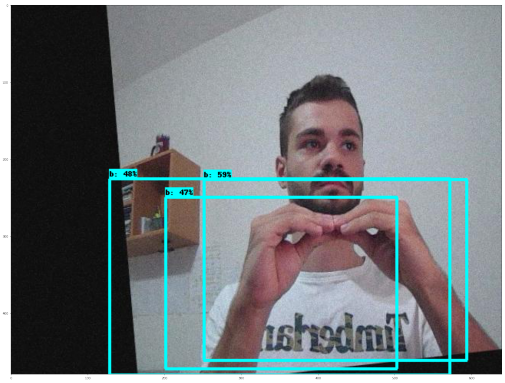
Preview of the real time predictions of the model can be found [8]



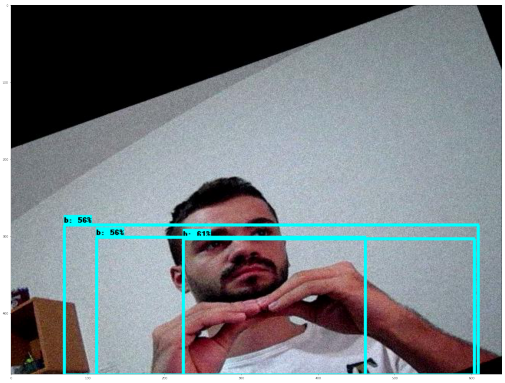
(a) A



(b) A



(c) B



(d) B

Fig. 5: Model prediction of letters A and B.

TABLE III: Precision and Recall when maximizing IoU

Metric	Score
Precision	0.6862
Recall	1

TABLE IV: Precision and Recall when maximizing both parameters

Metric	Score
Precision	0.6
Recall	1

IV. Methods

A. *Object detection: Tensorflow 2.0*

Object detection can be defined as a branch of computer vision which deals with the localization and the identification of an object. Object localization and identification are two different tasks that are put together to achieve this singular goal of object detection. Object localization deals with specifying the location of an object in an image or a video stream, while object identification deals with assigning the object to a specific label, class, or description. With computer vision, developers can flexibly do things like embed surveillance tracking systems for security enhancement, real-time crop prediction, real-time disease identification/ tracking in the human cells, etc.

In contrast to TensorFlow 1.x, where different Python packages needed to be installed for one to run TensorFlow on either their CPU or GPU (namely tensorflow and tensorflow-gpu), TensorFlow 2.x only requires that the tensorflow package is installed and automatically checks to see if a GPU can be successfully registered. For the results obtained in this project, we used Tensorflow

object detectio Api. The new api containes a model ”zoo” in which they provide a series of pre-traiend models each with his own pros and cons.

B. Augmentation

Due to the low amount of data we did augmentation on the existing one. Augmentation is the process of altering the image by flipping, rotating, scaling, cropping or putting Gaussian noise. All this methods were used to get more data to train our model.

C. TFRecords

The TFRecord file can be seen as a wrapper around all the single data samples. Every single data sample is called an Example, and is essentially a dictionary storing the mapping between a key and our actual data. TFRecord file is stored sequentially, enabling fast streaming due to low access times. And secondly, the TFRecord files are natively integrated into TensorFlows tf.data API, easily enabling batching, shuffling, caching, and the like.

D. Pretrained mobilenetv2

MobileNet is an object detector released in 2017 as an efficient CNN architecture designed for mobile and embedded vision application. This architecture uses proven depth-wise separable convolutions to build lightweight deep neural networks. More information about the architecture can be found here [4]

E. Intersection over Union

IoU computes intersection over the union of the two bounding boxes; the bounding box for the ground truth and the predicted bounding box. An IoU of 1 implies that predicted and the ground-truth bounding boxes perfectly overlap as shown in Figure 6.

V. Code and Data availability

All the code and data used for this project is available at [3].

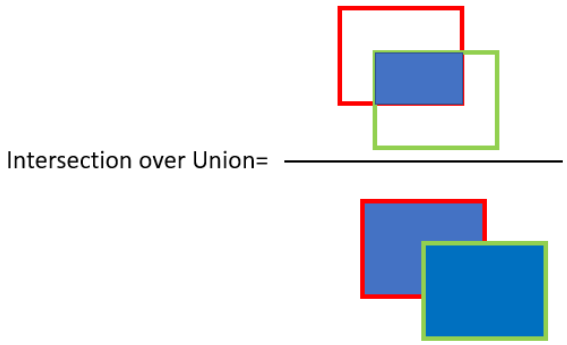


Fig. 6: Intersection over Union

References

[1] *Article Znakoven*. https://sdk.mk/index.php/magazin/koj-saka-mozhe-da-go-nauchi-makedonskiot-znakoven-jazik-promovirana-veb-platforma-za-gluvi/.

[2] *Augmentation Image*. https://github.com/aleju/imgaug.

[3] *Github repo*. https://github.com/risto-trajanov/real-time-sign-language-recognition-macedonian.

[4] Andrew G. Howard et al. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. 2017. arXiv: 1704.04861 [cs.CV].

[5] *Label Image*. https://github.com/tzutalin/labelImg.

[6] *National Association*. http://www.deafmkd.org.mk/.

[7] *Object detection*. https://github.com/tensorflow/models/tree/master/research/object_detection.

[8] *Real time preview*. https://github.com/risto-trajanov/real-time-sign-language-recognition-macedonian#real-time-preview.

[9] *Talking Hands*. https://znakoven.mk/.

[10] *Tensorflow Models*. https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md.