BABES-BOLYAI UNIVERISTY CLUJ-NAPOCA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

BACHELOR THESIS

Static Sign-Language Recognition

Supervisor:
Prof. Dr. Laura DIOSAN

Author:
Norbert-Cristian BERECZKI

2018

Abstract

In the last three decates there have been significant contributions related to sign-language recognition. So it is imperative to continue contributing to this topic in order to improve how deaf-mute people interact with society. Many papers have been published presenting methods using either hand-colored gloves or using special devices such as Microsoft Kinect. In the real world, most of the time, devices like these are difficult to aquire. However, image streams and videos are more common means of procuring data from a user. In this work we present a comparisson between a very robust, but slow, model and a more fast, but not as accurate model. In order to develop such a tranlator we employed Machine Learling techniques. Our models are based on using RetinaNet with pre-trained backbones.

This work is the result of my own activity. I have neither given nor received unauthorized assistance on this work.

Contents

Chapter 1

Introduction

In Romania there are more than 23 000 people with hearing loss deficiency and only 15% of them can find a job due to the fact that employees do not trust their abilities. Also, when they go to public institutions to pay their bills and taxes they do not get any authorized translator. So, it becomes imperative to start contributing to this field more and more in order to improve their well-being. In this article we present a comparison between some backbones for our model: the very accurate ResNet and the more faster MobileNet. We start by describing the problem to be solved and some related work, then we move to providing some theory on these models. We then present some experiments we have done with these models. The user manual for an application we developed, to get a feeling of our work, follows. We end with giving some conclusions about the work.

Chapter 2

Scientific Problem

2.1 Problem definition

The problem which we want to solve is of great importance to the deaf-mute community.

In the recent couple of decades there have been many contributions related to sign-language recognition. Most of this work is concentrated on hand-colored gloves and Microsoft Kinect. However, in real life, it would be much easier for the user to have a recognition system that only requires image streams.

Now, we have reduced the problem to classifying certain images in the stream. We proposed a solution for classifying fingerspelled letters in images and also detecting their bounding box.

Chapter 3

Related work

We start by talking about Brandon Garcia's paper [8], which resembles this one by the means that they also make use of Convolutional Neurla Networks. They use transfer learning on a pre-trained GoogleNet and freeze most of the layers, reinitialing the classification layers using Xavier Initialization. The pre-trained model was trained on the ILSVRC2012 dataset and then, using transfer learning, they trained the model on the Surrey University and Massey University datasets and attained validation accuracy below 80% for all letters and 98% for the first five letters.

We also want to mention the work of Lungociu[2] who attained 80% recognition accuracy on a set of 14 letters using artificial neural networks and some digital preprocessing techniques. In his work he used strong preprocessing techniques in the following order: image scaling, skin detection, noise reduction, shape detection, shape signature and computation of fourier descriptors. The ANN would take as input a vector of 32 fourier descriptors and output the classification probability related to each letter.

Arguably, the most significant advantage of neural networks is that they learn the most important classification features. However, they require considerably more time and data to train. To date, most have been relatively shallow. Mekala et al. classified video of ASL letters into text using advanced feature extraction and a 3-layer Neural Network [4]. They extracted features in two categories: hand position and movement. Prior to ASL classification, they identify the presence and location of 6 "points of interest" in the hand: each of the fingertips and the center of the palm. Mekala et al. also take Fourier Transforms of the images and identify what section of the frame the hand is located in.

Some tried to create dynamic sign-language translators. Starner and Pentland used in their work[7] Hidden Markov Models and a 3-D glove that tracks hand movement.Since the glove is able to obtain 3-D information from the hand regardless of spatial orientation, they were able to achieve an impressive accuracy of 99.2% on the test set. Their HMM uses timeseries data to track hand movements and classify based on where the hand has been in recent frames

Chapter 4

Proposed approach

4.1 Theoretical Aspects

We start describing some fundamental concepts behind the models used, however further references can be found here [3]. RetinaNet belongs to the category of one stage dense detectors. We say these because for each spatial location on some convolutional feature maps we are trying to mach a fixed number of anchor boxes.

The models consist of a feature pyramid network and 2 task specific subnetoworks: classification and bounding box regression. The backbone of the models can vary, however we used in our work ResNets and MobileNets. In the following subsections we will summarize each part of the architecture.

4.1.1 Residual Networks

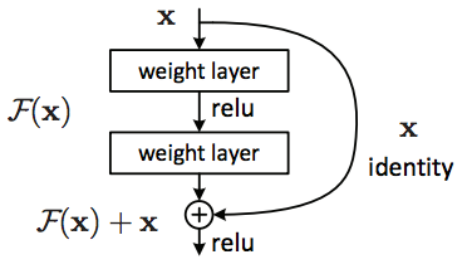

Figure 4.1: A building block of a Residual Network representing a skip connection

In this subsection we describe the concept of residual networks and how they come into play in RetinaNet. In many convolutional networks including the VGG network, as one stacks more and more layers over each other, the accuracy of the model degrades and becomes even worse as when you would have only a few layers. This is demonstrated in practice in [5]. So, residual networks come into rescue with the idea of *skip connections*. In the past one would let the stacked layers fit a mapping of the input into some output feature say $H(x)$. Now, we let the layers fit another function $F(x) = H(x) - x$, call it a residual function. The original mapping is recast to $F(x) + x$ as seen in Fig. 4.1. The authors of [5] say that it is easier to optimize the residual mapping than the original one.

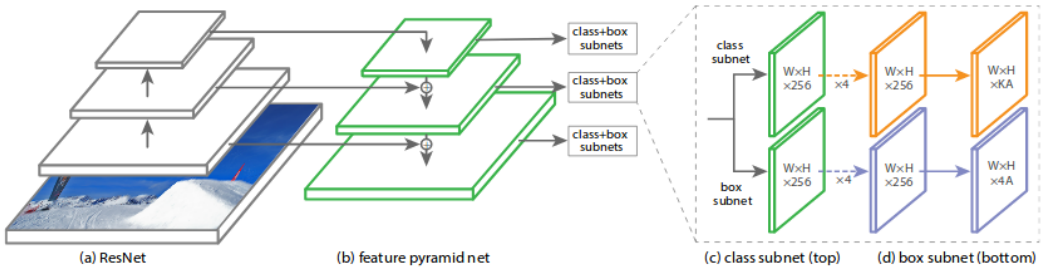

Figure 4.2: Architecture of RetinaNet

4.1.2 Feature Pyramid Network

We will describe the FPN submodule using the ResNet backbone. You can also see the FPN submodule in the left part of Fig. 4.2. The main purpose of FPN is to produce a multi-scale feature pyramid which has strong semantics at all levels. The idea behind this is that if you have only a simple pyramid of feature maps then you will not have the same semantic power on layers more to the center of the network compared to layers deeper in the network. So, as described in [6] this was solved by the use of lateral connections and top-down paths(which are practically upsampling using nearest neighbour interpolation). Now, having the pyramid of semantically strong features(right side of Fig. 4.2) for each spatial location, on each layer, we are attaching a set of A anchor. We then send each set of A anchor boxes to both of the task-specific subnetworks.

4.1.3 Classification Subnetwork

This subnetwork consists of a fully convolutional network which in turn consists of some convolutional layers and produces an output of the form WxHxKA, K being the number of ground-truth classes plus the background class. In the training phase we apply to the output of this subnetwork the focal loss. We describe why we use the focal loss below.

4.1.4 Focal Loss

Instead of the cross-entropy loss function we used a more novel loss function called the focal loss which, at its most simples form, looks like this:

$$FL(p_t) = -(1 - p_t)^{\gamma} \log(p_t)$$

. The reason behind using focal loss is that (instead of using Online Hard Example Mining) the loss naturally emphasizes learning from hard examples, by downweighting easy negatives (sometimes by a factor of x1000) and concentrating on hard ones. The focal loss function stands at the end of the classification FCN and is applied to all 100k anchor boxes sampled from a single image.

4.1.5 Box Regression Subnetwork

This subnetwork has almost the same design as the classification subnetwork, it only differs in the output where it outputs a 4A array of targets. For each anchor box

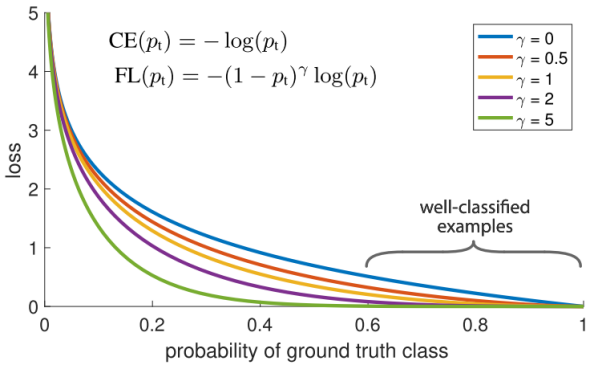

Figure 4.3: We propose a novel loss we term the Focal Loss that adds a factor $(1 - p_t)^\gamma$ to the standard cross entropy criterion. Setting $\gamma > 0$ reduces the relative loss for well-classified examples ($p_t > .5$), putting more focus on hard, misclassified examples. As our experiments will demonstrate, the proposed focal loss enables training highly accurate dense object detectors in the presence of vast numbers of easy background examples

there are outputed 4 targets which represent the geometric transformation that the anchor box need so it will contain inside the object it is centered upon. The loss used for this subnetwork is the *L1* loss for bounding box regression used also in Fast R-CNN.

Chapter 5

Application

5.1 Methodology

5.1.1 Training

First of all, we trained 2 instances of ReinaNet, both have a pre-trained backbone of ResNets, trained on the ImageNet object detection dataset. For the first one we only fine-tune the backbone and we fit the 2 subnetworks. The second one has the backbone frozen and we applied transfer learning by fitting only the 2 task-specific subnetworks.

For training we merged together two datasets. First we took the Massey University Gesture Dataset 2012, which contains 2524 close-up, color images that are cropped such that the hands touch all four edges of the frame. This dataset was captured from five users and a frame in the dataset averages the size of 500x500.
The second dataset we used was the one found here [1]. This dataset provided bounding boxes and contained frames that average 320x240 in image size. We also used data augmentation techniques on this dataset by means of: horizontal flipping(with probability of 0.5), vertical flipping(p=0.5), gamma adjustment(p=0.3), blurring(p=0.3), sharpening(p=0.2) and added noise(p=0.1) for each image in this dataset. Both datasets were merged and split into 2 parts train and validation. The training split contained 4315 images and the validation split 720 randomly sampled images.

We will talk here only about the fine-tuned model because the resemblance in very striking between the two. The training took almost 30 hours for the fine-tuned model and ran for 22 epochs each having 10 000 iterations using a mini-batch of 1. The training was done on a Nvidia GPU Titan X of 12 GB of RAM, which was enough for our model, however it was not enough to train both instances at the same time. We present in Fig. 5.1 some images with the evolution of the loss functions and the mAP during the passing of each epochs.

From Fig. 5.1 we can see that the model converges in only 8-10 epochs and succeeding in having an almost perfect mAP for the classification task. However, there is still error in the detection of the bounding box, but we do not mind it because it is no that significant nor important as the classification error. In the next section

we present the hyperparameters we used for the model and the way the model does inference.

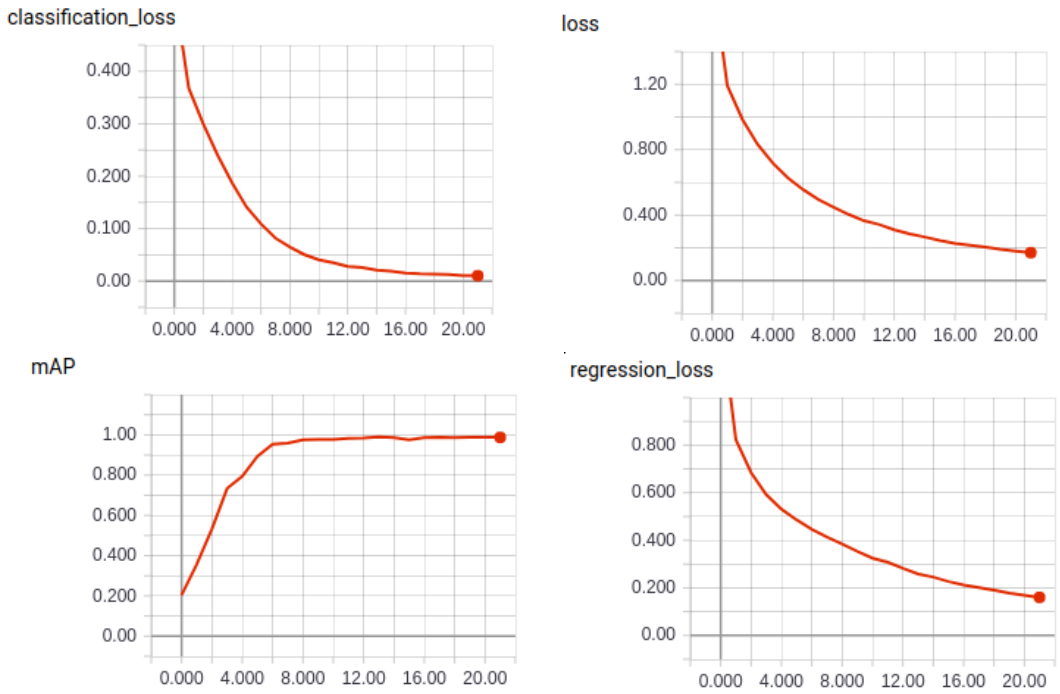

Figure 5.1: Loss metrics for the RetinaNet fine-tuned model

5.1.2 Hyperparameters and inference

The depth of the residual network is 50 and we use only 5 pyramid levels. On each level (also, each pyramid level has resolution 2^l lower than the input, where l is the level of the pyramid) we propose at each spatial location a fixed size anchor with 3 aspect ratios. In total the number of proposed regions is 100k, however at inference time only top-1k scoring are being decoded and after that non-maximum supression is applied. For backpropagating the gradients we used the adam optimizer with a learning rate of 0.00001 and a clipnorm of 0.001. We also use for training a Learning-Rate-On-Plateau-Reducer with a patience of 2 epochs (of lr no improving), auto mode and a factor of 0.1.

We have also done another experiment with a MobileNet backbone with proved to have a low accuracy point 42%, so we will not include it here. The reason for doing that is that on a laptop with an Nvidia 840M with 2GB of VRAM the average inference time of the RetinaNet-MobileNet was 0.7 seconds per image, whereas the average inference time of the RetinaNet with a ResNet50 backbone was 1.5 seconds, which does not qualify exactly for a real-time detector. However, we conducted inference-time-experiments on a computer that contains Nvidia Titan X GPU, and we observed that tha average inference time is 0.25 seconds. This would employ a detector to work with 4 frames per second which still is not so great.

labels	A	B	C	D	E	F	G	H	I	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
precision	0.967	1.0	1.0	0.906	1.0	1.0	0.968	1.0	0.903	1.0	1.0	1.0	0.968	0.967	1.0	0.968	1.0	0.967	0.967	0.909	1.0	0.967	1.0	0.938
recall	0.967	0.967	1.0	0.967	1.0	0.967	1.0	1.0	0.933	1.0	0.967	0.967	1.0	0.967	0.967	1.0	0.867	0.967	0.967	1.0	0.933	0.967	1.0	1.0
f1	0.967	0.983	1.0	0.935	1.0	0.983	0.984	1.0	0.918	1.0	0.983	0.983	0.984	0.967	0.983	0.984	0.929	0.967	0.967	0.952	0.966	0.967	1.0	0.968

Figure 5.2:

5.1.3 Results

In Fig. 5.2 we present validation results (in the form of precision, recall and f1 score) for each letter. We also present in Fig. 5.3 a confusion matrix where on the horizontal axis we have the predicted labels and on the vertical axis we have the ground truth labels. We also report the following findings: Average Precision: 97.46%, Average Recall: 97.36%, Average F1 score: 97.41%, Top-1 Validation Accuracy: 97.36% and Top-5 Validation Accuracy: 99.86%. We also present in Fig. 5.4 some images from our model.

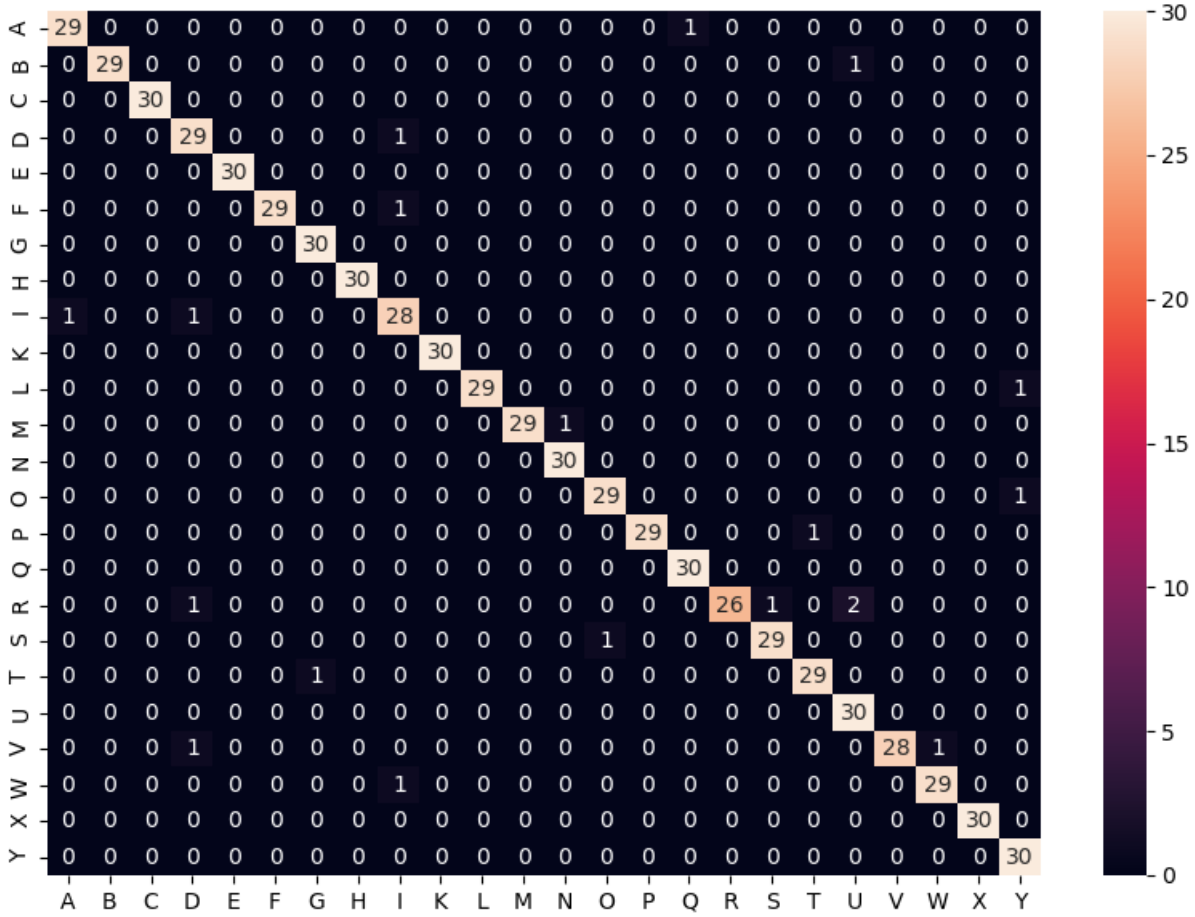

Figure 5.3:

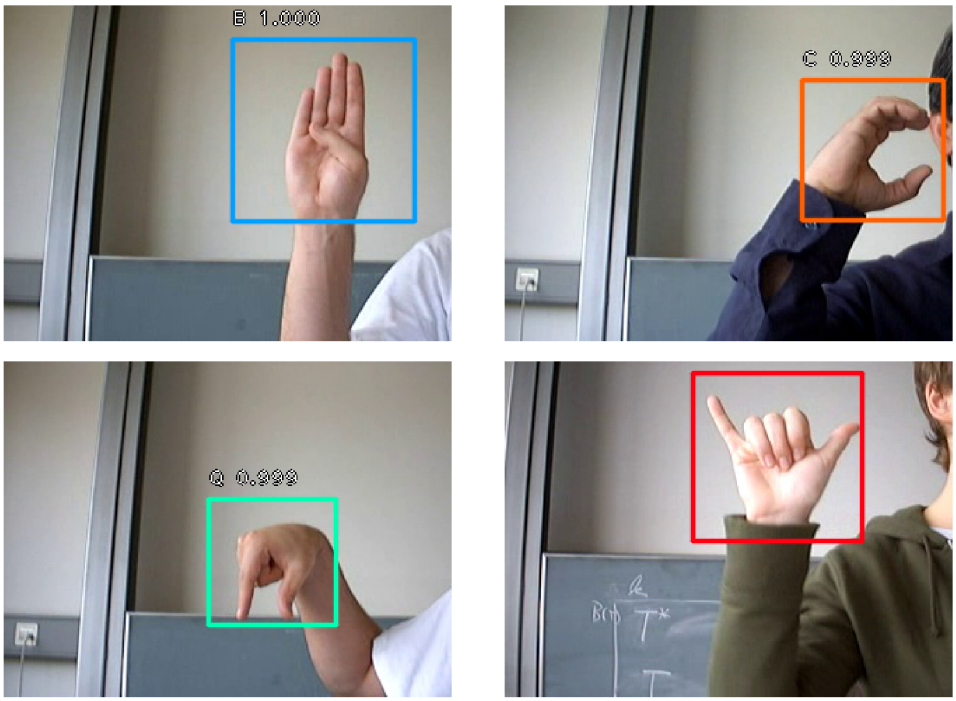

Figure 5.4:

5.1.4 Project documentationm

For the input we take a randomly sized image and output a set of probabilities for each label and its bounding box. However we only take the top score. So a normal output would be `(bounding_box(x1,y)),y2,label,score` where `x1,y1` are the upper left corner of the box and `x2,y2` are the lower right corner of the box, `label` represents the predicted label for the sign in the image, `score` represents the probability that `label` is in the image.

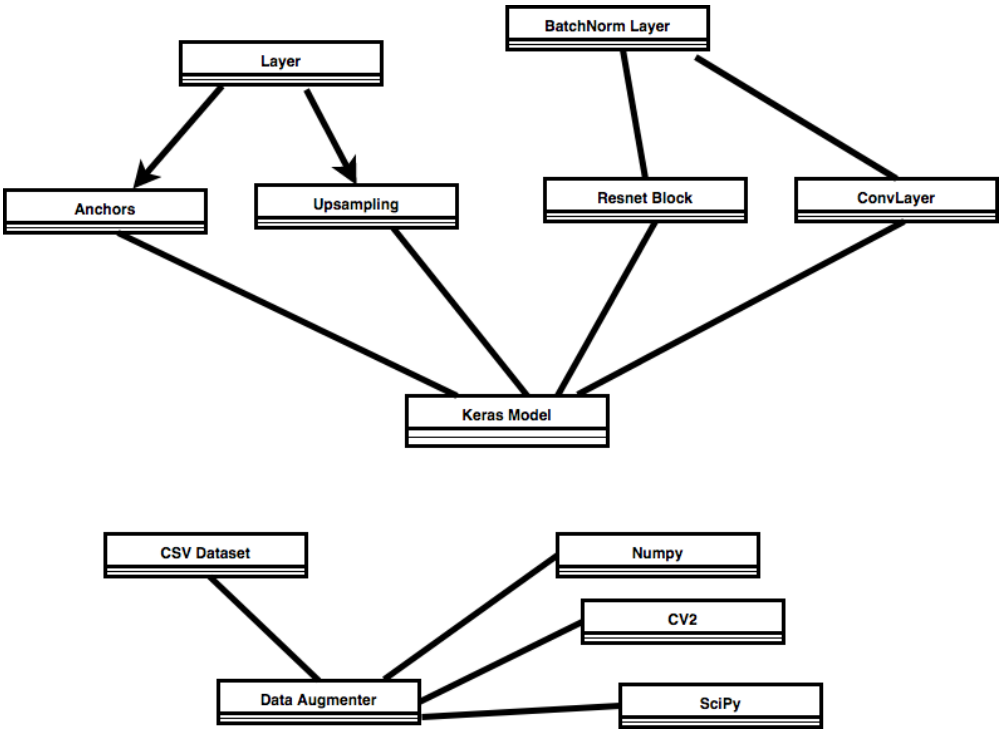

Figure 5.5: for the training and Diagram

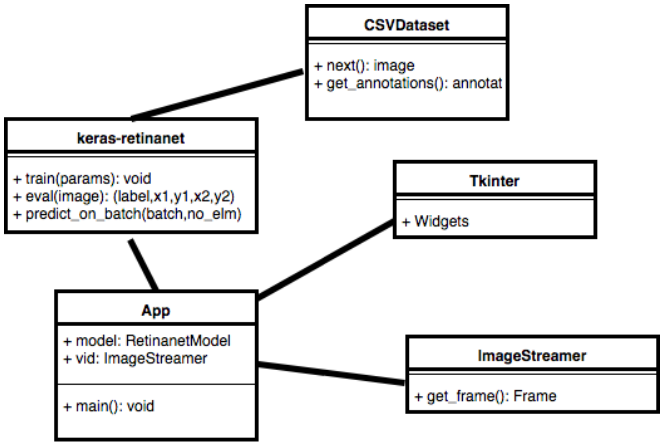

Figure 5.6: Gui architecture for inference

5.1.5 User Manual

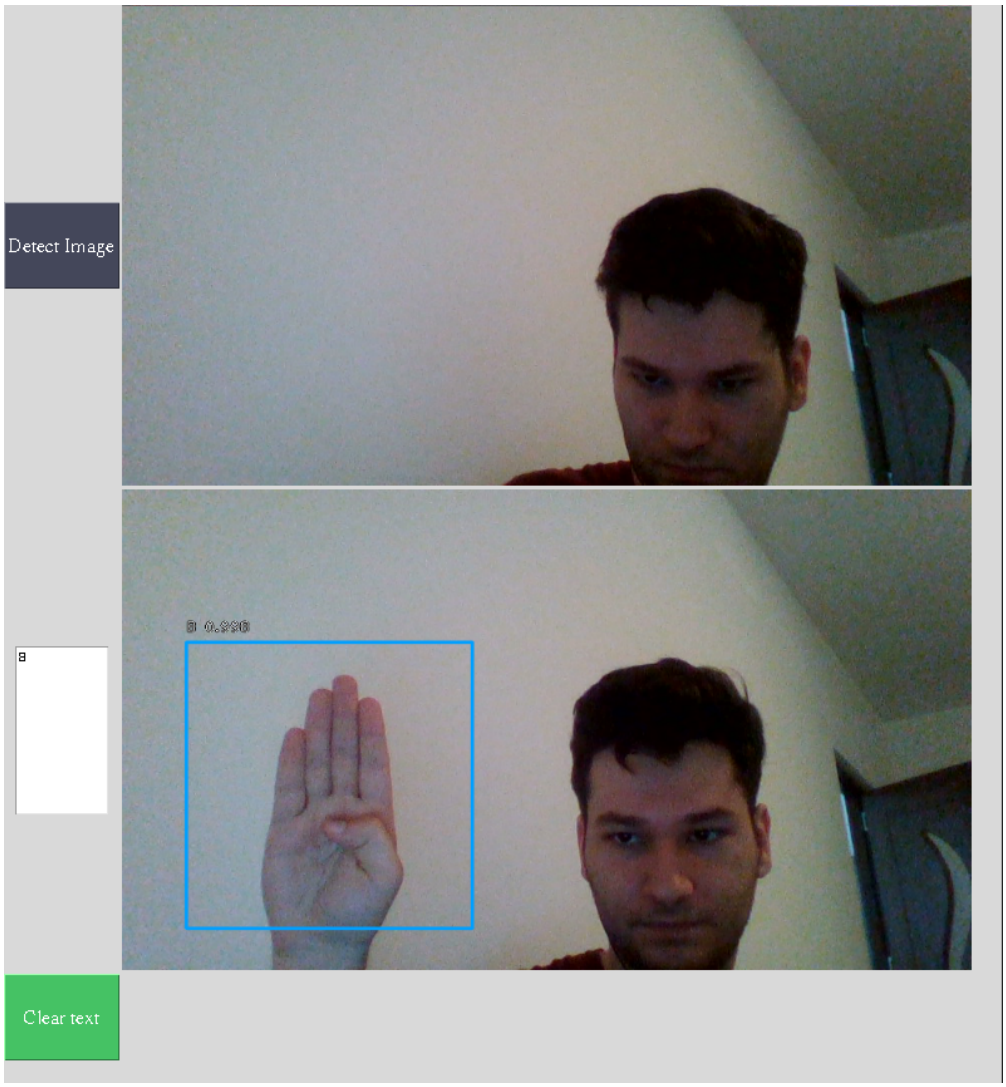

Figure 5.7: Screenshot of the application

In order to use the application one must install the following libraries using either pip or conda. `cv2`, `scipy`, `keras`, `tensorflow`, `numpy`, `PIL` and for the GUI interface `tkinter` . In order to use the model one must also install the keras-retinanet by the following commands:

```
git clone --recursive https://github.com/fizyr/keras-retinanet.git

cd keras-retinanet && pip install . --user

pip install --upgrade git+https://github.com/fizyr/keras-retinanet

pip install --upgrade git+https://github.com/broadinstitute/keras-resnet
```

The GUI is very simple and does not account for de complexity of the model used for sign-language detection. In Fig. 5.7 we can see a screenshot made from the app on which we will present the flow of the app. We start the app by the following command: `python3 main.py`. This will open a new window on which we have more components. Firstly, a continuous flow of frames can be seen in the upper right part of the window. We also have a *Detect Image* button. which on press takes a snapshot with the laptop's camera and runs the RetinaNet model on it. As a result we show the image with the drawings of the detection on it. We aslo put the letter detected into a text box so that one can form (not easily) words. If the *Clear text* button is pressed then the text box (used for accumulating the letters regarding a word) is emptied.

Chapter 6

Conclusion and future work

6.1 Conclusions

We present some conclusion after doing this project. Firstly, a trade-off between accuracy and speed must be made. As mentioned in above sections inference time for RetinaNet for a usual laptop is 1.5 seconds. However in practice one does not have this much time so we believe that by finding a more weaker model we could achieve a better inference time. We also propose that further work to be continued by using one stage detectors (because they have more fast training and inference time) such as SSD or YOLO v3 which currently overtakes RetinaNet in the COCO leaderboard and also shows to have a very low inference time.

Another thing is to expand the dataset. It seems that in practice RetinaNet does not perform as well as on the validation set. We also propose that further work is done on dynamic gestures because they comprise the lexicon of the American Sign Language. There is still a lot of work to be done in this area to ease the life of people with hearing deficiency.

Bibliography

[1] URL: https://github.com/mon95/Sign-Language-and-Static-gesture-recognition-using-sklearn.

[2] Lungociu Corneliu. "Real Time Sign Language Recognition Using Artificial Neural Networks". In: *Studia Univ. Babes-Bolyai* LVI.4 (2011).

[3] Tsung-Yi Lin Priya Goyal Ross Girshick Kaiming He Piotr Dolla. "Focal Loss for Dense Object Detection". In: (). DOI: https://arxiv.org/pdf/1708.02002.pdf.

[4] Brandon Garcia and Sigberto Alarcon Viesca. "Real-time American Sign Language Recognition with Convolutional Neural Networks". In: *System Theory (SSST) IEEE 43rd Southeastern Symposium 14-16 March 2011* (2011).

[5] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *arXiv preprint arXiv:1512.03385* (2015).

[6] Tsung-Yi Lin et al. "Feature Pyramid Networks for Object Detection". In: (2017), pp. 936–944.

[7] Weaver Joshua Pentland Alex Starner Thad. "Real-Time American Sign Language Recognition Using Desk and Wearable Computer Based Video". In: (1998).

[8] T.Starner and A. Pentland. "Real-Time American Sign Language Recognition from Video Using Hidden Markov Models". In: *Stanford Journal* (2014).