Deep Learning Based Home Security System for Low Power Embedded Devices

Emre Ozdemir

Abstract—In this work deep learning based home security system is created with low power Raspberry Pi computers. Nvidia Digits frameworks is used for model training where different training models are compared. Data acquisition is conducted by recording video and pre-processing for 3 different classes which produced 256 x 256 size 1400 pictures for training models. Trained model is deployed on a Raspberry Pi4 2GB computer with 5MP PiCam for inference. Lastly, conclusion and future studies are given.

Index Terms—Robot, IEEEtran, Udacity, LaTeX, deep learning.

1 Introduction

HOME security is a crucial part of modern civilization. Many methods exist for keeping your loved ones and belongings safe, such as moving to a secure apartment and locking the door. However, aim of this work is assuming reader forget all counter measures and needed home security system based on deep learning, which can recognize home owners and ”potentially” intruders in the future. Trained model can be deployed on a embedded device such as Nvidia Jetson or Raspberry Pi. -in this case devices that can keep working even during a power shortage is preffered.Trained model should be robust enough to recognize people and fast enough to be deployed on a low power CPU device, considering low FPS inference can be tolerable for static monitoring cases. First presupplied P1_data which consist of ”Bottle”, ”Candy Box” and ”Nothing classes” is tested on Nvidia Digits system.

1.1 Practicing With NVIDIA Digits

1.1.1 P1 Pre Supplied Data Set Results

P1 data contains 10000 pictures, with 3 different classes: Bottle, Candy Box and Nothing. Firstly, data set is created for classification task with, 75 percent training, 15 percent validation and 10 percent divisions. Then GoogleNet network is used for classification task with 30 epochs and 0.01 learning rate. Overall 75.4 percent accuracy is achieved with less than 5 seconds inference time as given in Fig 1.

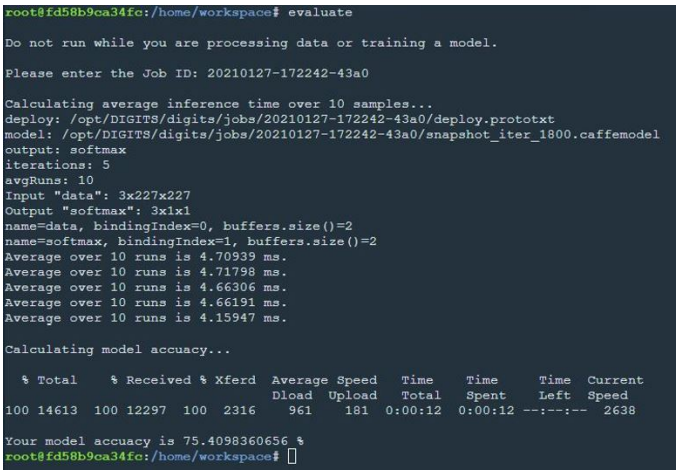

Fig. 1. Supplied Data Set Results.

2 Problem Definition and Data Acquisition

Since 2 people live in my house 3 different class is needed for home security system. 2 classes for 2 person (me and my girlfriend) and 1 class for nothing. For data collection I used my cellphone (an old iPhone SE) by recording 180 degree turntable videos of me and my girlfirend. Process is done for 3 different angles (face level, from higher and from lower) for 2 different lighting conditions (bright, dim). Each video is around 8-10 second which provides 80-100 pictures for each case.

Recorded videos require some preprocessing to be able to use in supplied training networks in Digits. For preprocessing ffmpeg and for renaming phtyon is used in

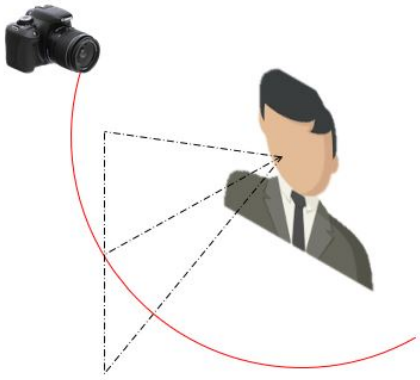

Fig. 2. Data Acquisition.

Win10. Supplementary codes and commands are given in the GitHub Repo.

1) 1920x1080 pixel raw video is cropped to 708x708 size which is enough to capture our faces.
2) 708x708 size video is resampled into 256 x 256 size
3) 256x256 video is splitted into 256x256 pixel picture series
4) Roughly 500 picture for each class is renamed with a python script to be able to use in Digits

Collected data consist of 1460 pictures with 3 channel RGB with following division:

- 516 picture for class ”Seda”
- 484 picture for class ”Emre”
- 460 picture for class ”Nothing”
- 1460 pictures overall

Fig. 3. Examples of Collected Pictures. Classes: ”Emre”, ”Seda” and ”Nothing” from left to right.

3 Model Training

Colleced data is uploaded into Nvidia Digits system and ”SEN_1460set” DataSet is generated. First data is shuffled and divided into 75 percent training, 15 percent validation and 10 percent test data. For training both GoogleNet and AlexNet is used. Epoch number is set to 50 for Alexnet and 80 for GoogleNet in order to observe the over-fitting points. Learning rate is set to 0.01 and SDG alghoritm is selected. LeNet is not used since it requires 28x28 inputs.

Training took 5 minutes for Alexnet and 22 minutes for GoogleNet. It can be seen that models starts to overfit after epoch number 20.

4 Hardware Deployment

Selected hardware for inference is Raspberry Pi4 2GB and 5MP PiCam with it’s 35 dollar price tag. CPU board is placed in a 3D printed air slotted cover with 20mm fan for keeping it cool.

Trained model is downloaded at Epoch number 17 snapshot. GoogleNet model is around 36 MB compared to AlexNet’s 216MB model which makes is much more easy to deploy on small hardware. OpenCV based Python code is prepared for feeding live video from PiCam to inference model. Here is a live feed snapshot taken from inference.

Alexnet performed much slowly as it requires more memory and deals with more parameters which cause more load on CPU. Used hardware is cooled with a fan which kept the CPU temperature around 50degC (without cooling Pi starts to overheat and eventually slows down due to thermal throttling).

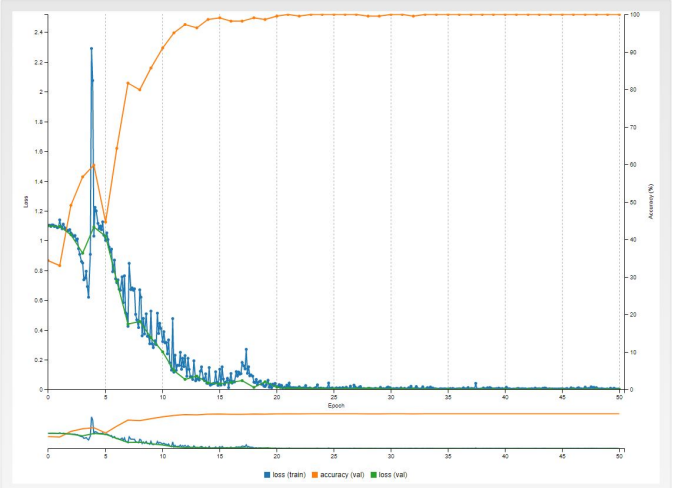

Fig. 4. AlexNet Results with SEN_1460set.

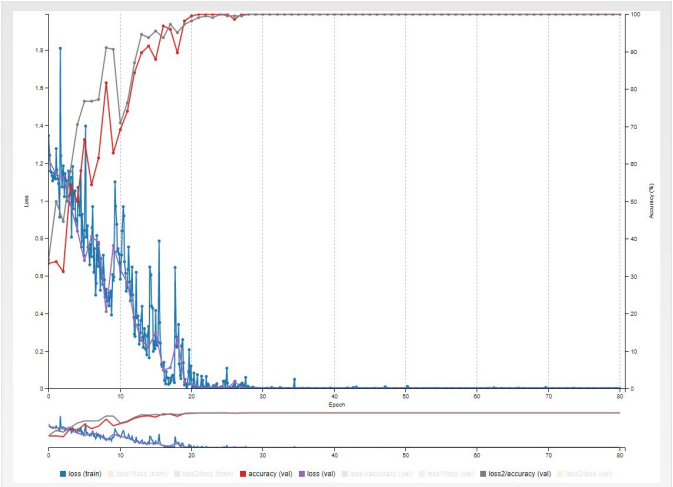

Fig. 5. GoogleNet Results with SEN_1460set.

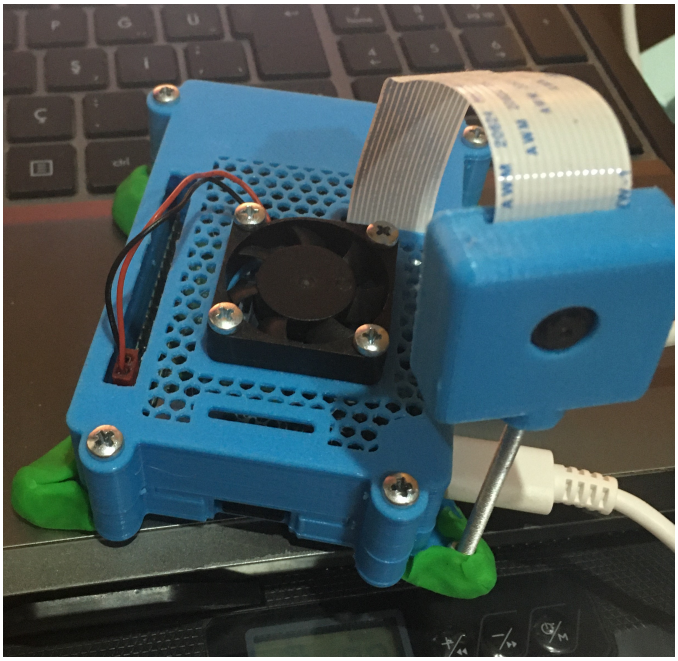

Fig. 6. Inference Hardware.

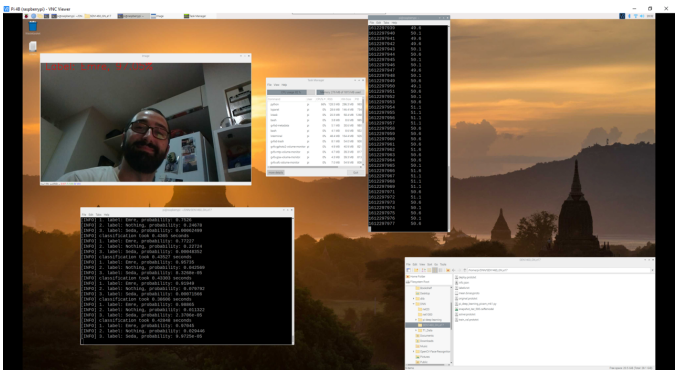

Fig. 7. Inference in Action.

Overall, GoogleNet model with trained 1500 picture performed very well in terms of recognition and performance trade-off.

5 Conclusion / Future work

Although this work is based on building a home security system, it is clear that 3 classes with 1500 pictures will not be enough for a robust product. Therefor, one who wants to take this study further should start with collecting more data with more classes and not only with faces.

For hardware deployment, best fit for this job is Nvidia Jetson with it's small size and high power. No fps data is collected for this study, but Raspberry inference performance is around 1 seconds. However in terms of cost vs performance, Raspberry is still a good option for hobby type applications.

Regarding with training model selection SquezeNet is a promising choice for small sized applications where model size is less than 0.5MB with good accuracy. As a future study researcher will investigate integrating SquezeNet into Digits framework.

Up until now, only classification tasks are investigated. However, digits has capability for detection and segmentation task which is something researcher want to investigate further.