Efficient facial detection on embedded system with CNN for surveillance tasks

Rafal Bielech
CICS
UMass Amherst
rbielech@umass.edu

Abstract

The ability for a system to perform facial detection is becoming increasingly more popular, as we rely on this technology for convenience and security in our everyday lives. Some of the products that are commercially available that are based on this technology are smart door bells with camera attachments, border control kiosks, photo editing filters, phone unlocking mechanisms and many more. Face detection has been an interesting area of research that is greatly benefiting from the advancements in artificial neural networks in order to complete the task with lesser hardware demands while improving the accuracy of the system. In this project, I adapted a new high-performance multi-task architecture with minor tweaks to recognize faces in a real-time in a video feed. The video stream will be used as input to the facial detection model in order to recognize faces and draw bounding boxes around the location with the highest confidence level. I intend to utilize the research and the architecture introduced in this project to introduce new functionality to surveillance and security tasks by integrating facial detection that has an ability to alert users in real time. The emphasis on efficiency is a high priority as the platform that the system is intended to run on is a Raspberry Pi 3B equipped with 1 GB of RAM and 1.2GHz ARM CPU with a 8 MP camera module strapped to the motherboard.

The next step of my project focuses on coupling the facial detection model with surveillance software in order to give a real world application for the project.

1. Introduction

This project aims to couple an exciting addition to existing surveillance tasks that can be utilized in an embedded system. Currently, most consumer surveillance systems do not offer any *smart* features in their product line in form of real-time surveillance footage analysis. Instead, the usual model for any form of surveillance tasks is to record video and store the footage in great quantities on large disks for further analysis, in case of an accident or adversity. One clear drawback with this approach is that huge quantities of storage are needed to store relatively low amount of footage. To put this into perspective, 500 megabytes could be sufficient to store about 10 minutes worth of video footage in 1920 by 1080 resolution just from one camera, however, this gets exponentially bigger when a cluster of cameras are added to the system.

When considering the task of surveillance work to detect human activity, only a small portion of the frames are actually important to the users. An applicable scenario for this could be in case of a home invasion and it is important to identify the suspects involved. This means that the large storage needs can be minimized greatly as we can now focus on storing the important and interesting frames instead of gigabytes of video footage. In order to improve the surveillance system even more, real time notification can be added to provide real time notification capability.

Problem statement – My goal is to design a facial detection model that can be an integrated with a real-time surveillance system that is running on a Raspberry Pi 3. Given the continuous camera live stream broken down frame by frame in native resolution (640 by 480 as lowest all the way to 1920 by 1080), it is my goal to efficiently and accurately draw bounding boxes with high confidence around the human faces. This means that each frame will be passed into the model. I am striving to achieve performance of > 2 fps with a high accuracy (> 70%).

Some of the foreseen difficulty of this application is introduced in the various angles that people could present themselves in a frame. Additionally, the model should be able to recognize faces that are further away, and not only be able to accurately determine faces that are close to the camera. Moreover, a poorly lit environment could pose issues in this application as the objects appear grainier and the features are hard to identify. The quality of the frames is also largely dependent on the camera quality and its ability to deliver clear pictures. Most Raspberry PI cameras range between 5 and 8 MP.

The model that is implemented in this paper is based on

a paper featuring an implementation of a multi task cascaded convolutional network [2] which breaks down the task of detecting faces into three stages. The motivation for choosing this particular architecture is based on Zhang et al's [2] claims of high accuracy and real time performance.

1. P – Net (Proposal Network) : First stage in the system that gives a large number of bounding box proposals to the later stages
2. R – Net (Refinement Network) : Second stage in the system that takes the output from P – net as input and filters out weak proposals and refines the bounding box dimensions
3. O – Net (Output Network) : Last stage in the system that further refines the output from previous stage and determines the most accurate bounding boxes for the detected faces in an image.

This paper is organized by introducing related works for facial detection, followed by an explanation of the model, training steps, validation, and conclusion.

2. Background on surveillance and Related Works and

Aside from trying to add facial detection capability to surveillance tasks, other approaches focus on background subtraction for analysis. Seiki et al. introduce a method that uses background subtraction between frames to detect changes in a scene by utilizing image vectors. One caveat with this approach is the probably higher number of interesting frames that will be produced in a real-time footage from a video. Aside from persons moving in a frame, the background subtraction will also flag frames as interesting if there is movement of animals, lighting, etc. Thus, facial detection discards any other movement aside from human movement in frames.

There have been several notable approaches in the area of facial detection. The MTCCN approached introduced by Zhang et al. [2] may appear like a refinement to other attempts for efficient real-time facial detection that has been experimented with in the past.

The pioneering work in facial detection seems to be carried out by Viola and Jones [9], where the authors were able to achieve results of 15 frames per second with comparatively low performing hardware, as the paper has been published in 2004. The model in this paper has been implemented with AdaBoost learning algorithm. The paper introduces a notion of utilizing cascades which is also a notable component of the model in the MTCNN paper [2].

Another approach that has greatly improved the efficiency of image feature detection was done with Faster R-CNN [11], which was an improvement over Fast R-CNN [7]. Faster R-CNN consists of a RPN and objection detection network. The RPN (region proposal network) produces an array of proposed bounding boxes that will be refined later in the second component of the network.

The multi-task approach is clear in the Viola and Jones approach as well as the Faster R-CNN approach. MTCNN's architecture is similar to those two previously mentioned approaches. The first proposal network focuses on generating a large number of bounding box proposals which are later refined in the further steps in the model.

3. Method

3.1 Architecture

The architecture of the network has been adapted from the paper "Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks" [2]. The networks consists of Proposal Network, Refine Network, and finally Output Network which has an ability to outputs the facial landmarks marking the eyes, mouth, and nose.

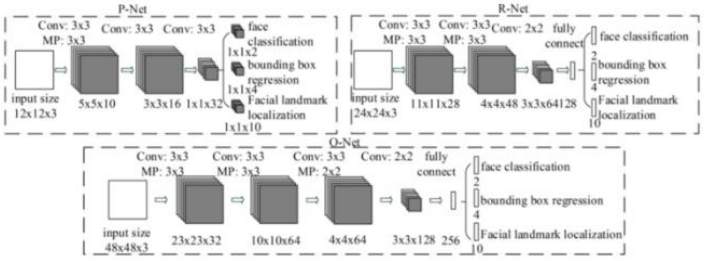

Figure 1 : The model consists of three separate networks with the results **cascading from P-net to R-net and lastly to O-net**

MTCNN image pyramid [2] to create multiple scaled copies of the original image. An image pyramid consists of the image that is scaled equally in the width and the height by some number less than one. The three scale factors that I have explored in this project are: 0.5, 0.6, and 0.7. The scale is then applied for the picture in the x and y direction

3.2 Training and Testing

Initially, I was using *Labeled Faces in the Wild* [1] dataset but since in this dataset most of the images were present in the middle of the 250 by 250 pixel image and most images only contained one face, therefore, using this dataset could lead to lower efficiency and accuracy for the model later on if the faces were found in other locations than the middle of the image.

The new dataset, *A Benchmark for Face Detection in Unconstrained Settings* [7], was used in the training of the model. This dataset contains images of people in various locations, with some photos only having one person, while

others having multiple people in the image with varying degrees of variation in posture, facial expressions, and conditions that the faces have been displayed.

To prepare the data for training/validation/testing, I selected the first 14,000 images from the dataset, having divided 12,000 images for training, 1,000 images for validation, and 1,000 images for testing. Based on my initial exploration of the data, I found that the images were diversified enough that no special techniques had to be involved to specially divide the images into their respective segments. For the purpose of this model, it caused no issues if images of the same persons were repeated.

Using OpenCV and Haar Cascades, I created an annotation file for the 14,000 images with each image being annotated in following format:
Image_file_path
[[x1, y1, x2, y2] // person 1
[x1, y1, x2, y2] // person 2
…
[x1, y1, x2, y2] // person n]
Since my model is based on creating rectangular bounding boxes instead of elliptical shape to bound the face. I was not able to find an annotation file for the data, thus, I created one with the help of existing facial detection tools.

3.3 Proposal Network

For every one of the scaled images that were created in the image pyramid, a 12 by 12 kernel is used to scan every part of the image with 2 stride. The architecture of the P-net includes three convolutional layers, three preLu activations functions, and one maxpool layer stringed together before the network splits off into two convolutional layers to produce the confidence probabilities and the bounding box locations. A softmax loss function is used to determine the probability of a face being found in the bounding box.

Zhang et al's [2] use non-maximum suppression (NMS) to merge bounding boxes that are largely overlapping into one bounding box. This is a method used to decrease the large number of bounding boxes that were produced by the kernel in the earlier steps. Figure 2 shows the large number of bounding boxes that were produced by this stage on an image from the LFW dataset [1]. Since the network has been training on the scaled coordinates from the images in the image pyramid, the coordinates are scaled back to the original image coordinates before they are passed onto the next stage.

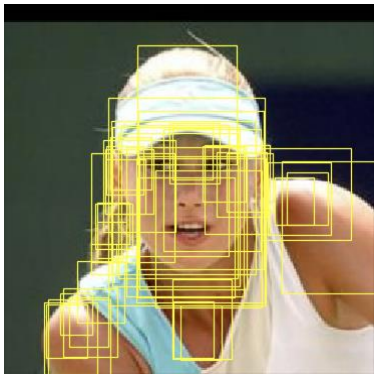

Figure 2: **Proposal network stage bounding box proposals** that were produced after testing on an image in LFW dataset [1]

3.4 Refinement Network

The bounding box locations computed from the previous network are now used as input to the refinement network. As the name implies, the input from the previous network is further refined to cut down the amount of proposed bounding boxes and only leave out the bounding boxes with the highest confidence. The refinement network is similar in structure to the Proposal Network, however, it is a bit longer with more layers. There are now three convolutional layers, with four preLU activation layers, and finished off with fully connected layers. Similarly as with the proposal network, the last layer branches off into two fully connected layers with a softmax loss function. The information found at this stage is the new bounding box locations and their corresponding confidence.

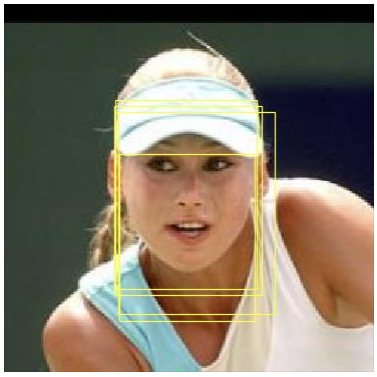

Figure 3: **Refinement network stage bounding box proposals** that were produced after testing on an image in LFW dataset [1]

Comparing the results from Figure 2 and Figure 3, there are drastic differences in how the number of boxes that are returned from the R – net and P – net. Figure 3 shows that a large number of bounding boxes that were off in Figure 2 have since been discarded, for example around shoulders and neck.

3.5 Output Network

As figure 1 shows, the Output network is even larger than the previous network. O-net consists of five preLu activation functions, three fully connected layers, four convolutional layers, and three maxpool layers. The output from the Refinement network is used as input to the Output network. Same as before, softmax loss function is used in one of the branching from the last layer. Similarly like with the previous networks, the bounding boxes with the lower confidence levels have been eliminated and using NMS, we are left with one bounding box per face in the image.

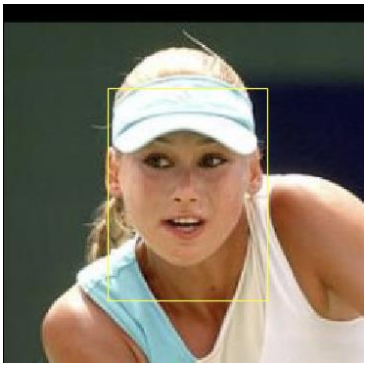

Figure 4: **Output network stage bounding box proposals** that were produced after testing on an image in LFW dataset [1]

Comparing figure 4 and figure 3, we see that the number of bounding boxes has decreased from four to one and the bounding box encompasses the whole face.

3.6 Setup

The implementation in this project has been done in the Python language with the use of NumPy and PyTorch. The latest version of PyTorch was used to build the network and the loss function. Cv2 played an integral part in this project for image manipulation. Cv2 was used for resizing images, processing images, and to draw bounding boxes around the results that were returned from the face detection model.

3.7 Tuning

In the validation step, the hyper parameters were tuned to achieve the best results. Ultimately, the learning rate was settled on 1*e-2 and the batch size of 128.

4 Evaluation and integration with surveillance system

The greater motivation for this project was to show that a facial detection system can be integrated for surveillance tasks in an embedded environment, such as a RPi in real-time. To facilitate this, I have implemented a quick surveillance system that alerts me in real-time over email when there is a face or multiple faces detected in a frame.

Figure 5 shows the final product with a web interface available locally by navigating to port 5555 of the RPI's IP address on the local network. The interface has been built with Flask and displays dynamic information about the RPi like uptime and CPU temperature.

The video feed from the camera module (seen on the bottom right corner of Figure 4 surrounded by two infrared lights for night surveillance) is parsed into a frames. Each frame is inputted to the face detection model to scan for faces. If there was a face found in the frame, the frame + 4 next frames are sent as an attachment over email alerting me in real time about a potential intruder on property.

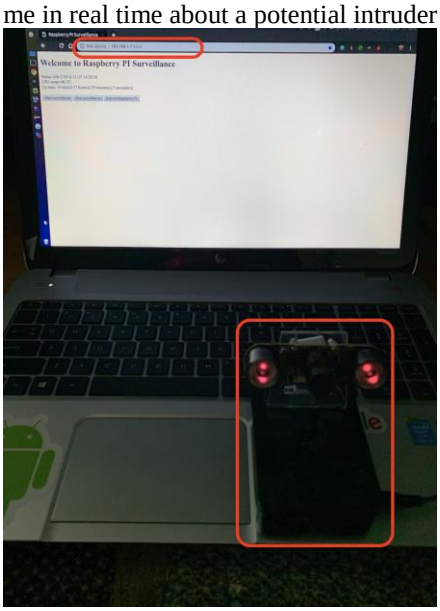

Figure 5 - **Prototype for Facial detection surveillance system**

Per the paper's claims [2], the facial detection can be used in real-time without too much of strain on the system. Given the RPi's limited hardware capability, the facial detection was able to be used with a real world result of about 2 frames per second on a 640 by 480 feed from the camera module. In my testing scenarios, there were only one or two people in each frame, as the computation cost would significantly increase on a higher resolution video feed and greater number of people per frame. In comparison, an i5 laptop with 16gb of RAM is able to achieve about 16 frames per second on a video feed from a camera.

Figure 6 shows the CPU usage percentage with the facial detection system being on and off for 60 minutes. The red line shows an average of more than 80 CPU % compared

to ~ 5 CPU % when the Pi was idle and the facial detection system was off.

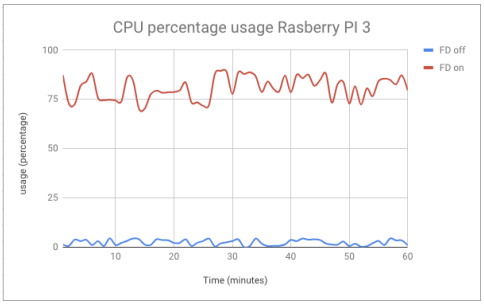

Figure 6 - **CPU usage effected by the facial detection model**

5. Evaluations

Statistics
Accuracy on finding correct number of faces = 84.7%
Accuracy IOU= 59.4%
FPS on laptop video feed: ~ 15 frames
FPS on Rpi video feed: ~ 2 frames
For my project, the most important evaluation metrics were the FPS and the accuracy of the model because the computation time is crucial for real-time analysis. The accuracy was calculated using Average Intersection over Union (IoU). IOU is the intersection of prediction from the model and the ground truth from the annotation file over union of prediction from the model and the ground truth. This was the chosen approach for accuracy evaluation to take account of deviation between the annotation file's ground truth labels and the model's returned bounding box locations. Figure 7 [12] shows a visual of what kind of information the IoU returns.

Accuracy in this evaluation is a metric testing the model's ability to detect the correct number of faces in a dataset irrespective of the locations. This encompasses false positive and false negatives.

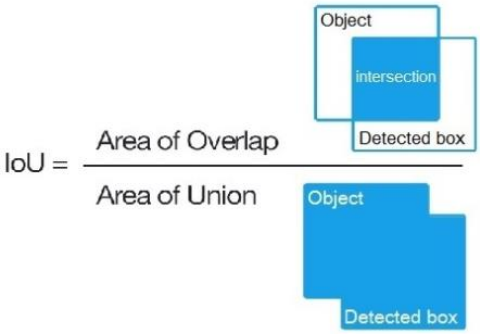

Figure 7: **IOU visualization**

I have further broken down the computation time for each of the networks on the testing image set.

Average Proposal Network cost = .157 seconds
Average Refinement Network cost = .096 seconds
Average Output Network cost = 0.017 seconds

As we can see, the proposal network cost has the largest computation time, which is largely due to the large number of bounding box propositions and the small kernel (12 x 12). Further, refinement network cost is significantly lower followed by an even lower output network cost.

6. Conclusion

Efficient facial detection is an highly interesting area of research, especially now with a large application of facial detection for numerous reasons. In this project, I have implemented a method for facial detection that can be integrated for surveillance tasks with a high accuracy of about .85 and about 2 FPS on a Raspberry PI 3B. Some of the other more popular approaches include LTSM [5], Faster R - CNN [3] and YOLO (You only look once) to name a few. I have selected to implement a multi-task cascaded convolution network due to the high accuracy in that has been noted in the paper [2], while maintaining good performance. I have made some small innovations to the model introduced in order to optimize it for my application. I found that in the original paper [2], each network also returns facial landmark localizations in the last branching, but I have chosen to discard this extra layer in the network since I am not interested in showing the localizations in the frames.

Even though I have achieved good performance and accuracy in this project, there are some improvements to be made. One performance improvement that can be made is to decrease the frequency of studied frames that the model analyzes. In this project, every frame is a candidate for the neural network; however, if we let every other frame to be selected as a candidate for the model, this might have favorable advantages for the performance.

7. Future Work

With the current status of the project, I was able to detect faces in a video when dissecting the video frame by frame and using each frame as an input to the model to draw bounding boxes around the faces in the image. If there are faces detected in an image, separate thread in the surveillance program will take care of the correspondence with the user for alerting in real-time. A possible extension to this project is to perform object tracking (face as the object in this case) as the person is in the view of the camera. This would allow the system to start tracking the person once they have been identified by the model introduced in this project. A large advantage of this

extension is that the system might have awareness of the person if they are in view of the camera, even if their back is turned to the camera and the face is no longer visible. Integration with a Faster R-CNN [3] model could be done to perform facial tracking in the images.

Another proposition for extending of this project is to take facial detection forward to do facial recognition. One approach is to utilize the facial feature locations of the detected faces as a means of performing facial recognition [4]. One of the outputs from the Output Network (O - net) in this project returns the locations of the eyes, nose, and mouth for each detected face. Having these locations, this project could further be expanded with the mentioned previously [4] to compare the distances between these five locations and try to predict the person associated with each face. One huge advantage of this is that the model could be more equipped to distinguish between interesting finds (faces not recognized) versus non - interesting finds (recognized faces), as recognized faces would not pose a threat in the surveillance scope of the project.

References

1. Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments. University of Massachusetts, Amherst, Technical Report 07-49, October, 2007.

2. Zhang, Kaipeng, et al. "Joint face detection and alignment using multitask cascaded convolutional networks." IEEE Signal Processing Letters 23.10 (2016): 1499-1503.

3. Jiang, H. and Learned-Miller, E. (2016). Face Detection with the Faster R-CNN. [online] Arxiv.org. Available at: https://arxiv.org/pdf/1606.03473.pdf [Accessed 25 Nov. 2018].

4. Rhesa Septian Siswanto, A., Satriyo Nugroho, A. and Galinium, M. (2014). Implementation of face recognition algorithm for biometrics based time attendance system - IEEE Conference Publication. [online] Ieeexplore.ieee.org. Available at: https://ieeexplore.ieee.org/abstract/document/7013165 [Accessed 25 Nov. 2018].

5. Lai, Hanjiang et al. "Deep Recurrent Regression for Facial Landmark Detection." IEEE Transactions on Circuits and Systems for Video Technology 28.5 (2018): 1144–1157. Crossref. Web.

6. Viola, P. & Jones, M.J. International Journal of Computer Vision (2004) 57: 137. https://doi.org/10.1023/B:VISI.0000013087.49260.fb

7. R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, OH, 2014, pp. 580-587. doi: 10.1109/CVPR.2014.81

8. Vidit Jain and Erik Learned-Miller.
FDDB: A Benchmark for Face Detection in Unconstrained Settings.
Technical Report UM-CS-2010-009, Dept. of Computer Science, University of Massachusetts, Amherst. 2010.

9. Paul Viola and Michael J. Jones. 2004. Robust Real-Time Face Detection. Int. J. Comput. Vision 57, 2 (May 2004), 137-154. DOI: https://doi.org/10.1023/B:VISI.0000013087.49260.fb

10. M. Seiki, H. Fujiwara, and K. Sumi. A robust background subtraction method for changing background, 2000.

11. Ren, Shaoqing et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks." IEEE Transactions on Pattern Analysis and Machine Intelligence 39.6 (2017): 1137–1149. Crossref. Web.

12. Rosebrock, Adrian. "Intersection over Union (IoU) for Object Detection." PyImageSearch, 24 June 2018, www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/.

13. Kaipeng Zhang, MTCNN_face_detection_alignment, (2017), GitHub repository, https://github.com/kpzhang93/MTCNN_face_detection_alignment

14. Rosebrock, Adrian. "Non-Maximum Suppression for Object Detection in Python." *PyImageSearch*, 2 Aug. 2018, www.pyimagesearch.com/2014/11/17/non-maximum-suppression-object-detection-python/.