Face Recognition Using Deep Learning and TensorFlow

Nihaal Maipady Srihari, Dept. of Computer & Information Science & Engineering, University of Florida, Gainesville, USA

[nihaal.maipadysr@ufl.edu](mailto:nihaal.maipadysr@ufl.edu)

*Abstract*— Face Recognition (FR) is a biometric technology that identifies people based on their facial features. FR is used in a variety of applications such as video surveillance, ID verification in offices, criminal identification and security systems. Many of the present-day approaches to Object Detection are based on the R-CNN and the Fast R-CNN frameworks. The drawbacks of these approaches are that they are time consuming and expensive in space and time. This paper proposes a methodology to distinguish and recognize individual faces using TensorFlow’s Object Detection API by using the faster R-CNN model to train the network.

*Index Terms*— Face Recognition, TensorFlow, R-CNN, fast R-CNN, faster R-CNN

# Introduction

We are proposing a methodology to detect and distinguish individual faces using TensorFlow’s Object Detection API. The input to the classifier is a color image. The faster R-CNN model is used in this work as it has good accuracy as compared to other models. We are using the inception v2 architecture as the feature extractor. The desired output would be an image with boundary boxes drawn around the faces and a number showing the confidence score.

# Dataset

To build a robust classifier, the dataset must have images with different lighting conditions, backgrounds, noise and expressions. We are using the Frontal Face dataset collected by Markus Weber at California Institute of Technology. The dataset consists of 450 face images with dimensions 896 x 592 pixels in jpeg format. The pictures are of 25 unique people under different lighting/expressions/backgrounds. There are approximately 20 pictures of each person. We will bifurcate the train and test data in an 80-20 split respectively.

# Related Work

The object detection problem has two parts namely, object localization and object classification. Object localization refers to the process of drawing bounding boxes around the objects in a given input image thereby locating the objects in the image. Object classification involves identifying the categories into which the objects in the image belong. Thus, object detection is the process of drawing bounding boxes around the objects and classifying them into their respective categories.

The genesis of deep neural networks can be traced back to 1949 with the attempt to simulate the working of the human brain[10.warren]. In 1986, Rumelhart et al.[] described a new learning procedure called back-propagation. In the following decades, research on neural networks stagnated due to the lack of computational power, issues with overfitting and lack of large training datasets. In 2012, AlexNet proposed by Krizhevsky et al. won the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) competition. It used the ReLU non-linearity units instead of the tanh activation function which increased the speed. Data augmentation and dropout was used to mitigate the issue of overfitting. This was followed by the introduction of many architectures viz., GoogleNet, 152 layer ResNet, VGG. We are using an improved version of GoogleNet called the inception v2 in this work. Thus, Convolutional Neural Networks (CNN) became one of the most sought-after deep neural network architectures.

# CNN Architecture

All of the CNNs mentioned above and many others share similar network architectures. An image is served as an input to the CNN. The CNN consists of several layers of convolution and pooling layers. This is followed by a couple of fully connected layers and a softmax function for classification and a regressor for drawing the bounding boxes. The output will be the image with a bounding box around the objects and a confidence score.

An image may contain many objects thus requiring many boundary boxes being drawn. In a traditional CNN, we must know the number of classes beforehand. If we proceed by building a traditional CNN for a problem where the number of classes are unknown, we must come up with a large number of regions of interest. The objects may be present at various locations and may have different aspect ratios. This is not desirable method as the problem will computationally blow up.

To mitigate the above problem, Ross Girshick et al[2015]. proposed a Region-based Convolutional Neural Network: R-CNN.

The proposed R-CNN has three modules:

* *Region Proposition*

Generate region proposals by using selective search algorithm on the image. Around 2000 regions were generated using this technique.

* *Feature Extraction*

Each of the 2000 regions are then compressed into a square and fed as input to a CNN. The AlexNet CNN used here will produce a 4096-dimensional output feature vector.

* *Classification*

The output feature vector produced by AlexNet is fed as input to the linear SVM classifier which classifies every object into a particular class. The SVM must be independently trained for every class. A regression model is also present to reduce localization errors.

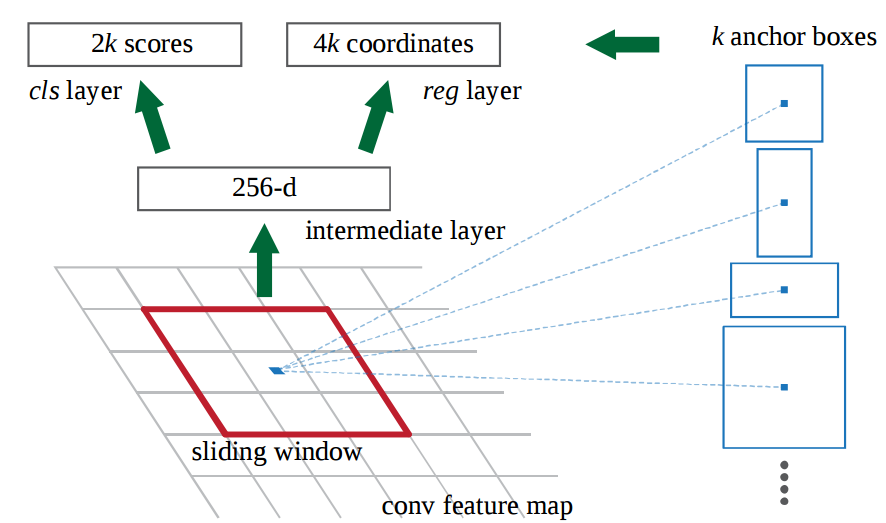
Though R-CNN is a great improvement from the naïve approach, there are many drawbacks. Firstly, these are three independent modules which must be trained separately and share no computation. Secondly, a feature map must be generated for each region in an image which is time consuming. So, to unify the three modules, Ross Girshick et al[], proposed an improved model called the Fast RCNN model.

In the Fast RCNN model, instead of feeding the CNN with the region proposals, the image is itself fed as input to the CNN. The CNN processes the image to produce output feature maps. Then, the regions are identified, warped and sent through a RoI Pooling layer from which we can extract fixed size feature vectors. These feature vectors are the fed to fully connected layers which then branch off into a soft max layer for classification and a regressor for drawing the anchor boxes around the regions of interest. Fast RCNN is faster than R-CNN owing to the fact that the feature maps are generated only once per image as opposed to producing a feature map for every region in the image.

*Faster R-CNN*

Both R-CNN and the Fast R-CNN use the selective search algorithm to propose regions of interest to the CNN. One of the performance bottlenecks for the Fast R-CNN model is the region proposal module, more specifically the selective search algorithm which is slow and time consuming. To solve this issue, Ren et al. proposed an improved Region Proposal Network (RPN). The Faster R-CNN model has two modules: The Region Proposal Network and the Fast R-CNN module.

Initially the image is passed through a CNN, in this case, an Inception v2 architecture. The output of the CNN is a feature map. This feature map is fed as input to the RPN. A sliding window is moved across the feature map to obtain a low dimension (256-d) feature vector. For each location of the sliding window, the RPN generates many regions of interest. The center of the sliding window is a number of anchor boxes. Anchor boxes of different dimensions 1\*1, 1\*2, 2\*1 are used to detect various objects of different sizes and shapes. The RPN is also used to predict the “objectness”- a binary value, in the image, i.e whether an object is present in the image or not.



The classification layer carries the 2k scores of possible objects for each proposal, where k denotes the sum of the total possible locations of the object. Only those regions whose “objectness” scores are beyond a pre-determined threshold will be passed through to the next layers. The novelty of this model is that the number of region proposals are brought down from 2000 to 300 proposals per image. The regression layer carries 4k coordinates, which is used to predict the bounding boxes around the image. Due to large number of regions, there will be large number of anchor boxes that overlap each other. To delete overlapping anchor boxes, Non- Maximum Suppression is employed. The anchor boxes with lower Intersection over Union (IoU) values are deleted. The formula for calculating IoU values is given by the below equation.

(1)

The Intersection over Union provides a measure of the accuracy of the predicted boundary boxes as compared to the ground truth.

The region proposals are subsequently passed to the Fast R-CNN module. These regions are reshaped using the RoI Pooling layer. The pooling layer is used to downsize the image. The image size is reduced to prevent overfitting and to reduce the training time, thus increasing the efficiency of the model. Max-Pooling is used in this case. A window of n\*n dimension slides across the feature map and extracts the maximum value in the window. Hence, we retain the most important features in the image, in addition to reducing the size of the image.

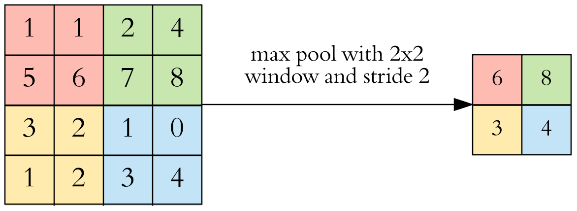


Fig 2. Max-Pooling

The concept of max pooling is depicted in Fig 2.

After the RoI Pooling layer, we have a bunch of Fully Connected (FC) layers which accept a 1-d vector as input. As the output of the RoI Pooling layers are 3-d vectors, we must flatten the output of the final pooling layer to a 1-d vector. These feature vectors are fed as input to the softmax activation function and the regression module. ReLU (Rectified Linear Unit) is applied to introduce non-linearity.

The complete Faster R-CNN model is shown below in Fig. 3.

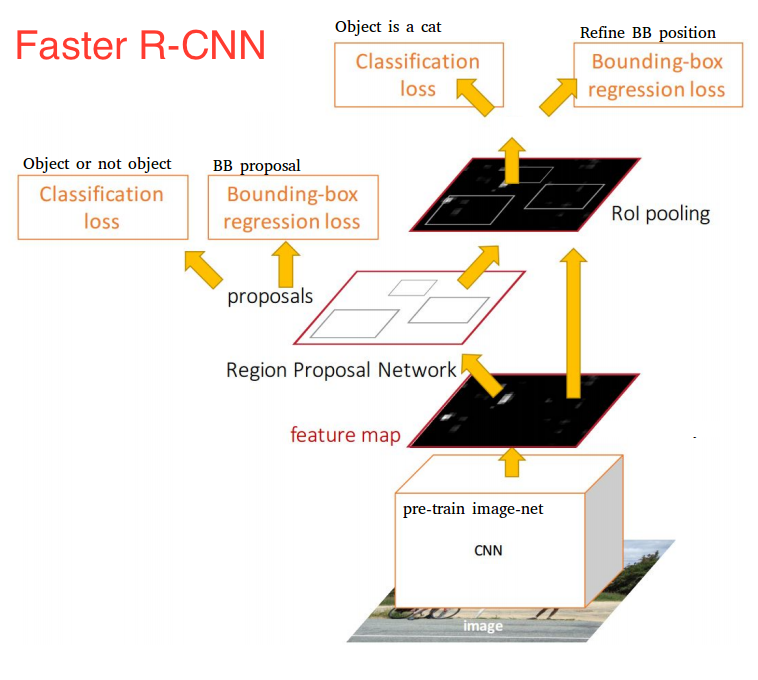


Fig. 3: Faster R-CNN model

# Methodology

## Model Selection

There are many pre-trained models offered by TensorFlow. A suitable model must be selected based on the hardware being used. Figure 4 shows a snippet of different models available. In the figure, mAP (mean avaerage precision) indicates the performance of the model when tested on the COCO dataset. The Faster R-CNN Inception V2 model was selected for this work owing to its high accuracy.

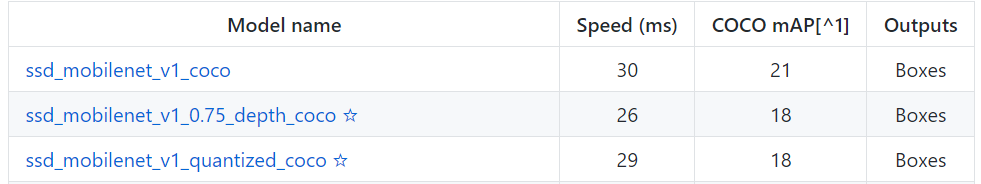


Fig. 4: Pre-trained Models in TensorFlow

## Dataset Labelling

We used LabelImg tool for labelling the input images. LabelImg saves the image as an XML file with the labels for each image in the dataset. These XML files contain the coordinates of the bounding boxes for each object in the image. These xml files when converted to csv files provide the ground truth of the bounding boxes in the images.

## Generate TFRecords

The TensorFlow’s Object Detection API requires that all the labelled training data be in TFRecord file format. To achieve this, we must first convert the xml files into csv files. Further, a python script is written to convert the csv files into TFRecords. In the tf\_record generator file, specify the class names and their corresponding class\_id. The TFRecord (.record) files are fed as input to the training model.

## Create Label Map and Configure Training Model

The next step is to create a label map according to the input dataset. The label map defines a mapping of class name to class id. We have 11 classes P1 to P11 defining Person 1 through Person 11. Further, we configure the face detection training pipeline. This defines the models and the parameters used for training and points to the training images and data. We must write a configuration file indicating the path to the tfrecord file and the label map.

## Training the Model

A python script is written to train the model and provide the path to the training data directory and the object detection pipeline configuration file. We have used a total of 200 pictures of 11 people to train the face detector. Then, we used TensorBoard which is a tool that provides a way to visualize the key metrics in the training process such as loss and accuracy. We can also view histograms of weights, biases, or other tensors that change over time. Run the python script to start training the classifier. Once the classifier reaches a desirable amount of loss (under 0.05) consistently, the object detection classifier is ready for testing.

## Testing the Model

Once the model is trained to the desired level, the classifier is used to test the images. During training, the classifier is saved at specific checkpoints. The checkpoints can be obtained from the training folder. Then, we must export the frozen inference graph. This frozen\_inference\_graph.pb file is our object detection classifier. Write a python script to test the data. The data to be tested is provided as an image in the directory itself. We will be using open-cv and pillow for image processing. The output is an image. The image will have a boundary box around the it and a number denoting the confidence score. The confidence score which is between 0 and 1 depicts the confidence of the classifier that the image belongs to a particular class.

# Results and Discussion

## Data Processing

We provided a total of 200 color images with dimensions 896 x 592 pixels in jpeg format as training data. The images were of 11 persons each named from P1 through P11. 80% of the images were provided as test data and the remaining 20% was provided as test data. As sample input image is provided in Fig 5.



Fig. 5: Sample Input Image

## Results

The Faster R-CNN model was trained to 45000 steps. The learning rate is . Below figures depict the results. We have designed the classifier to detect the faces of people that the classifier was trained on. The classifier correctly predicts the classes of the images. A sample of the results are provided below. The class names and the confidence scores are specified in the last line **“[{b'P1': 0.999463}]”**. The class is P1 and the confidence score is 0.999463. As we can see the confidence score is high for the test images. This is a testament to the high accuracy of the Faster R-CNN Face Detection model.

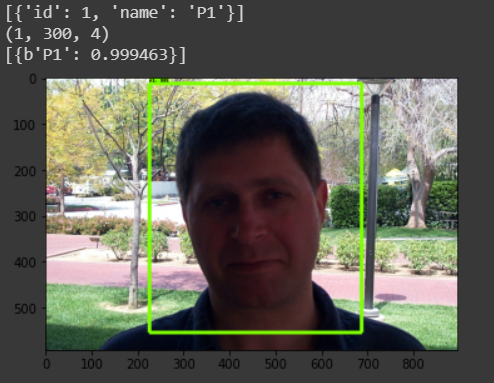


Fig. 6: Sample Output Image

Figures 7, 8, 9 show the results depicting the same person “P9”.

The images used for testing were taken in different lighting conditions, facial expressions, various gestures, partial face covering and multiple backgrounds with different objects. This shows the versatility of the classifier. There is a bounding box that was drawn by the regression module to detect the face in image. On the edge of the bounding box, there is a confidence score depicting the confidence of the classifier that the image belongs to the particular class.



Fig. 7: Output Image of person P9

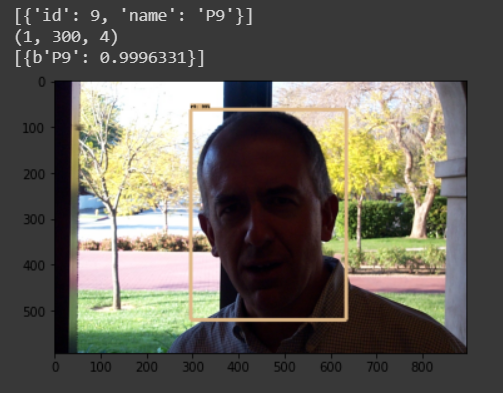


Fig. 8: Output Image of person P9

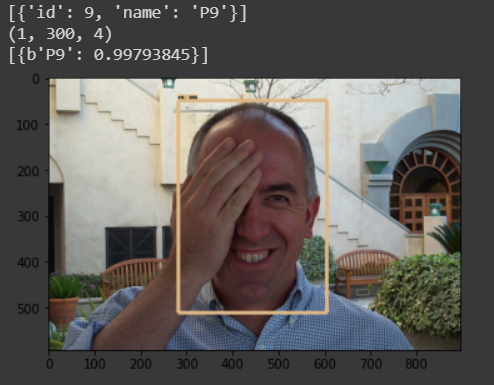


Fig. 9: Output Image of person P9

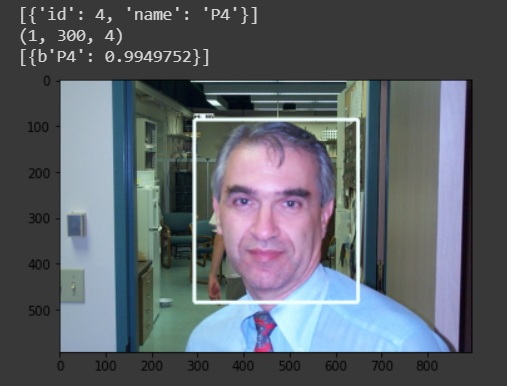


Fig. 10: Output Image of person P4

The results of the work are tabulated and presented in Table 1.

The confidence score is shown as a percentage in the table below.

|  |  |
| --- | --- |
| Person Name | Confidence Score (%) |
| P1 | 99.94 |
| P2 | 99.92 |
| P3 | 99.95 |
| P4 | 99.49 |
| P5 | 99.88 |
| P6 | 99.96 |
| P7 | 99.77 |
| P8 | 99.87 |
| P9 | 99.81 |
| P10 | 99.91 |
| P11 | 99.86 |

Table 1: Performance of the Face Detection Model

References

1. Mei Wang, Weihong Deng, “Deep Face Recognition: A Survey” [*arXiv:1804.06655 [cs.CV]*](mailto:https://arxiv.org/abs/1804.06655)
2. M. Coşkun, A. Uçar, Ö. Yildirim and Y. Demir, "Face recognition based on convolutional neural network," *2017 International Conference on Modern Electrical and Energy Systems (MEES)*, Kremenchuk, 2017, pp. 376-379.
3. Krizhevsky, Alex & Sutskever, Ilya & Hinton, Geoffrey. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Neural Information Processing Systems*. 25. 10.1145/3065386.
4. Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems* (pp. 91-99).
5. Girshick, R. (2015). Fast r-cnn. In Proceedings of the *IEEE International Conference on Computer Vision* (pp. 1440-1448).
6. H. Jiang and E. Learned-Miller, "Face Detection with the Faster R-CNN," *2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017)*, Washington, DC, 2017, pp. 650-657.
7. Hu, Guosheng, Yongxin Yang, Dong Yi, Josef Kittler, William Christmas, Stan Z. Li, and Timothy Hospedales. "When face recognition meets with deep learning: an evaluation of convolutional neural networks for face recognition." In Proceedings of the *IEEE International Conference on Computer Vision Workshops*, pp. 142-150. 2015.
8. J. Hosang, R. Benenson, P. Dollar, and B. Schiele. What makes for effective detection proposals? *arXiv:1502.05082*, 2015
9. C. Szegedy, A. Toshev, and D. Erhan. Deep neural networks for object detection. In *NIPS*, 2013.
10. Markus Weber, Frontal Face Database, California Institute of Technology,1999, <http://www.vision.caltech.edu/html-files/archive.html> [Online]
11. Z. Zhang, Y. Wang, J. Zhang and X. Mu, "Comparison of multiple feature extractors on Faster RCNN for breast tumor detection," *2019 8th International Symposium on Next Generation Electronics (ISNE*), Zhengzhou, China, 2019, pp. 1-4.
12. <https://towardsdatascience.com/what-is-deep-learning-and-how-does-it-work-f7d02aa9d477> [Online]
13. <https://towardsdatascience.com/deep-learning-for-object-detection-a-comprehensive-review-73930816d8d9> [Online]
14. <https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365eY> [Online]