YOLOnet

Will Lauer and Hannah DeBalsi
wlauer@stanford.edu, hdebalsi@stanford.edu

March 23, 2018

Abstract

We explored the problem of face recognition using a hybrid model composed YOLO for face detection, and FaceNet for face classification. Our model, YOLOnet, is intended to offer real-time face recognition, combining YOLO's real-time production of bounding boxes with FaceNet's fine-grained distinction. By replacing the face detection stage of FaceNet with a binary classifier version of YOLO, we hoped to increase the speed of face recognition without compromising the accuracy of FaceNet. The baseline, an unmodified FaceNet model run on Labeled Faces in the Wild images, had a test set accuracy of 98.8%, while our trained YOLOnet pipeline had a comparable test set accuracy of 98.7%. In addition, the baseline FaceNet model took approximately 11.75 minutes, including detection and classification, to test on 500 pairings of full LFW images. In contrast, our model took approximately 6.5 minutes to test on 500 pairings of the same LFW images that had been cropped to thumbnails corresponding to bound boxes around faces. Since YOLO offers real-time performance for bounding box generation in place of the 5.5 minutes of the 11.75 total minutes needed for the baseline model to detect faces, we conclude that our model offers comparable accuracy with a noticeable speedup.

1 Introduction

In our years of watching Game of Thrones, few things have become more frustrating than the friend who still refers to Daenerys Targaryen as "the dragon lady." To do our part in rectifying this problem, we wanted to implement an algorithm that is successful at quickly locating and identifying faces in images. The task is similar to that of other object detection challenge, but with a greater focus on fine-grained distinctions between people, and working with smaller amounts of available data. Beyond helping viewers of Game of Thrones with character recognition, this project has a variety of use cases, from identifying characters in other confusing television shows to Snapchat filters, automatic tagging on Facebook, and security systems. In this paper, we will discuss our model, our results, and potential for future work.

2 Related work

There are numerous existing algorithms for object detection, and we wanted to explore the options to find the fastest model for the task of face detection. In terms of speed, algorithms such as RCNN, Fast RCNN, and Faster RCNN all face performance bottlenecks due to slow region proposal methods. Thus, we opted in favor of YOLO for our region proposals. It offers real-time performance on object detection tasks, very similar to our goal of face recognition.

One downside of YOLO is that it struggles to generalize to variant ratios or configurations, which is not ideal for a task like facial recognition where faces can have a wide variety of sizes, angles, and lighting. Thus, rather than solely using YOLO for both detection and classification, we opted for a more robust algorithm for the problem of face classification.

While YOLO struggles to perform on fine-grained distinction, FaceNet performs very well in this, with regard to faces[4]. FaceNet is traditionally trained using triplets of pictures: an anchor, a positive image, and a negative image. The model trains by minimizing the difference between the embeddings of the anchor and positive images and maximizing the difference between the embeddings of the anchor and negative images, known as triplet loss. In *A Discriminative Feature Learning Approach for Deep Face Recognition* [5], an alternative loss function for FaceNet training is proposed, and our model uses this loss function. The alternative loss function combines a softmax cross entropy loss with center loss. Center loss forces the model to learn a center for the deep features of each class, and minimizing the center loss causes the model to minimize the difference between each deep feature and its center. The center of a deep feature is a vector of the same dimension as the given deep feature. Traditional triplet loss requires creating triplets of images. If these triplets are poorly chosen, the model may be slow to converge, and the computational complexity and data labeling procedures increase if the triplet are carefully chosen. Thus, the authors of the paper argue that center loss is an efficient way to force the model to minimize the difference between the deep features of a class, and when combined with softmax cross entropy loss, the model is forced to both minimize the difference between the deep features within a class and maximize the difference between features between classes.

3 Dataset and Features

For this project, we used two datasets, Labeled Faces in the Wild (LFW) and the WIDER Face Dataset. The former consists of headshots of various celebrities, with pairings identifying positive relationships for Facenet. The latter dataset consists of multiple people in each image, with annotations detailing bounding boxes around their faces. This was our input to YOLO, to learn the problem of face detection. We fed variants of LFW images to David Sandberg's Facenet implementation.

4 Methods

Our algorithm combines the YOLO algorithm and the FaceNet model. We use the YOLO algorithm for binary face detection and the FaceNet model for face classification. The input to the model is an image, which is run through the YOLO algorithm. The output of the YOLO algorithm are bounding boxes around faces in each image. We then crop the images around each bounding box, and feed each of these bounding box thumbnails into our FaceNet model. The FaceNet model then classifies the face in each bounding box.

The first stage of our algorithm has 24 convolutional layers followed by 2 fully connected layers, as described in *You Only Look Once: Unified, Real-Time Object Detection* [3]. The YOLO algorithm divides each input image into an $S \times S$ grid, each with B bounding boxes, and classifies objects into C classes. The model outputs an $S \times S \times (B * 5 + C)$ tensor. Each of the $S \times S$ grid cells have B bounding boxes, and each bounding box is defined by 5 values: the confidence prediction of the bounding box, and the x, y, width, and height values of the bounding box. The confidence prediction reflects the confidence that the box contains an object and how accurately the box bounds the object. Since our version of the YOLO algorithm is only for face detection, we have 0 classes, meaning our model outputs an $S \times S \times (B * 5)$ tensor. We chose the same S and B values as the YOLO paper, where $S = 7$ and $B = 2$. Then, for each bounding box in each grid cell, we have a confidence prediction for the bounding box as well as the x, y, width, and height values of the box. The model removes bounding boxes that have a confidence below a fixed threshold, and also performs non-max suppression to remove overlapping bounding boxes.

Our loss function is based on the loss function found in the YOLO paper, as shown below:

$$\begin{aligned}
&\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \\
&+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2 \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i\right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i\right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \qquad (3)
\end{aligned}$$

Since we are only training our model for binary classification, we do not include the fifth line of the above cost function that accounts for the difference in the probabilities of each class. Our loss function is only the first four lines of the above function. We use the same values of λ_{coord} and λ_{noobj} as they do in the paper, 5 and 0.5.

The second stage of our model is based on a FaceNet model by David Sandberg [6]. Sandberg's model first generates bounding boxes around face candidates. It then classifies the faces in each bounding box. Sandberg's model uses a scaling pyramid to generate bounding boxes around face candidates, which means the model makes several passes through the image looking for faces of different sizes [7]. Compared to this multi-pass pyramid detection algorithm, a YOLO model provides the speed enhancement of real-time object detection. By exchanging the detection stage of the original Sandberg FaceNet model with our YOLO model, we predicted that we would lead to a significant speed-up of the algorithm without sacrificing the accuracy of FaceNet.

As discussed above, the face recognition stage of FaceNet works by passing an image through a convolutional network and generating a 128-dimensional embedding of the image. To train the network, our FaceNet model uses a softmax cross entropy loss combined with center loss, as shown below:

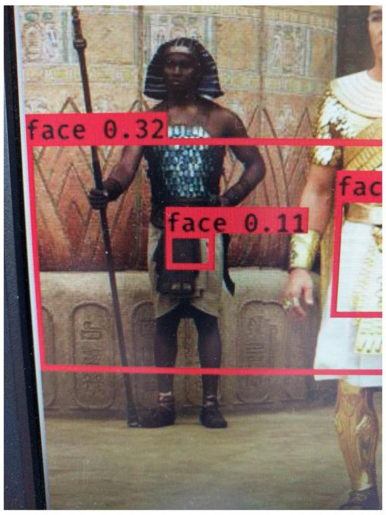
$$\begin{aligned}
\mathcal{L} &= \mathcal{L}_S + \lambda \mathcal{L}_C \\
&= -\sum_{i=1}^{m} \log \frac{e^{W_{y_i}^T \boldsymbol{x}_i + b_{y_i}}}{\sum_{j=1}^{n} e^{W_j^T \boldsymbol{x}_i + b_j}} + \frac{\lambda}{2} \sum_{i=1}^{m} \|\boldsymbol{x}_i - \boldsymbol{c}_{y_i}\|_2^2
\end{aligned}$$

The softmax cross entropy loss penalizes the model for incorrect classifications, forcing the model to learn the differences between deep features of different classes. Meanwhile, the center loss penalizes the model for difference in deep features and their center within a class, forcing the model to minimize intra-class feature difference.

5 Experiments/Results/Discussion

We initially attempted to build a YOLO model from scratch by implementing the architecture described in [4]. This task was challenging, partially due to a lack of specific details in the paper. We determined that it would be more effective to leverage existing YOLO algorithms, of which there are plenty available online. We decided on Darknet-based YAD2K implementation, by Allan Zelener.

We trained the model to produce bounding boxes around faces in images, achieving loss function output of less than 3.4 in terms of training error, which compares well to the to initial training loss of 713 before training. However, owing to lack of computation power, we were unable to train on more than a few thousand samples, which caused overfitting issues on the training set, and led to some humorous outputs in the test set, such as the following:



We did have an accident occur after this - when trying to retrain on a larger dataset, the script provided wrote to the same file where our previous weights were stored, meaning we lost our trained weights that had taken a substantial amount of time to generate. Since we did not have enough time to retrain the entire model, we decided to simulate the output of our YOLO model, and to focus on the speed and accuracy of FaceNet given these inputs.

Ideally, we would have had a file with bounding boxes coordinates for each input image passed into our YOLO network. We would have then cropped each of these images to generate thumbnails of each bounding box to pass into the FaceNet algorithm for training. To simulate this output, we cropped images in the Labeled Faces in the Wild dataset. We split each LFW image into nine even boxes, and generated five 2 box by 2 box semi-random croppings of each image to simulate bounding boxes generated by the YOLO stage of the algorithm. The cropped images were stored as separate thumbnail crops to feed into the FaceNet algorithm.

We split our data so that we had approximately 50% of the cropped images for training and 50% of the cropped images for the dev set. We generated a total of 66,165 cropped images, so in total, we had approximately 33,082 images in our training set and the other 33,082 images in our dev/test set. We used such an even training versus dev/test data division because we were using a pretrained model. The need for training data was lessened by the training that had already occurred and we wanted to ensure we had sufficient data for development and testing.

We trained our FaceNet model on 50 epochs. In addition, we used dropout for regularization with a keep probability of 0.8. We used a learning rate of 0.1, and we used $\lambda = 0.01$ for our loss function. We decided on these values of hyperparameters because the original FaceNet model had used these parameter values and had achieved successful test results. In an attempt to optimize our training, we also used RMSProp.

To test our model, we generated 500 random pairings of the original, uncropped LFW images, with a mix of our cropped images. Approximately half of these pairings were positive pairings and the other half were negative pairings, in keeping with the format used by FaceNet to distinguish faces.

The original Sandberg FaceNet model achieved a test-set accuracy of 98.8% on the 500 pairings of uncropped LFW images in 11.75 minutes, with 10-fold cross validation. Our pre-trained modified model achieved a test-set accuracy of 98.2% on the cropped LFW image pairings, and our trained model achieved a test-set accuracy of 98.7% on the cropped LFW image pairings in 6.5 minutes. Since our model would replace the 5.5 minutes of FaceNet bounding box generation of the 11.75 total minutes with real-time YOLO bounding box generation, we conclude that our model could provide a substantial speed-up with comparable accuracy.

6 Conclusion/Future Work

With more time, we would work on implementing our model as a cohesive unit whole. Right now, we save the output from YOLO to files, and read them in again to FaceNet. It would be much more efficient to create a full network where the last layer of YOLO is connected to the first layer of FaceNet, but in the interest of time, and what we hoped to accomplish on this project, we felt this would be best saved for the future.

7 Contributions

Will: I wrote much of our rudimentary YOLO implementation for the milestone. Once we switched to using pre-trained models, I handled the YOLO end of things, and implemented the croppings method to substitute for the YOLO outputs after the learned weights mishap.
Hannah: I did a lot of research on existing algorithms to help come up with the idea for our algorithm. I also trained our FaceNet model, wrote code to adjust the input to our model, and ran the tests on the model.

8 Code

Github link: `https://github.com/willlauer/CS230-project`

References

[1] Girshick, Ross. "Fast r-cnn." arXiv preprint arXiv:1504.08083 (2015).
[2] Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." Advances in neural information processing systems. 2015.
[3] Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.
[4] Schroff, Florian, Dmitry Kalenichenko, and James Philbin. "Facenet: A unified embedding for face recognition and clustering." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.
[5] Wen, Yandong, et al. "A discriminative feature learning approach for deep face recognition." European Conference on Computer Vision. Springer, Cham, 2016.
[6] https://github.com/davidsandberg/facenet

[7] van Noord, Nanne, and Eric Postma. "Learning scale-variant and scale-invariant features for deep image classification." Pattern Recognition 61 (2017): 583-592.
[8] Tensorflow
[9] https://github.com/allanzelener/YAD2K