Removal of Glasses from User Photos

Chris Stockton
Texas A&M University
College Station, TX USA

Abstract

Many people need to wear glasses but prefer the way they look without them. The goal of our project is to allow users who wear glasses to see themselves in pictures as if they are not wearing them. Current object detection technologies such as YOLOv2 have proven extremely viable for normal object detection. With the relatively recent development of generative adversarial networks (GAN), there has been a vast increase in the ability to manipulate images and create new authentic seeming images.

Our system was designed to implement a combination of state of the art object detection and image manipulation methods in order to locate and seamlessly remove eye glasses from user photos.

1. Introduction

Image manipulation is a task performed with a broad range of uses. With programs like Photoshop, a person could manipulate an image to make a product look better, or even to produce propaganda. In recent years, the use of artificial intelligence has slowly been creeping its way into programs like Photoshop to aid users in editing their images. However, image editing still requires a large amount of hands-on guidance from the user. Even with the current implementation of AI in Photoshop, users must still identify by hand which areas of an image that they want to edit. If a user is looking to edit the same part of many images, they must still comb through each image individually to select what they would like to edit. For people who need to wear glasses to properly see, just about all pictures of their face will contain the same pair of glasses. If they were to want to see their pictures without their glasses on, they would have to individually edit each of their photos. This creates a high demand for a system to automatically detect which images need to be edited, and also where each image needs editing. The rise in artificial intelligence and machine learning has opened a new path to perform such tasks without much human intervention. The technology for automatic object detection already exists today, but has not yet been adapted to face this exact issue.

Perhaps the best example for the state of the art in object detection would be the YOLO framework. The original YOLO method provided a completely new approach to object detection by using only a single neural network to look at an image, dividing the image into small regions and then predicting bounding boxes for each region. There have since been a few updates to this framework. YOLOv2 improves upon the original YOLO. Some of these improvements include classification on higher resolutions, convolutional layers with anchor boxes, and speed. It boasts higher accuracy and speed than YOLO at lower accuracies, as well as achieving an accuracy comparable to Fast R-CNN. At a high resolution it is more accurate than all other previous methods, while running at faster speeds than the R-CNN's and SSD500 [2].

The latest version, YOLOv3, is even faster and more accurate. It is as accurate as and three times faster than SSD at 320 x 320. Unfortunately, for overall mAP, performance dropped significantly [3, 4]. One of the few limitations of the YOLO algorithm is that it struggles with small objects such as a flock of birds [5]. This won't be a problem for us because our project is meant for individuals with glasses.

2 Related Work

Object classification is nothing new, now there are a multitude of ways to solve the problem. One such way is the R-CNN. This method is slow compared to others due to it performing a ConvNet forward pass for each object.The developers produced the Fast R-CNN in response to this by using the SPPnet. This method computes a map for the whole image, which is then used to classify each object [7].

Over the years R-CNN has moved on to the Masl R-CNN which is an extension of the Faster R-CNN by adding a branch which predicts segmentation masks on every Region of Interest [8].

For facial detection, a common technique is the combination of Histogram of Oriented Gradients (HOG) and Support Vector Machine (SVD). This technique's detection ratio varies from mediocre to good depending on the quality of image. Unfortunately, it is not capable of real-time detection on the CPU [1].

This technique computes a pyramidal representation of an inputted image. Then a sliding window approach is used on each entry of the pyramid. For the current window, the HOG is computed and passed to the SVM classifier. Once the pyramid is finished a non-Maxima Suppression (NMS) usually is used [1].

In further research into this topic, I discovered that this exact proposal had already been pursued by 5 senior members of the IEEE (Institute of Electrical and Electronics Engineers) [9]. Instead of using a generalized object detector such as YOLO, this team trained its model to locate a glasses frame pattern on an image after first identifying the eye region where glasses are found.

3. Proposed Technical Plan

3.1. Data Set

We initially planned on using the SoF (Specs on Faces) for this dataset [6]. This dataset provides a collection of 42,592 facial images. The SoF data is created with a range of 112 people, wearing glasses under various light conditions. The dataset was initially designed to challenge facial recognition programs with the issues of low visibility and obstruction from glasses. Because our system is intended to focus solely on glasses identification, we may choose to discard some of the data extreme cases of poor lighting, as large portions of the facial area are not visible.

The main reason for the choice of this dataset above others, is that most other face datasets contain only a small proportion of images containing glasses. While the large amount of images containing glasses in this dataset is very useful, this set of data does not provide any images without glasses, to show the expected output of the glasses removal.

3.2. Object Detection

Our proposed system will only perform image manipulation on images that contain a face wearing glasses. While our data set only contains pictures of faces, we will first identify if a face exists in an image, and then identify a glasses object. This is intended to allow for scalability of the application to data sets that are not just face images.

We will utilize the YOLOv2 object detection system in order to detect both the face and potential glasses object in each image.

3.3. Image Manipulation

After an image is determined to contain glasses, the system will attempt to recreate the image without the glasses object. Due to the lack of glassesless images in our dataset, we are unable to train our system with ground-truth targets. Because of this, we plan on using a GAN framework. A GAN will allow our system to train using unpaired data by continuously generating new versions of the removed area with better and better accuracy.

4. Revised Technical Plan

4.1 New Data Set

In addition to the SoF dataset, we decided to use some images from the Google Open Images Dataset [10]. The Open Images Dataset from Google contained over 40,000 high-resolution jpeg files that contained human faces wearing glasses. This dataset proved to be even more useful than the Specs on Faces dataset, and thus was the primary source of data for this project.

4.4 Switch to YOLOv3

We initially planned to use YOLOv2 in order to perform detection for eyeglasses. However, running YOLOv2 on a windows machine requires TensorFlow [11], an open source end-to end machine learning platform. Unfortunately, YOLOv2 requires certain TensorFlow packages that were discontinued with the release of TensorFlow 2.0. Because of this, we decided to instead use the newer version of YOLO, YOLOv3.

YOLOv3 uses the COCO Dataset[12] by default for object detection. The COCO Dataset has 182 different labels organized into 2 categories, 'things" and "stuff". Of these 182 labels, the most specific label available to us was for "person" so in order to be able to identify glasses we trained the model on 500 images from the Google Open Images Dataset. Figure 1 shows a few sample images from our dataset before detection. Figure 2 shows the same sample images after detection using only the default COCO pretrained weights.

Figure 1: Sample Images from Google Open Images Dataset.(Test Set)

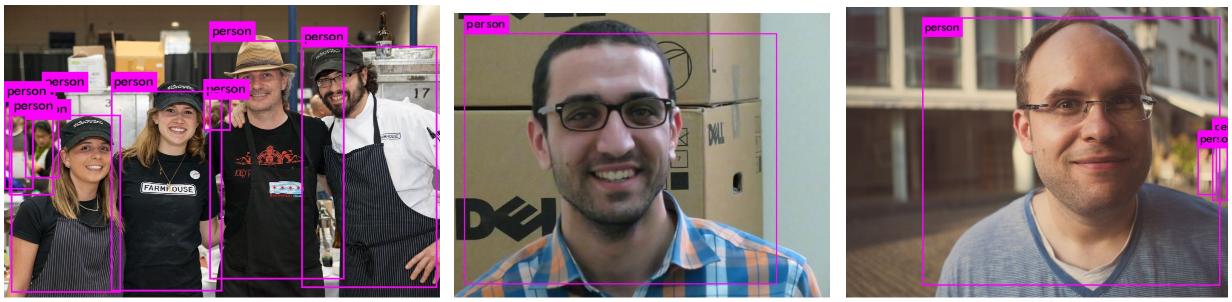

Figure 2: Sample Test Set Images After Standard YOLOv3 Object Detection with COCO Dataset weights

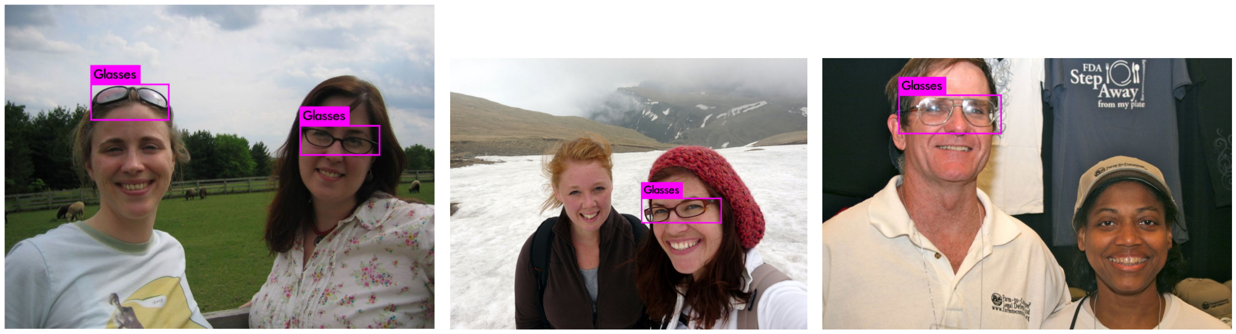

Figure 3: Sample Images from Test Set After YOLOv3 Object Detection with Custom Trained Weights

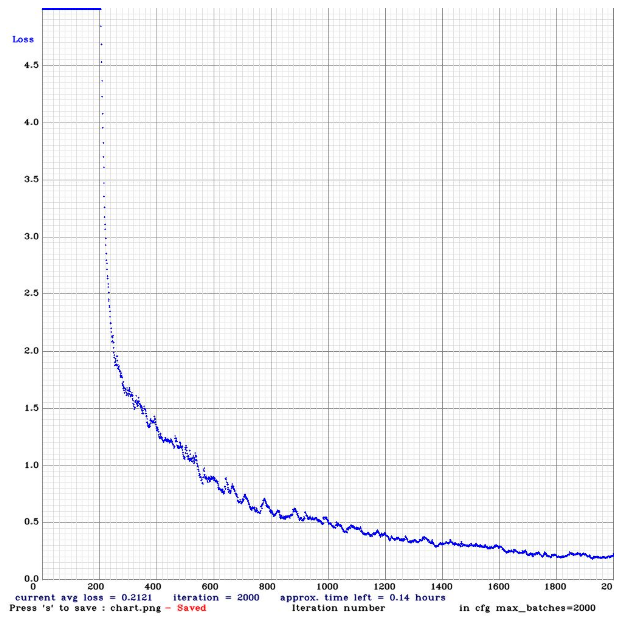

Figure 4: YOLOv3 Training Custom Weights Average Loss

5. Experiment & Results

5.1 Glasses Detection

The implementation of this project started off with many obstacles. Training an object detector is a very intensive process for a computer. With 99% of my computer's CPU being dedicated to training the model, it was predicting an estimated 340 hours to complete the training. Thinking this was perhaps inaccurate, which is not uncommon with time-until-completion estimates, I let the program run overnight to find in the morning that the time until completion had only decreased to 333 hours. At this point only 62 of the planned 2000 iterations had passed and the average loss was still in the 2000s. From what I could find on the internet, model training all but needed to be done with the help of a GPU. While my computer does have a GPU, it is an NVIDIA Quadro M1200, a model that is incompatible with CUDA, and was not able to be used.

In order to make use of a GPU to train the model, I uploaded the YOLO and Darknet code to Google Colab where I could make use of a virtual GPU.

Using Colab's virtual CPU, I was able to train the YOLOv3 detector by doing 2000 iterations on 500 images. The average loss started in the 2000's but was able to drop below 1.0 around 550 iterations. The average loss started to converge at about 0.2-0.25 after 1600 iterations. Figure 4 shows the model's average loss function over time.

After training the new custom weights, I was successfully able to identify glasses on the images of our testing set, as seen in figure 3.

5.2 Glasses Removal

In the case of removing glasses from a person's face, the most important thing is maintaining the overall integrity of the users face. A photo without glasses is useless to the user if the face inside no longer resembles the user.

Unfortunately, I was unable to implement a process that could properly edit the glasses out of an image while still maintaining the integrity of the user's face. While Generative Adversarial Networks are very capable of creating new images that look like real faces, this application requires the generation of a new image that not only looks like a real face, but an exact face. This would require the GAN to be trained on a set of images of the user without glasses to then be able to generate new instances of that exact user's face. While it is not unrealistic to require this data in a real world sense of the application, I unfortunately did not have access to this type of data.

6. Expansion

6.1 Possible Solution

While I was unable to create a solution for proper image manipulation, I have a few ideas as to how to get there. The main issue in generating a new image was maintaining overall image integrity. A possible way to overcome this issue would be to not try and create an entirely new image and instead just extract the glasses area of the image. We could do this by using the coordinates generated for the bounding box created during the detection portion of the application. Then, we could use the GAN to create a new version of just the glasses area, without ever interacting with the rest of the face. Even with manipulating this smaller portion of the image, there would still be the issue of maintaining the integrity of the user's face within the glasses area of the image. Features like eye shape, eye color, nose shape, and skin tone would have to be maintained in order for the application to be usable.

References

[1] Le, James. "Snapchat's Filters: How Computer Vision Recognizes Your Face." Medium, Cracking The Data Science Interview, 22 July 2018, medium.com/cracking-the-data-science-interview/snapchats-filters-how-computer-vision-recognizes-your-face-9907d6904b91.

[2] Aidouni, Manal El. "Understanding YOLO and YOLOv2." Manal El Aidouni, 25 June 2019, manalelaidouni.github.io/manalelaidouni.github.io/Understanding%20YOLO%20and%20YOLOv2.html.

[3] Joseph Redmon and Ali Farhadi, YOLOv3: An incremental Improvement, arXiv: 1804.02767, 2018.

[4] Aidouni, Manal El. "Understanding YOLO and YOLOv2." Manal El Aidouni, 25 June 2019, manalelaidouni.github.io/manalelaidouni.github.io/Understanding%20YOLO%20and%20YOLOv2.html.

[5] sharma, Mukund. "Everything You Need to Know to Train Your Custom Object Detector Model Using YOLOv3." Medium, Analytics Vidhya, 17 Oct. 2019, medium.com/analytics-vidhya/everything-you-need-to-know-to-train-your-custom-object-detector-model-using-yolov3-1bf0640b0905.

[6] Afifi, Mahmoud, and Abdelrahman Abdelhamed. "SoF Dataset." *SoF Dataset*, 2019, sites.google.com/view/sof-dataset.

[7] R. Girshick, "Fast R-CNN," *2015 IEEE International Conference on Computer Vision (ICCV)*, Santiago, 2015, pp. 1440-1448.

[8] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick, Mask R-CNN, arXiv: 1703.06870, 2017

[9] Chenyu Wu, Ce Liu, Heung-Yueng Shum, Ying-Qing Xu, Zhenyou Zhang."Automatic Eyeglasses Removal from Face Images" *IEEE Transactions on Pattern and Machine Intelligence*, March 2004. https://people.csail.mit.edu/celiu/pdfs/eyeglasses-TPAMI.pdf

[10] Google Open Source. Open Images Dataset v6 https://opensource.google/projects/open-images-dataset

[11] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[12] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, Piotr Dollár. Microsoft COCO: Common Objects in Context. http://cocodataset.org/#home