CSCI 1430 Final Project Report: Face Anonymizer

Shallow Faces: Andrés Beck-Ruiz, Claire Oberg, Jed Fox.
Brown University

Abstract

We created a tool for anonymizing faces in images by replacing them with faces selected from a set of AI-generated faces (from the StyleGAN 2 paper). The results were often deeply unpleasant.

1. Introduction

We wanted to be able to effectively anonymize images of people – either alone or in groups. Currently, journalists and apps like Signal provide some level of anonymization, but it comes at a great cost: current state-of-the-art approaches simply replace all or part of the face with a black rectangle or oval, or blur it beyond recognizability. These solutions all remove the human connection from the image — you can see body language, but all the expressiveness of the face is lost.

Our proposed system attempts to retain much of the detailed expressiveness of the human face, by replacing the subject's identifiable face with one from a dataset of AI-generated ones. We believe that this will preserve facial posture while effectively hiding details like the exact appearance of facial features that make someone uniquely identifiable.

2. Related Work

Our AI-generated faces were extracted from Stwyle-GAN 2 [3], with no curation on our end besides selecting an arbitrary subset of the images due to the amount of training time required. Additionally, we referenced online tutorials [4][1] to aid in developing features, and used a pretrained model from dlib [2] to detect facial features.

3. Method

We built several interlinked components:

- `cnn_code` uses a neural network to guess the gender of a face in an image.
- `faceswap` replaces a face at a provided location in the image with another provided face.
- `frontend` is a React-based UI that allows uploading images and processing a camera feed live.
- `server` is a Flask-based server that combines `cnn_code`, `faceswap`, and `teeth` to implement the CV-based functionality of the frontend.
- `teeth` analyzes faces using the 68-face-landmarks model, and detects whether the face's teeth are visible or not.

For the CNN model, the architecture used is as follows:

- Two convolutional layers with 64 filters, a 3x3 kernel size, a stride of 1, and a ReLU activation
- A max pooling layer with a 2x2 pool size
- Two convolutional layers with 128 filters, a 3x3 kernel size, a stride of 1, and a ReLU activation
- A max pooling layer with a 2x2 pool size
- Two convolutional layers with 256 filters, a 3x3 kernel size, a stride of 1, and a ReLU activation
- A max pooling layer with a 2x2 pool size
- Two convolutional layers with 512 filters, a 3x3 kernel size, a stride of 1, and a ReLU activation
- A max pooling layer with a 2x2 pool size
- Two convolutional layers with 512 filters, a 3x3 kernel size, a stride of 1, and a ReLU activation
- A max pooling layer with a 2x2 pool size
- A dropout layer with a rate of 0.3
- A flatten layer
- A dense layer with 128 neurons and a ReLU activation function
- A dense layer with 2 neurons and a softmax activation function

We edited and used the code from homework 5 to train our model on a dataset for gender classification. To check the accuracy and loss of our model, we viewed the results of training on Tensorboard. Finally, to classify the gender of each image in our randomly generated faces dataset, we developed a short script to predict each image using the model and store the results in a JSON file.

In order to predict the gender of single images, we followed a tutorial (included in the bibliography) for gender classification using pre-trained weights. We built a function to run this algorithm every time a user selects a new image on the frontend. Both this function and the script to predict images in the randomly generated faces dataset are included in `predict.py`.

Teeth detection was carried out by computing the area of a triangle composed of three feature points around the mouth. A larger triangle area indicates that the person is smiling or has their mouth open. Initially, we used a threshold area to differentiate between "smiling" and "not smiling" states (this is what the green/red rectangle in the web version indicates). However, we decided to switch to looking for faces with similar triangle-area scores, ensuring that an incorrect estimation of the area threshold wouldn't impair the effectiveness of the algorithm.

We pre-calculated the tooth triangle-area score for all of our candidate replacement faces and cached it in the same JSON file as the result of the gender classification model.

When finding a face to anonymize an uploaded image on the frontend, we predicted the gender of the face and computed the tooth triangle area score. Then, we searched for a face in our randomly generated faces dataset that had the same gender and a similar tooth score and swapped it with the face in the uploaded image.

In order to swap faces we followed a tutorial [1] that explained key steps for swapping faces. We first used python's dlib face detector to detect the face and then used a model to identify the 68 key feature points of the face. We then calculated the Delauney triangulation between the points which gave us a number of triangles between each point. The triangles were then stretched to fit the corresponding triangle on the other face and each triangle was replaced. This resulted in the new (ai generated) anonymous face on the existing head.

4. Results

Look at all the fun images!

4.1. Technical Discussion

There were a few technical decisions made regarding the face swapping aspect of the code. The first was the decision to use python dlib's get frontal face detector. This detector uses HOG + Linear SVM to detect the features of the face. The alternative was to use a CNN to identify the faces which

Figure 1. Jennifer Lawrence

Figure 2. Jennifer Aniston

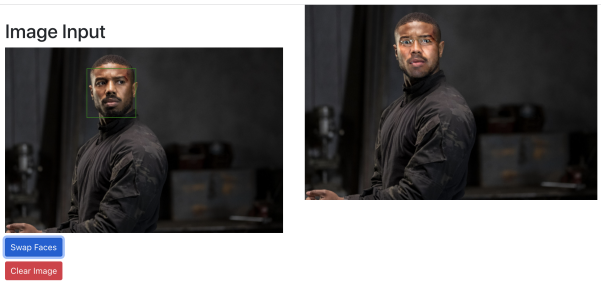

Figure 3. Michael B. Jordan

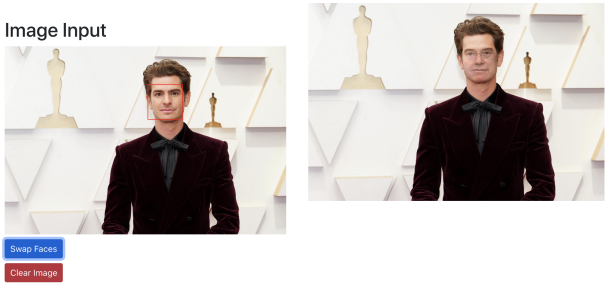

Figure 4. Andrew Garfield

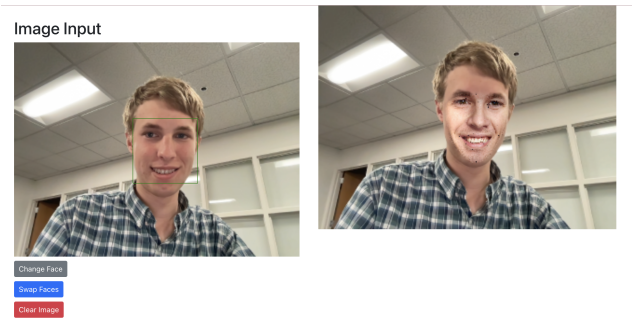

Figure 5. Screen capture from the webcam mode

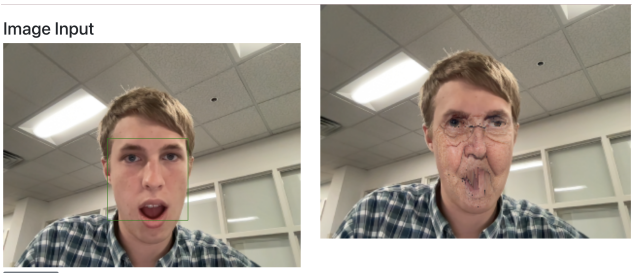

Figure 6. Example of a bad result: I selected the face while my mouth was closed, then opened my mouth.

Figure 7. Loss over time while training our CNN model for gender classification

would do a better job of identifying faces at angles or in different lightings, however, it is significantly slower than the dlib detector.

Another decision made was to use a preexisting shape predictor to identify the 68 landmarks on the face; the alternative options considered were to train our own model or use a model with a different number of landmarks. This model worked incredibly well, so we decided to move forward with using this one in the implementation.

Another decision made was the choice to use cv2's normal seamless clone as opposed to the mixed clone option. The mixed clone option blends the skin tone and facial features more than the normal clone which is helpful in making the

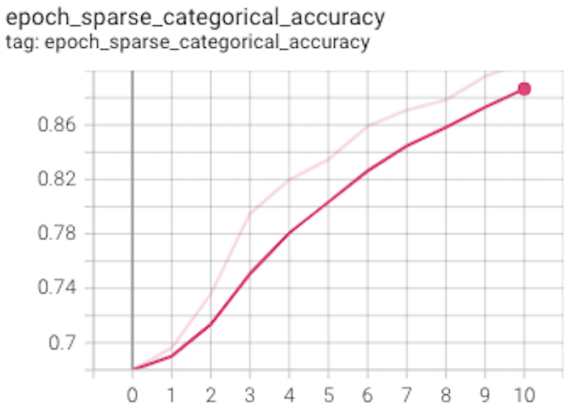

Figure 8. Accuracy over time while training our CNN model for gender classification. Our model achieved a high of 94.5% testing accuracy during the 10 training epochs.

face look more realistic. However we ultimately decided that the faces created using mixed clone were too similar to the original face, so normal clone was the best option.

In terms of the CNN model for predicting gender, we were unable to determine why the model was inaccurate for predicting single images. While the model accurately predicted gender for the training dataset, as shown in the result section, and for the randomly generated faces dataset, it consistently predicted images as being female regardless of which image was uploaded. We attempted to debug this issue by using the same standardization method for predicting single images that we used for predicting the entire dataset by computing the mean and standard deviation of the training dataset. We also attempted to classify single images from the randomly generated faces dataset, which included only the face in the image, to rule out the possibility that the model was making predictions based on other parts of the image. However, we were still unable to figure out why our model was predicting gender incorrectly for single images even after taking these steps. Therefore, we used an online tutorial and pre-trained weights for predicting gender in order to accurately classify uploaded images on the frontend.

It is also important to note that for our project, we used gender classification based on only two genders (male and female). We recognize that gender classification is problematic, as many people do not identify as the gender someone might classify them as or identify as non-binary. Predicting gender allowed us to choose a better face to anonymize uploaded images with. With more time, we would choose the best image based on matching individual facial features.

4.2. Societal Discussion

1. **Critique:** *In response to the project's main goal of aiding protesters, this could actually undermine the credibility of protesters and public influencers who apply*

this software; wearing someone else's face (or synthetic face) takes away the liability one has when saying something wrong, which could undermine their credibility. People might connect less with someone knowing that the face isn't real, even if the alternative is not having a face to show.

Response: While these are valid concerns, we believe that the benefit of allowing protestors to anonymize themselves to avoid backlash outweighs the possibility of an audience connecting less with the protestor or undermining their credibility. Additionally, it's already possible to record a video where your face is out of frame (like CGP Grey does) or record an audio track over a free stock video, and both of these methods allow for similar anonymity to our method.

2. **Critique:** *This project also contributes to development of deep-fake technology and its implication: spreading misinformation and misrepresentation, which could bring harm to people's reputation and could incite conflicts (for example, when political statements are faked by public figure lookalikes exploiting Shallow Faces' algorithm).*

   Response: We felt that despite the existence of tools which perform much more seamless face swapping than we were hoping to achieve, very little misinformation actually uses such techniques. Since people aren't heavily using these existing tools for misinformation, we doubt that our method would be used in this way. For example, despite many, many photographs of Joe Biden being publicly available as training data, the closest we've seen in terms of viral manipulated media of him was a video where he was simply slowed down — a technique that's been available to just about anyone for years.

3. **Critique:** *Furthermore, it is important to have a diverse set of faces from across races to achieve the team's goal of accurate representation. Even humans suffer from the cross-race effect as a result of not being exposed adequately to people of other races. With how much more rigid the training for AIs can be, it is very important to have a diverse dataset in order to prevent the tool from working better on certain kinds of faces.*

   Response: We have ensured that our dataset used to train our CNN model for predicting gender contains a diverse set of faces in terms of age and race. Furthermore, our dataset of AI generated faces contains a variety of ages, genders, and races, given that it is randomly generated. While there are probably axes of diversity that we have not considered, adding additional diversity to the set of replacement images (or the training set for models we're using) is the sort of thing that will be relatively easy to change in the future as more concrete concerns are raised.

4. **Critique:** *One of the team member's roles has a listed task of "gender estimation". This is a rather problematic way to go about this task, as estimating gender is a loaded task. Even humans evaluating a person face-to-face can't always accurately identify someone's gender, and there are plenty of people who don't identify with a given gender and/or have androgynous looks. It might be better to try and match the faces purely based on how they look, instead of by sorting them into gender categories in order to find matching faces.*

   Response: This is a valid concern; however, we are estimating gender as a starting point for narrowing down the list of candidate faces we can use for our anonymization algorithm, and we do not publicly display the results of this estimation. We recognize that there are many gender identities and expressions and that not everyone might identify with the (binary) gender we classify them as. With more time we would like to replace our gender estimation algorithm with a carefully-considered set of facial characteristics that would make users comfortable with the artifical face they are given without revealing enough information to identify them.

5. **Critique:** *Applying technology to protesters could even endanger those whose face resemble the replacement faces. For one, replacing mass protester faces might protect the protesters at the expense of lookalikes. For another, if a social media influencer discusses politically charged topics on platform in countries with less freedom of speech, and the protester chooses not to declare that they are using face replacement technology, they might put lookalikes in danger.*

   Response: In order to protect any lookalikes of the randomly generated faces, we could include a watermark in any edited images. This would declare to anyone viewing the image that the faces in the images are not real people.

5. Conclusion

We implemented an algorithm to anonymize the faces in a photo by replacing the original face with an ai-generated face. We also created a website to interact with this tool. On this website photos can be uploaded and anonymized and there is also an option to anonymize a live video feed. This could be used to protect the identity of protestors while still showing a face, allowing viewers to better identify with the protestors.

References

[1] Sergio CanuHi. Face swapping (explained in 8 steps) - opencv with python, Jun 2019. Available at https://pysource.com/2019/05/28/face-swapping-explained-in-8-steps-opencv-with-python/. 1, 2, 5

[2] GuoQuanhao. shape predictor 68 face landmarks.dat, Nov 2019. Available at https://github.com/GuoQuanhao/68_points/blob/5062450a98a958501ff307703f5721f8fb11692d/shape_predictor_68_face_landmarks.dat. 1

[3] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. In *Proc. CVPR*, 2020. 1

[4] Sefik Serengil. Age and gender prediction with deep learning in opencv, May 2021. Available at https://sefiks.com/2020/09/07/age-and-gender-prediction-with-deep-learning-in-opencv/. 1

Appendix

Team contributions

Andrés I worked on training a CNN model, based on the Homework 5 code, to classify the gender of a person's face included in an image. I then wrote a script to run this model on our database of randomly generated faces to classify each image as male or female. Finally, I followed a tutorial online to classify single images as male or female, as my model was not predicting the gender of single images correctly. I also did some work implementing the front end.

Claire I worked on the code to swap faces using python dlib and open cv by following a tutorial from pysource [1]. This code uses a preexisting model to detect the feature points of the face, divide the face into triangles, and replace the triangles of the second face with the triangles of the first.

Jed I built the algorithm that attempts to detect whether the subject is smiling. I also did most of the work for setting up the frontend and the backend server, including piping data over HTTP and integrating with the Web camera capture API.