

Face swapping and harmonization using neural nets

Akshay Kumar
Stanford University
akshayk@stanford.edu

Abstract

The advent of online social media networks such as facebook, instagram or pinterest has ignited the question of privacy concerns on online social networking sites. Accused of affecting US presidential election results, facebook has also been sued by privacy advocates for face recognition because it identifies a user in an image without their consent. Motivated by this, in this paper, we will try to study if it is possible to fool a face recognition software by means of exchanging the faces.

Towards this goal, we will develop a three step approach meant to exchange the face of a user in an image by the face of another (base or anonymous) user such that the face detector can not recognize the original face. The three steps of our algorithm are : face detection, face swapping and face blending. Finally, we will evaluate our results by comparing the output images from our algorithm against target image to check if they belong to the same person or not.

1. Introduction

The advent of online social media networks like facebook, pinterest, etc. has brought the problem of face detection to a fore. DeepFace [21], the state of the art algorithm used by Facebook for face detection, achieves an accuracy of 97.35% on the LFW (Labeled Faces in the Wild) dataset (better than human performance). Apple launched FaceID as an alternative to password to securely unlock your iphone/macbook.

However, all these technologies beg the question of online privacy of your facial photos on internet. The question is multi fold in the sense that multiple online privacy groups have filed lawsuit against, for e.g., facebook and there have been work around spoofing face detection techniques used to unlock laptops [20].

Motivated by this, in this project, we will approach the problem of anonymization of faces in a image. Given a face we want to replace it by a different face which can not be easily recognized by a face recognition algorithm. We will follow a three step approach: (1) detect face in an image; (2)

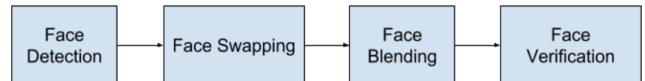


Figure 1. Overall framework of our algorithm

swap it with a different face; (3) blend the new face into the image using techniques like harmonization [11]. Finally, we have an additional step related to Face Verification for evaluating our method.

2. Related Work

Face Anonymization is a well studied problem in computer vision. The current literature is based on multiple techniques like similarity based metric [13], adversarial training [16] or very simple techniques like gaussian blurring of the recognized face [18]. However, a overall feature of all these approaches is that the underlying face is still the face.

In this project, we take a departure from this approach and try to anonymize the face as a three step process by first recognizing face in an image, swapping it and the blending the new face into the original image.

Face detection. Face detection has been an active field of research and a seminal paper in this field is Viola-Jones object detection framework for face detection [23]. Viola-Jones uses Haar feature to form matchable facial features and builds integral image so that rectangular features can be detected in constant time. It then uses a variant of Adaboost to create strong classifiers and select the best features. Finally, they get a cascade classifier by combining selected features into a cascade architecture.

Recently, researchers have started experimenting with deep learning architectures for face detection. We will now discuss about them. Zhang et al. [24] used a three stage fully convolutional network: (1) Proposal Network (P-Net) to obtain possible bounding box windows. (2) These bounding box are fed to Refine Network (R-Net) to eliminate multiple possible candidate boxes either by merging highly overlapping boxes or using bounding box regression (We will do some similar in our project as well. More details in

next section.). detection. (3) Finally, they look at the facial landmarks in remaining candidate boxes to correctly determine the correct face box [Figure 2]. There has also been some past work around using SVMs [15] for face detection. Edgar et al. [15] trained an SVM classifier for faces in an image and achieved a detection rate of 74.2% on their dataset.

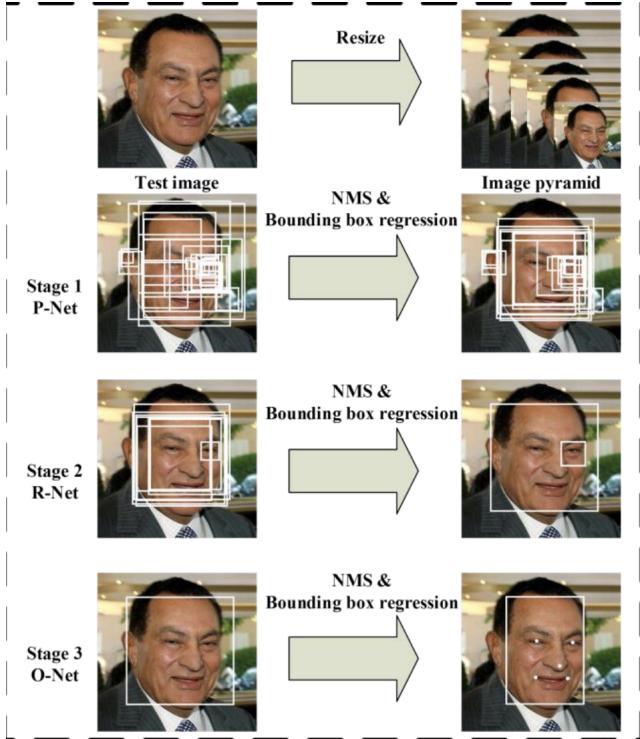


Figure 2. CNNs for face detection [24]

The major problem with face detection was the size of search space : the sliding window for face detection has to look at all possible boxes. R-CNN overcomes this by looking at Region of interests which are regions with high likelihood of a face. Faster implementations of R-CNN, Fast R-CNN and faster R-CNN speed up the process by only first passing the image through a conv image map and then apply Regions of Interest. In this project, motivated by [3], we will adopt a CNN based approach to detect face in an image. Note that face detection is different from face recognition and in this part, we are mainly concerned with face detection. We will briefly touch upon face recognition in the end for evaluation strategies.

Face Swapping. There has been some work around face swapping [10, 5, 14]. They rely on a pattern of face alignment by replacing facial features of the first image by the facial features of the second image by detecting facial landmarks in the first image, fitting the second image into the first by rotating and scaling the face in second and harmonizing it.

However, an essential problem with this approach is it is not receptive to face orientation and this can easily distort the new face orientation. We used the approach of [7] to swap faces of Donald Trump and Barack Obama. The output image, as seen in Figure 2, is highly distorted by original face orientation and the texture is still unaffected.



Figure 3. Face Swap : Superimposing Obama's face on Trump's [7]

Face Blending. Image blending is a well known problems and uses techniques like Active Shape Models to determine distinctive textures and features like face and hands and uses them to transpose a new face rotated into the original image [6]. However, in context of face blending, they suffer from the same drawback that it is easy to trace back the original image.

Another possible approach is to use style transfer techniques using deep neural nets to blend a new face into a content image [8]. In this approach, we have a target image (the images composed of face swapped) and two input images : the original image (the old face) and the swapped image (the new face). We adopt a deep dream approach ?? and compare similarity based on euclidean distance to original image's face and style (using gram matrix) to target image. However, this leads to deep dream kind of images which does not work for face blending.

We will use Poisson blending to blend the face into the target image (described in detail in next section).

2.1. Face Verification

Face Verification falls into the same realm of face recognition and any state of the art algorithm like Facebook's Facenet [19] or Deepface [21] can be used to verify that two faces are of the same person. They achieve accuracy of more than 97%.

3. Method

As discussed previously, we break down our approach into three components: face detection, face swapping and

face blending. We use a CNN based model along with sliding window for face detection. For face swapping, we simply swap the faces of the source and destination image. For face blending, we rely on poison blending to blend a face into its surrounding image. The three parts are discussed more in detail here:

3.1. Face Detection

Our face detection algorithm is similar to [1]. The advantage with concentrating just on face detection rather than the whole problem is that here, we only have one face to detect. The current state of art mechanism for CNNs based face detection is more complex as it detects all the faces in an image. In our case, the task at hand is simpler.

Our overall approach is to scan through the image and check parts of it (called a bounding box or a mask) for a face. We use a CNN based model for learning a bounding box around the human face in the image. The bounding box is composed of two things: coordinate and size. The model architecture is as follows (Figure 3.1):

- 4 5 X 5 Convolution Layer
- ReLU layer
- 16 3 X 3 Convolution Layer
- ReLU layer
- MAX POOL layer
- 32 3 X 3 Convolution Layer
- ReLU layer
- FC layer
- ReLU layer
- Softmax

We first resize the input image to 300 X 300 size. To detect the location of face in an image, we follow a sliding window approach as described in [22]. A faster approach is using Regions of Interest (RoI) [4] approach but we ensured quick computation speed by using higher strides.

The sliding window is of size 40×40 , 50×50 , ..., 100×100 . We use a stride of 20 during training. Each candidate box is then reshaped to size 32 and used to train the network.

The network gives a score for each of the sliding window, *i.e.*, high score sliding window are highly likely to contain a face. For generating the final output box, the commonly used technique is Non maximum suppression (NMS) [17] : In NMS, we set a threshold on the output score and discard sliding windows below this output score. It then merges the boxes which intersect with each other. Finally, it predicts the box with maximum output.

However, in our case, the image has only 1 single face. Hence, NMS is not needed. Instead, we simply pick the mask with highest score as that as good enough approximation of the correct face. Refer Figure 3.1 for a sample output of our face detection framework.

3.2. Face Swapping

This part is simple : we simply swap the faces found in two images after appropriate scaling.

Since the faces detected in source image and destination image can be of different sizes, we scale the face in source image to be of the same size as that of destination image.

Refer Figure 3.2 for a sample output.

3.3. Face blending

Once we have the source face swapped into the destination image, we will use Poisson Image Editing to merge the source face into the destination image. Poisson Image Editing by first described by Patrick et. al. in [17].

Poisson Image Editing is based on the fact that it's very easy for us to recognize abrupt changes in an image (for e.g., a different face with a different image in our case). This can be mathematically termed as an abrupt change in gradient. Poisson Image Editing tries to preserve the gradient such that colors do not change abruptly.

We will now pose this problem mathematically as described in [2].

As shown in Figure 3.3,

- g : Selected region of source (Source face in our case)
- S : Destination image
- \mathbf{v} : Gradient of region in g
- Ω : Region g that is now replaced on domain S (source background)
- $\partial\Omega$: Boundaries between the source and destination images.
- f^* : Known functions that exist in domain S (destination image in our case)
- f : Unknown functions that exist in domain Ω

Governed by equation described in [2], we will try to minimize the change in gradient upon moving from source face to the destination image:

$$\min_f \int \int_{\Omega} |\nabla - \mathbf{v}|^2 \text{ with } f|_{\partial\Omega} = f^*|_{\partial\Omega}$$

whose solution is the solution of the Poisson's equation

$$\Delta f = \operatorname{div} \mathbf{v} \text{ over } \Omega \text{ with } f|_{\partial\Omega} = f^*|_{\partial\Omega}$$

Here, $\operatorname{div} \mathbf{v} = \frac{\partial v}{\partial x} + \frac{\partial v}{\partial y}$. Δ is the laplacian operator

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}.$$

The laplacian operator to the Poisson equation will give us

$$\operatorname{div} G = -4f(x, y) + f(x-1, y) + f(x, y-1) + f(x+1, y) + f(x, y+1)$$

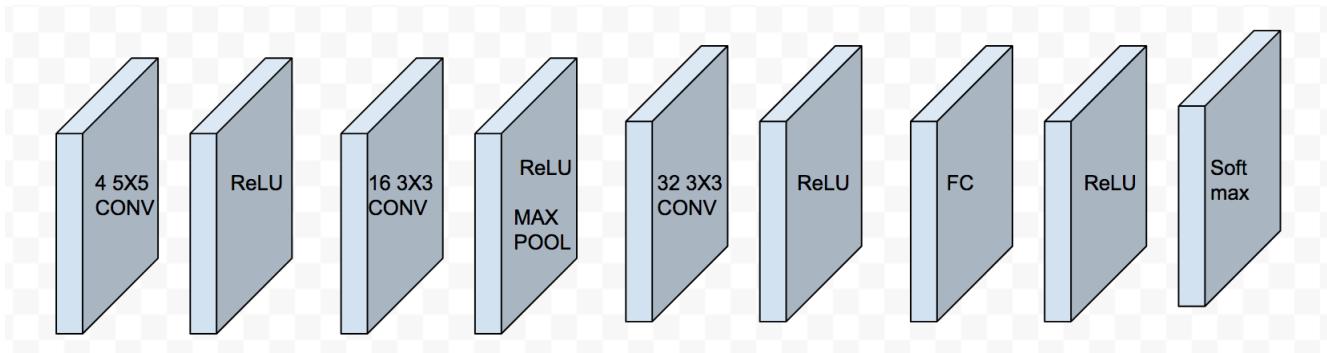


Figure 4. Neural Network architecture for face detection

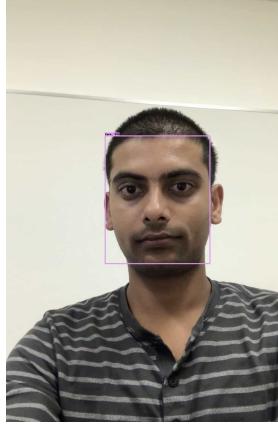


Figure 5. Face detection using approach described in Section 3.1



Figure 6. Swapping out of two faces in an images. Trump's face interposed on top of Obama's

3.4. Face Verification

We use the approach of Zhang et. al. [24] as described in [12] to verify if two faces are similar or not.

Towards this end, we use python's facenet library to compare the similarity of two faces. The model gives a 128 bit vector as output to an input image, which we call as embeddings. We compute the L2 norm between two embeddings.

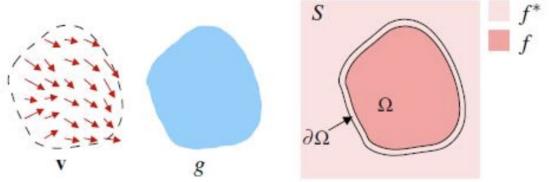


Figure 7. Poisson Image Blending Overview [2]

$$d(u, v) = \sqrt{\sum_i (u_i - v_i)^2}$$

We decide if two images are of the same person based on a threshold for embedding similarity, i.e., the two images belong to same person if embedding distance is less than, say, 1.

4. Dataset

The first part of our approach, face detection, requires image dataset to train the CNN based model for face detection. Towards this end, we use annotated LFW (Labelled Images in the Wild) dataset to train the classifier [9].

LFW dataset contains 13k images of faces collected from web. In addition, each of the image has been labelled with the name of the person. Please note that LFW dataset is also augmented by the name of person however we discard that data for our purposes. The faces on original LFW dataset were detected by Viola-Jones algorithm. Refer Figure 4 for sample images from the LFW dataset.

We further augment the dataset by flipping the images horizontally. The training:validation:testing split used is 60:20:20.

5. Results

During the first phase of our algorithm (face detection), we started with a model pretrained on LFW [1]. This model had a validation accuracy of more than 98%. We used



Figure 8. Sample images from LFW dataset

OpenCV for the final part of our project. In particular, for Poisson Image Blending, we used seamless cloning of opencv.

We will now present both the qualitative and quantitative results of our algorithm.

5.1. Qualitative Results

As a crude test of our approach, we will first present the results by swapping the faces of the same person.



Figure 9. Algorithm's output of two similar photos of Obama

Figure 5.1 shows the output of the algorithm for the image of Barack Obama (both source and destination image are the same). The first row corresponds to the source image and the last row corresponds to the destination image.

The various images, in order are (left to right, top to bottom):

1. Original source image
2. Source image with face detected as blacked out
3. Cropped out face from the source image
4. Destination image
5. Destination image with target face surrounded by a box
6. Final image : Destination image with source image face blended into destination face.

We will now present the results for a few more pair of images. Results in Figure 5.1, 5.1, 5.1 and 5.1.

These examples have been cherrypicked since they uniquely demonstrate some of the good swappings and some bad swappings.



Figure 10. Face swapping for Sachin Tendulkar and Virat Kohli



Figure 11. Face swapping for David Beckham and Christiano Ronaldo

- In Figure 5.1, the image blending has not been so



Figure 12. Face swapping for Niki Lauda and James Hunt



Figure 13. Face swapping for Tom Cruise and Daniel Craigman

smooth and parts of eyes of both source and destination image is visible in target image.

- In Figure 5.1 and 5.1, the swapping is better and parts of faces (especially eyes) have been replaced by the source face.
- In Figure 5.1, the image blending didn't happen correctly. It can be observed that the eyes in destination image got replaced by hairs of source image.

As evident from these images, the algorithm used by us is highly susceptible to the original orientation of the two images. Also, things like relative position of facial features like eyes play a major role in how the output image looks like.

5.2. Quantitative Results

The original problem statement was to anonymize the face in an image. Since there's no notion of accuracy of our algorithm (there's nothing to compare the output against), we will define a confusion matrix kind of notion to evaluate the quantitative performance.

Using face verification framework, we look at the similarity of the output image from both a random person corresponding to both source image and a random person corresponding to destination image.

The results have been summarized in Table 5.2. While evaluating the performance, to be on the safe side, we compared the similarity of output image with original destination image, not a another image corresponding to the same person as that in destination image. As a result, the error reported here is amplified.

We will now describe the results obtained here. For (1) and (4), in output image, it can be clearly seen that the source face is on top of destination face in output image. Hence, the face verifier correctly predicted that the face in output image is similar to source image but not to destination image. In (3), the face swapping wasn't perfect and the eyes of destination image are supposedly replaced by hairs of the source image. Hence, the face verifier predicts that the output image is similar to source image but not to destination. In (2), the eyes haven't been completely swapped out and eyes from both source and destination image can be seen in output image. Hence, the face verifier says that output image is similar to both the images.

Note that we mainly care about the error between destination image and output image since our main focus is to anonymize the destination image (but it's a good exercise nonetheless to look at error from source image as well. We would ideally want output image's face to match source image's).

Overall, we ran this algorithm on 25 pairs of (source, destination) image and compared the similarity of ouput image to them. In 17 cases, output image was not similar to the destination image. In 15 cases, output image was similar to the source image.

The overall confusion like matrix is shown in Table 5.2.

6. Conclusion

We tackled the problem of face anonymization in online social media settings using a three step approach of CNNs based face detection, face swapping and face blending. The results we obtain were able to replace the face in the target face but with some caveats:

- Our model is highly susceptible to texture difference in the two faces.
- It can't take into account orientation aspects. For e.g., if the eye is aligned horizontal in source image and vertically in destination image, our algorithm will try to align the eye horizontally in destination image.
- As shown in one of examples, sometimes the faceswap can not eb so accurate and we may end up swapping different features of the face.
- We try to blend the source face into destination image by blurring the edges. Even if this is successful, the color/texture of the face/skin can be vastly different. As such, it is easy to recognize that the output image's face is composed of two different faces.

	Source Image	Destination Image	Output Image	Similar to Source Image	Similar to Destination Image
1				YES	NO
2				YES	YES
3				NO	YES
4				YES	NO

Figure 14. Confusion type matrix for the performance of the algorithm described above

	Output image similar to destination image	Output image not similar to destination image
Output image similar to source image	6	9
Output image not similar to source image	2	8

Figure 15. Overall confusion like matrix

One promising approach to now try is to compute start off with the target image to be the same as source image and then modify by be minimizing its face distance from destination image.

7. Contributions & Acknowledgements

We would like to thank CS231N course staff for helping us with all the stuff. Our original team consisted of three team members but one of them dropped the course

after project proposal and the other dropped the course after project milestone. So the work on final project (after project milestone) was done by only one person. Before that, the contribution was equal (equally among three for project proposal and equally among two for project milestone).

We used the codes from following github repo for our project for getting a pretrained model for face detect: FaceDetect

References

- [1] Facedetect. <https://github.com/PCJohn/FaceDetect>. Accessed: 2018-05-15.
- [2] Poisson blending. <http://eric-yuan.me/poisson-blending/>. Accessed: 2013-10-15.
- [3] Cnns for face detection and recognition. pages 1–7, 2017.
- [4] B. Alexe, T. Deselaers, and V. Ferrari. Measuring the objectness of image windows. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2189–2202, 2012.
- [5] D. Bitouk, N. Kumar, S. Dhillon, P. Belhumeur, and S. K. Nayar. Face swapping: automatically replacing faces in photographs. In *ACM Transactions on Graphics (TOG)*, volume 27, page 39. ACM, 2008.
- [6] A. Chou and Y. Hong. Face detection, extraction, and swapping on mobile devices.
- [7] M. Earl. *Switching Eds: Face swapping with Python, dlib, and OpenCV*, 2015 (accessed May 15, 2017).
- [8] L. A. Gatys, A. S. Ecker, and M. Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.
- [9] V. Jain and E. Learned-Miller. Fddb: A benchmark for face detection in unconstrained settings. Technical Report UM-CS-2010-009, University of Massachusetts, Amherst, 2010.
- [10] V. Kazemi and S. Josephine. One millisecond face alignment with an ensemble of regression trees. In *27th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, United States, 23 June 2014 through 28 June 2014*, pages 1867–1874. IEEE Computer Society, 2014.
- [11] F. Luan, S. Paris, E. Shechtman, and K. Bala. Deep painterly harmonization. *CoRR*, abs/1804.03189, 2018.
- [12] A. Mandal. *MTCNN Face Detection and Matching using Facenet Tensorflow*, 2018 (accessed Feb 18, 2018).
- [13] T. Muraki, S. Oishi, M. Ichino, I. Echizen, and H. Yoshiura. Anonymizing face images by using similarity-based metric. In *Availability, Reliability and Security (ARES), 2013 Eighth International Conference on*, pages 517–524. IEEE, 2013.
- [14] Y. Nirkin, I. Masi, A. T. Tran, T. Hassner, and G. Medioni. On face segmentation, face swapping, and face perception. *arXiv preprint arXiv:1704.06729*, 2017.
- [15] E. Osuna, R. Freund, and F. Girosit. Training support vector machines: an application to face detection. In *Computer vision and pattern recognition, 1997. Proceedings., 1997 IEEE computer society conference on*, pages 130–136. IEEE, 1997.
- [16] Z. Ren, Y. J. Lee, and M. S. Ryoo. Learning to anonymize faces for privacy preserving action detection. *arXiv preprint arXiv:1803.11556*, 2018.
- [17] R. Rothe, M. Guillaumin, and L. Van Gool. Non-maximum suppression for object detection by passing messages between windows. In *Asian Conference on Computer Vision*, pages 290–306. Springer, 2014.
- [18] N. Ruchaud and J.-L. Dugelay. Automatic face anonymization in visual data: Are we really well protected? *Electronic Imaging*, 2016(15):1–7, 2016.
- [19] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.
- [20] L. Souza, M. Pamplona, L. Oliveira, and J. Papa. How far did we get in face spoofing detection? *CoRR*, abs/1710.09868, 2017.
- [21] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1701–1708, 2014.
- [22] R. Vaillant, C. Monrocq, and Y. Le Cun. Original approach for the localisation of objects in images. *IEE Proceedings-Vision, Image and Signal Processing*, 141(4):245–250, 1994.
- [23] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE, 2001.
- [24] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503, Oct 2016.