CS230

Inverting Facial Embeddings with GANs

Edward Vendrow
Department of Computer Science
Stanford University
evendrow@stanford.edu

Akash Singhal
Department of Mechanical Engineering
Stanford University
akash13@stanford.edu

Abstract

Compression and regeneration of images is an extremely relevant task in the modern age. We seek to recreate images of people's faces by only storing a vector (embedding) that captures their identity. We implement many methods to invert this embedding into the face in the original image. Using a Generative Adversarial Network (GAN) to generate faces, we propose a greedy search algorithm on the latent vector space of the GAN to find a matching face. Our process yields latent vectors that generate images capable of fooling FaceNet [13], a state-of-the-art face recognition system.

1 Introduction

With the advent of the digital age, the ability to store large amounts of data has become extremely important. For most people, on a daily basis, this can involve storing large amounts of audio, video, and image files. The image and video files take an astonishingly large amount of space to store given the modern high resolution images and videos available. For this reason, the notion of regenerating data from its compressed format, especially for images, has recently become a very popular topic of research. The goal of this project was to be able to reasonably recreate an image of a face simply from its embedding - a small vector representing the identity of the subject provided by a face recognition network such as FaceNet. The problem of generating high resolution output images from low resolution input images has been extensively researched and has led to good results. However, the idea of being able to recreate the face of a person just from its embedding is still relatively unexplored. Along with applications in face image storage reduction, recreating the original desired face from an embedding has many other applications. For example, the latent representation of the face can be slightly modified to make interesting modification to the resulting image, such as adding sunglasses, changing facial expressions, or adding the appearance of age.

For this project we attempt to generate images of a faces from only their 128-dimensional embedding vectors. We begin by recreating the results of Zhmoginov et. al. [1] on inverting facial embeddings one identity at a time and discussing the problems with their approach. Then, we explore the use of Generate Adversarial Networks to generate faces from some latent vector space. Using a modified greedy search with a large pre-generated identity-to-latent-vector dataset, we perform a search over the StyleGAN[12] latent vector space to find faces matching the desired identity embeddings.

2 Related work

The problem of recreating a face from its embedding (without using auto-encoders) is still relatively unexplored in research. As such, Zhmoginov et. al. [1] use the FaceNet embedding of an image as their input and a guiding image to be able to generate a face that looks identical to the one used for the original embedding. After employing different regularizing weights and modifications, the authors are able to to generate faces similar in identity, but clearly fake. We will try to replicate some of the ideas developed in [1] and get similar results for one part of the project. Realizing that the images will always look fake, we moved our attention to use a Generative Adversarial Network (GAN) to generate images that look real and are reasonably close to our target embedding.

Since the introduction of the concept of a GAN, a lot of Computer Vision research has focused on developing different GAN architectures. In lieu of recreating faces, [2] employs a FaceID-GAN that introduces a third block, a classifier, along with the Generator and Discriminator. The purpose of the classifier block is to learn the identities of the faces during the training. FaceID-GAN works extremely well, however, its input is an image instead of an embedding. Another GAN architecture that achieves similar results as [2] on certain datasets, is introduced in [3] as TP-GAN. This model introduces an identity preserving loss, which uses a Light CNN for its identification and verification. Although this model is also used to generate frontalized images, the concept of a identity preserving loss made intuitive sense. Hence, in one of our implementations, we also tried incorporating an identity preserving loss using the last two layers of the FaceNet model. Another GAN architecture that seeks to generate identity preserving faces but again requires image inputs (and not embeddings) is MP-GAN [4]. This architecture is different in that it also uses identity feature extractors but has multiple stages of Generators and Discriminators to successively generate higher resolution images at each stage. The identity preservation results are extremely good and this architecture gave us more conviction towards using an identity preserving loss for our face generating GAN model.

To use identity loss, we needed to train a GAN by adding an identity loss to the typical adversarial loss of a GAN. The state-of-the-art GAN model for generating faces uses a Style based Architecture Model for GANs as detailed in [5]. Other architectures, which give good results for face synthesis at lower resolutions include the DCGAN model as it uses a more complex architecture than a typical GAN as depicted in [6]. Other GAN architectures we explored included the BE-GAN model detailed in [7]. This architecture has gained popularity since its release for its implementation of an equilibrium between the Generator and the Discriminator.

After generating images from the GAN, we attempt to recover the latent vector that would yield an image with the same embedding as the input image. An interesting method explained in [8] essentially generates some images from the GAN network and finds the image that has the closest embedding to the original image. They then use this new image's latent vector and try to modify it by adding some random noise to it. Another method is outlined in [9] which uses gradient descent to recover the latent vector that generated a particular image. For this purpose they use a mean squared error as the loss function.

3 Dataset and Features

For training the DCGAN (with and without the identity preserving loss) implemented to generate faces, we used 50,000 images from the CelebA dataset. Before training, we pre-processed the dataset by using OpenCV to detect each face and crop the image around it, then resize to 160x160 which is the input size to FaceNet. We then run FaceNet across all images the get embeddings for later use.

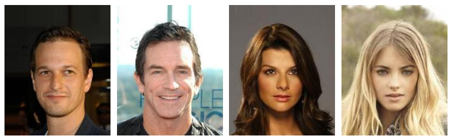

Figure 1: Example images from the CelebA dataset.

4 Methods

4.1 Direct inversion of facial embeddings

We begin by replicating the result of [1] on inverting face embeddings with convolutional neural networks. We use a FaceNet model mapping normalized 160x160 face images to a 128-dimensional "embedding" vector. Authors of [1] propose using gradient descent on this network to reconstruct a recognizable face just using the target embedding vector. We initialized an image as random noise, and set our initial loss function as $-e_1 \cdot e_2$, where e_1 is the generated embedding and e_2 is the target embedding. We use dot product loss instead of l_2 loss at the recommendation of [1]. Then, we add total variation loss, defined as:

$$\mathrm{L}_{tv}(p) = \left(||p - \hat{S}_x p||_2^2 + ||p - \hat{S}_y p||_2^2 \right)^{1/2}$$

Where $\hat{S}_x$ and $\hat{S}_y$ shift the image by one pixel right or down, respectively. This variation penalizes large differences in adjacent pixels, thus making the resulting image smoother. We then added guiding image regularization loss as $||a(p) - a(g)||^{22}$, where $a(x)$ gives the activation of an intermediate layer of FaceNet. This gives the squared error of the activations of an intermediate layer between the generated image and some guiding image. It is observed that using an earlier layer preserved lower-level features such as colors and textures, while later layers preserved higher-level facial features.

4.2 Searching over a latent vector space

Generative Adversarial Networks (GANs) have demonstrated outstanding performance in generating realistic photographs. Generally, the resulting generators accept a normalized vector (often called the latent vector) as input to generate an image.

Finding a generated image matching the desired identity is a difficult problem. Generators have no predetermined face generation patterns and very different latent vectors can result in similar-looking faces. Using gradient descent to find the desired image is thus generally infeasible, as this is a non-convex optimization problem. A more common approach is to conduct a search over the latent vector space of the generator as outlined in [9]. One common strategy is to repeatedly generate images from random noise, choose the best result, and shrink the magnitude of the noise until convergence. Often, these results are limited both by computational power and by the quality of the GAN. For example in [8], the algorithm does well, but the results do not look as visually appealing because the the generator cannot generate high resolution images.

We propose a modified strategy to find the desired latent representation of the desired identity. Furthermore, we will use NVIDIA's StyleGAN, which achieves state-of-art results in face generation quality. Since initial stages of the search involve a long and expensive search over randomly-generated latent vectors, we can significantly speed up this initial search by pre-generating pairs of latent vectors and the corresponding FaceNet embeddings created by generating an image, aligning, normalizing, then running the result through FaceNet. Using these generated pairs, it is very fast to determine the closest embedding vector to a target embedding, and find the corresponding latent vector used to generate it. Then the rest of the search proceeds from there.

Subsequently, we perform a greedy search to find an even closer match. Starting with the closest latent vector, we generate many embeddings by adding noise and running the resulting vectors through the generator and FaceNet. Then, we choose the best result, set this as our new starting point, and repeat. By decreasing the standard deviation of the input noise, we converge on a solution.

5 Experiments/Results/Discussion

5.1 Direct inversion of facial embeddings

We began by experimenting with direct inversion of facial embeddings as proposed in [1] while gradually incorporating each of the losses they use. We started with simple dot product loss on the resulting embedding of the image. We found the most interesting results were observed using gradient descent with learning rate decreasing slowly from 0.01 to 0.0001. Then, we added total variation regularization. From a series of trials across different weights, we obtained the best results with a L_{tv} weight of 15 (where the FaceNet embedding weight was 1). Finally, we added guiding image regularization. Best results were obtained using an intermediate layer closer to the end of the network, with a weight of around 2. Our results follow:

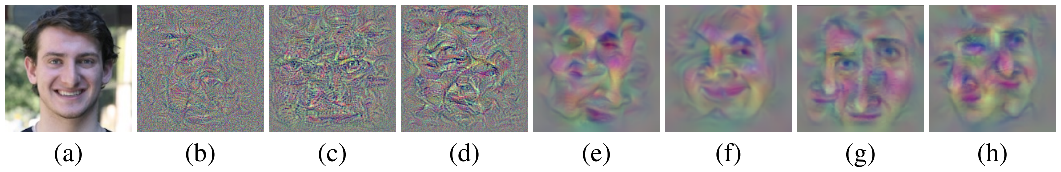

Figure 2: (a) is the desired image whose FaceNet embedding is used. (b) and (c) are generated using just facenet embedding loss. (d), (e), (f) are geenerated using increasing total variation loss, making an increasingly smother and more image. (g) and (h) are generated using guiding image loss on an intermediate FaceNet layer. We observe increasing identifiability, but not a convincing result.

Here the limitations of this approach are evident. It is computationally expensive to train this network on each identity separately, and the results are generally not very good. Even in the mean squared error of the activations of an intermediate layer of FaceNet between the generated image and some guiding image. Even [1] mentions that while this approach often preserves important facial features, the resulting image is clearly not real, especially since there is no penalty for fake-looking images in the loss. Instead, the authors recommend exploring GANs as an alternative approach to this problem. Utilizing the advice in [1] and realizing that without GANs, getting real-looking images from embeddings will be infeasible shift our approach to use GANs.

5.2 Identity-based search over the StyleGAN latent vector space

We began by training our own GAN to generate faces on the 50k image CelebA dataset using the common DCGAN architecture. After 12 hours of training, we had the following results:

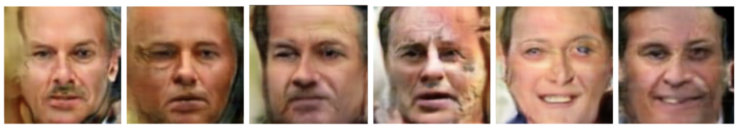

Figure 3: Results from training our own GAN

Although the resulting images began looking like faces, these were not good enough for our purposes. Instead, we use much better pre-trained networks for this purpose. NVIDIA released pre-trained weights for the generation of 1024x1024 face images, trained for days on hundreds of powerful GPUs. We decided to use this generator instead of our own due to its higher quality results.

We begin by pre-generating pairs of latent vectors and the corresponding FaceNet embeddings. Over 10 hours of computation, we generated 250,000 such pairs. Subsequently, we perform a greedy search previously outlined, starting with the closest latent vector just found. In our testing, we found best results by running 50,000 iteration with noise of standard deviation 0.5, then reducing to 0.3 and iterating 50,000 more times. It is not necessary to reduce noise further since smaller changes to the latent vector do not significantly affect the identity of the resulting image. Indeed, we found that adding noise with smaller than a 0.2 standard deviation changed the resulting image very little.

The following are some results from this search on real, randomly-chosen guiding images. We calculate FaceNet embedding loss as the l_2 norm of the difference in the embeddings of two images. For reference, we expect to see a loss of 4 to 6.5 for pictures of the same person, and a loss of above 14 for different people.

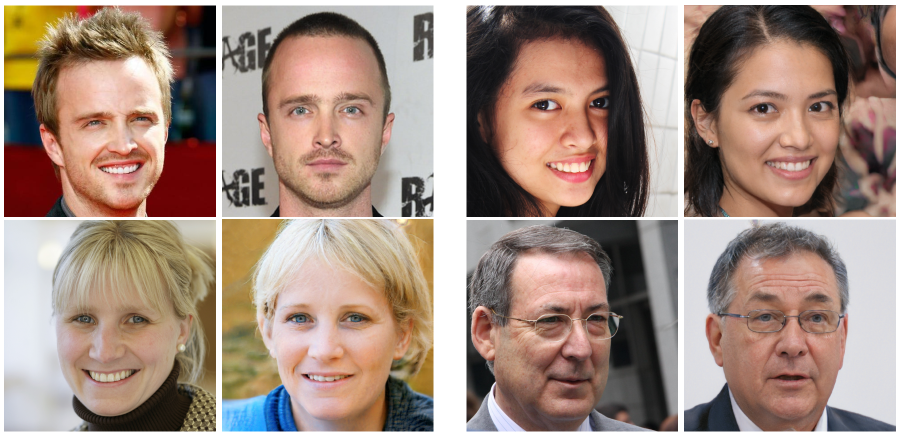

Figure 4: The top-left two images, both of actor Aaron Paul, have a FaceNet embedding loss of 6.112. In each of the following image pairs, the FaceNet embedding of the left image was used to recover the second image using our latent vector space search. The FaceNet embedding losses for the top-right, bottom-left, and bottom-right pairs are 6.736, 6.153, and 6.929 respectively.

We observe that the recovered images are fairly similar in identity to the images used to generate them. Importantly, with the bottom-left image we were able to fool FaceNet, as the FaceNet embedding loss was almost identical to the loss between the images of Aaron Paul.

We believe that much of the difficulty of generating an appropriately matching image comes from the types of faces generated by the GAN. We attempt to quantify this on our 250,000-sized list of pairs of latent vectors and embeddings by finding the best embedding match in the list for a given embedding. We observe that when inputting a generated embedding, the best similar embedding (ignoring itself) has an average FaceNet loss of 7, which is fairly good. However, using images from CelebA, we get a much higher average best loss:

Data Type	Mean Best Loss	Median	Standard Deviation
Generated	7.057	7.171	1.085
CelebA	10.200	10.252	0.862

This suggests that the generator creates certain faces with much higher likelihood than others given random noise. Thus it is easier to find a generated face matching another generated face, but it is difficult to find a face matching a real image.We suspect that this could also be a direct consequence of the generated images having a very different feature space to the real images, which also makes it hard for any generated image to completely replicate a real image.

6 Conclusion/Future Work

The goal of this project was to be able to recreate faces from their FaceNet embeddings. Our initial approach to solve this problem with guiding images and without a GAN successfully demonstrated the ability preserve the identity of the original image, even the resulting image would be discernibly fake. With StyleGAN we see the opposite problem: face generation is convincing, but it is difficult to find the correct latent vector. Our search algorithm gives promising results as a search over the latent vector space of StyleGAN. The final images both look convincing, and have similar identities to the guiding embeddings.

Given the relatively unexplored nature of this problem, there are many ideas and paths to explore. With more time, we would have liked to further probe into other search algorithms to find generated images that have embeddings as close to the original image as possible. Other ideas that we would have liked to implement would have been the conditional re-sampling of the latent vector during the gradient descent approach of recovering the latent vector. This method was implemented in [10] and with more time it would be worth exploring this avenue.

7 Contributions

Edward: Wrote code to pre-process data and train the various models used in the paper and researched on several current ideas of inverting facial embeddings along with novel GAN architectures such as FaceID-GAN.

Akash: Helped with the code and researched several approaches to inverting facial embeddings, including a survey of existing GANs and latent vector search approaches.

GitHub Code: http://github.com/evendrow/cs230

References

[1] Zhmoginov, Andrey and Mark Sandler. "Inverting face embeddings with convolutional neural networks." ArXiv abs/1606.04189 (2016): 1-12

[2] Shen, Yujun et al. "FaceID-GAN: Learning a Symmetry Three-Player GAN for Identity-Preserving Face Synthesis." 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (2018): 821-830.

[3] Huang, Rui et al. "Beyond Face Rotation: Global and Local Perception GAN for Photorealistic and Identity Preserving Frontal View Synthesis." 2017 IEEE International Conference on Computer Vision (ICCV) (2017): 2458-2467.

[4] Tang, Zhiyong et. al. "Multi-Process Training GAN for Identity-Preserving Face Synthesis." IEEE Access, vol. 7, pp. 97641-97652, 2019.

[5] Karras, Tero et al. "A Style-Based Generator Architecture for Generative Adversarial Networks." CVPR (2018).

[6] Radford, Alec et. al. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks." ArXiv abs/1511.06434 (2015): 1-16

[7] Berthelot, David et al. "BEGAN: Boundary Equilibrium Generative Adversarial Networks." ArXiv abs/1703.10717 (2017): 1-10

[8] Li, Zhigang et. al. "Generate Identity-Preserving Faces by Generative Adversarial Networks." ArXiv abs/1706.03227 (2017): 1-9

[9] Egan, Nicholas et. al. "Generalized Latent Variable Recovery for Generative Adversarial Networks." ArXiv abs/1810.03764 (2018): 1-9

[10] Lipton, Zachary C. et. al. "Precise Recovery of Latent Vectors from Generative Adversarial Networks." ArXiv abs/1702.04782 (2017): 1-4

[11] Liu, Ziwei, et al. "Deep Learning Face Attributes in the Wild." Proceedings of International Conference on Computer Vision (ICCV), 2015.

[12] NVlabs. "NVlabs/Stylegan." GitHub, 3 Dec. 2019, https://github.com/NVlabs/stylegan.

[13] Schroff, Florian, Dmitry Kalenichenko, and James Philbin. "FaceNet: A Unified Embedding for Face Recognition and Clustering." 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015): 1-10 Crossref. Web.