Photoshop 2.0: Generative Adversarial Networks for Photo Editing

Homero Roman Roman, Brandon Yang, Michelle Zhang
Stanford University
{homero, bcyang, mzhang8}@stanford.edu

Abstract

In this paper we explore how to use Generative Adversarial Neural Networks (GANs) to generate realistic face images that posses certain desirable facial characteristics. Moreover, we develop a procedure to take a real image and generate a similar image with specific facial characteristics adjusted by manipulating the vectors in the latent space. We mainly focus on experimenting with encoder networks for the generator and how the latent space of an image affects the generated result. By manipulating this vector space we are able to edit photos to add attributes. For instance, we are able to take a batch of non-smiling face pictures and add a smile to them on the generated output images.

1. Introduction

In recent times, neural networks have made huge strides in the tasks of image classification and segmentation. Neural networks have been able to surpass traditional methods of specific feature extraction by leaving the network itself with the task of recognizing features and learning weights from those. Convolutional Neural Networks (CNNs) in particular have shown great performance in being able to manipulate image data and do a variety of tasks that range from semantic segmentation to object detection and image captioning. These networks train by minimizing the loss over a train set and testing on an evaluation set to ensure generalizability.

GANs are a type of neural network that uses convolutions to train on images but unlike regular CNNs, GANs work by playing a game of min max between a discriminator and a generator. First, the discriminator is a network that trains to be able to classify real and fake images. The discriminator returns a metric of how confident it believes an input image is a real image where a very high score would indicate that that it strongly thinks it is a real image. Images are modeled as though being drawn from two separate distributions. The distribution of real images is modeled as being specified by the real world inputs while the distribution of generated images is determined by the outputs of the generator. The goal of the discriminator is to maximize its score for images that are actually from the training data and minimize its score for images generated by the generator. On the other hand, the generator is tasked with fooling the discriminator by generating fake images. As the generator trains alongside the discriminator it is able to generate images that are more and more realistic to the point that they are indistinguishable from real data.

Sadly, the original implementation of GANs by Goodfellow [8] suffers from mode collapse, where the generator collapses into a very narrow distribution that only covers part of the theoretical data distribution. This is very serious because it prevents generalizability by making the generator only able to produce images within a small range of similar samples. Also the original GANs have no clear metric about convergence. While minimizing the loss can be interpreted as being able to better learn from the input data, it is still not clear how that is reflected into better image quality.

Generator networks are commonly used to generate new images for datasets like CiFAR 10, and lots of research has focused on improving image quality and realism. Some recent research has looked into the semantic understanding of these generator networks and the latent space used to generate these images.

1.1. Contributions

Our main contribution to GANs is investigating the knowledge representation of GANs in the latent space. We first start by investigating the creation of autoencoder networks for GANs that turn real images into vectors in the latent space that generate images as close as possible to the original image when propagated through the generator network. We experiment with different encoder architectures, revealing properties about the generator and the latent space distribution. We then harness these encoder networks to study various properties in the latent space. For the faces dataset, we investigate vectors in the latent space corresponding to facial characteristics like smile and hairline. By identifying these vector transformations in the latent space, we develop a procedure with the encoder network to enhance real images with desired facial characteristics. For

instance, we are able to successfully add a smile to non-smiling images by simple addition of the smiling attributes to an image encoding in the latent space.

2. Related Work

Our work relates to previous research conducted on various GAN models including DCGANs and WGANs, conditional GANs, variational autoencoders, and recent experiments in photo editing using generative image models.

DCGANs and Improved Wasserstein GANs. Radford et. al proposed a new class of CNNs known as deep convolutional generative adversarial networks (DCGANs), which can then be reused via their discriminator and generator networks for feature extraction [20]. Prior to this, GANs implemented with a CNN architecture proved to have little success as in the LPGAN [5]. It was demonstrated that by applying dropout to neurons in the second to last layer of the generator network, the presence of windows in bedroom images of the LSUN dataset could be removed. Arithmetic between the learned latent space representations (i.e. the $\mathbf{z}$ vectors) of the images produced a $\mathbf{z}$ vector that agreed semantically with the operation. A more recently introduced type of GAN is the Wasserstein GAN (WGAN) [2, 9], which places less focus on the particular design of the network architecture and alleviates the instability issues related to traditional GANs. The image generation of such WGANs are comparable to that of the DCGAN and can therefore also provide some insight on the semantic understanding of GANs with less challenges in training the GAN itself.

Invertible Conditional GANs. The conditional version of GANs allows for the generation of images on a specified class of a dataset and have been applied in several settings to the MNIST digits dataset and face datasets such as CelebA to impose desired attributes on the images, as in aging faces [1, 4, 7, 17]. Invertible cGANs have followed, in which an encoder is paired with a cGAN in order to produce an inverse mapping of an image into the latent space [18]. This allows the model to now attain the ability to reconstruct an image conditional on some stated attribute (e.g. change a given person's hair color). This method has shown great promise in manipulating the latent space to take advantage of the semantic understanding of the GAN networks.

Variational Autoencoders. Variational autoencoders, at a high level, takes some input image and outputs an encoding of the image in the form of a vector. Numerous such autoencoders have been explored in order to model and analyze images [14, 19]. In combination with GANs, VAEs may be applied to allow for encoding in addition to generating an image sample. The approach is derived from the variational Bayesian method of optimizing approximation of the posterior distributions of probabilistic models through a more practical estimate for the variational lower bound [13]. In addition, other autoencoders as in the adversarial autoencoder (AAE) [16] and bidirectional generative adversarial network (BiGAN) [6] have extended the training criteron of VAEs to better capture the data manifold of $\mathbf{z}$ or simply better incorporate the encoder into the GAN architecture.

Photo Editing with GANs. From the development of new and improved GAN models, we consequently see a significant increase in the quality of image generation. Therefore, it is not surprising to see the rise of applications beyond simply manipulating the images through these models. The Neural Photo Editor [3] is an interface that uses a hybrid of a VAE and GAN to allow for direct manipulation of face images paired with "contexual paintbrush" and a provided visualization of the latent space. Heinrich, a NVIDIA engineer, too, has created an interactive web interface [10] to allow users to train encoders with a DCGAN implementation for the MNIST and CelebA datasets and visualize the generated images through specifying the various attributes for faces.

3. Dataset and Features

We chose to use the CelebA dataset [15], collected by researchers at the Chinese University of Hong Kong. While its original purposes were to be employed in face attribute recognition and face detection, the images have been applied by Neural Face, an implementation of a Deep Convolutional Generative Adversarial Network (DCGAN) by Facebook AI Research used to generate images of faces [11].

This large-scale dataset includes over 200,000 face images of various celebrities, with a great diversity of facial features and poses. Each 178×218-pixel RGB image is also annotated with 40 different binary attributes that include hair color, gender, and relative age, and 5 major landmark locations. For example, a celebrity in a particular image may be categorized as a female, having eyeglasses, bangs, and an oval face but is not smiling.

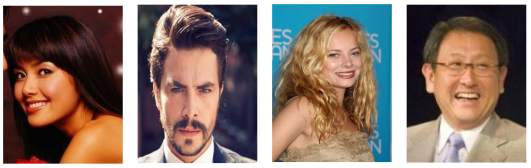

Figure 1. Sample of unmodified, "in the wild" face images from the CelebA dataset [11].

Figure 1 shows a random sample of images from the dataset. For our DCGAN autoencoders, we have the flexibility to use our own generated images, provided they are realistic enough. However, because the CelebA dataset is already annotated, this is more useful in extracting the $\mathbf{z}$

vectors that correspond to particular traits in the images. When training the autoencoders, we split the set of images into a 70/15/15 ratio for training, hyperparameter tuning, and testing respectively.

3.1. Data Preprocessing

For convenience, the images provided in the CelebA dataset have also already been appropriately aligned and cropped. Due to the difficult nature of training larger images in GANs, we perform further cropping to the dimensions 64×64 and resizing of the images to reduce their size and any background clutter. To reduce noise, we took an additional step, as was also taken in the original implementation of the DCGAN being used [20], for training images and scaled them to the range of [-1, 1] of the tanh activation function. After sampling the images, we rescale in order to visualize results. No further pre-processing was done on the images.

3.2. Biases and Noise

The dataset suffers from some biases that affects the training of the models. Namely, there exists a greater majority of images of female celebrities, and therefore, the autoencoder is more prone to producing an encoding that generates a face with more feminine facial features. There also is a significantly lower proportion of images with certain attributes as in having eyeglasses. Moreover, the images are noticeably heterogeneous, and face images at unusual angles (i.e. not frontal-view) may be much harder to learn.

4. Methods

We begin by training our generative model. We use the deep convolutional generative adverserial networks proposed by Radford, Metz, and Chintala for our generator and discriminator models [20]. In particular, we use the Neural Face implementation of the original DCGAN[11] to train our models. We trained this model for 25 epochs over the entire preprocessed CelebA dataset, using the Adam optimizer [12] with a learning rate of 0.0002 until the loss curve converged. We then sampled the generator to visualize the generated images as in Figure 2.

We then use our trained generator and discriminator to build our encoder architecture. In the original GAN architecture, the generator input $\mathbf{z}$ is a vector of random input between 0 and 1 sampled from a Gaussian distribution. In order to better visualize and learn about the latent space and apply visual transformations to real images, we explore different models to create a GAN autoencoder. The objective of the GAN autoencoder is to take an input image and produce a $\mathbf{z}$ vector in the latent space, such that the generated output is as close as possible to the original image. We train the encoder models with the architecture in Figure 3. Our approach to creating and training the GAN autoencoder is inspired by Heinrich's approach to photo editing [10].

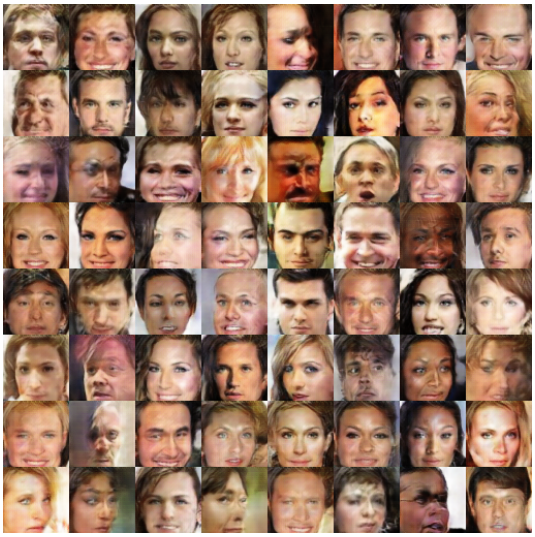

Figure 2. Sample of generated images.

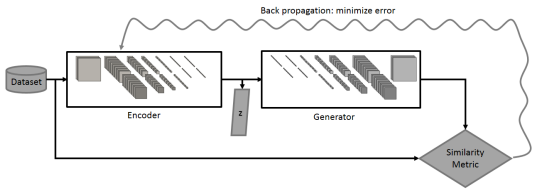

Figure 3. Network architecture for training the GAN autoencoder [10].

4.1. Encoder Models

Using this model architecture to train our encoder, we experiment with different encoder models for generating encodings in the latent space. To train these models, we divide the CelebA dataset into training, validation, and test sets as described above. We then train the encoder architecture on the training images in the CelebA dataset, by inputting an image from the dataset and measuring the loss as the result of a given similarity metric as shown in Figure 3. We then use the validation set for hyperparameter selection and to explore the effects of different architectures. Finally, to evaluate our encoders, we run our encoders on the test set and evaluate the average L1 loss, L2 loss, and SSIM loss for the test set. In addition, we sample images from the evaluation set and run them through the encoder and decoder architecture to report qualitative results.

4.2. Baseline

As a baseline result for our encoder network, we trained a 4-layer fully connected neural network with standard ReLU non-linearities, as shown by the FC encoder in Figure

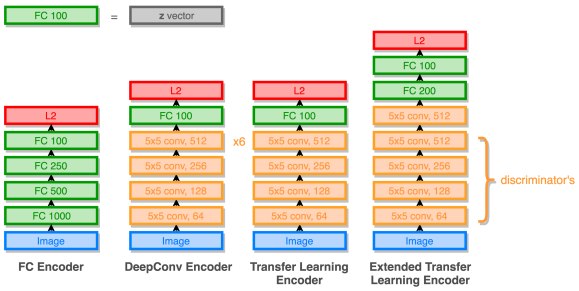

Figure 4. Encoder network architectures.

4 using the L2 loss and the Adam optimizer with a learning rate of 0.0002. The L2 loss for an individual example is given by

$$L_2(i_1, i_2) = \sum_{j=1}^{n} (x_j - y_j)^2$$

between the original image i_1 and generated image i_2 and average the losses of all the examples in a batch for the data loss.

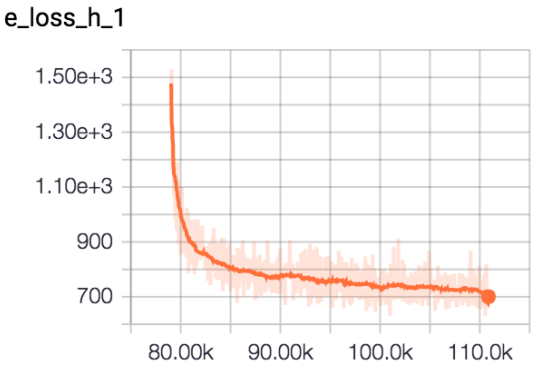

Figure 5. L2 training loss curve for the fully connected encoder.

4.3. Deep Convolutional Models

We then experimented with deep convolutional architectures for GAN encoding. Our final deep convolutional GAN model is shown in Figure 4, consisting of 9 convolutional layers separated by batch normalization and Leaky ReLU non-linearities followed by one final fully connected layer. We train our model with the L2 loss and the Adam optimizer with a learning rate of 0.0002.

4.4. Transfer Learning

After training these networks from scratch, we experimented with using transfer learning from the discriminator network. The discriminator network has already

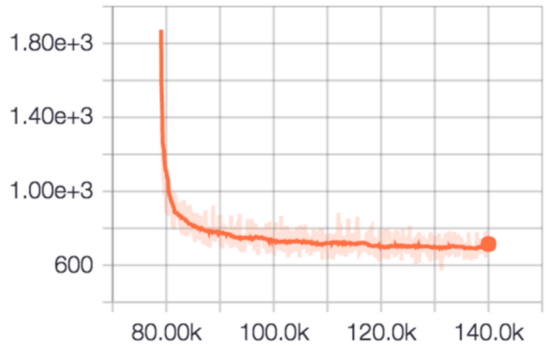

Figure 6. L2 training loss curve for the transfer encoder.

been trained for 25 epochs and uses the weights to predict whether or not an image is real or fake. Thus, the discriminator network is a great candidate for transfer learning for the autoencoder in its ability to extract important information from a given image [10]. As a simple transfer learning encoder, we re-use the weights from the discriminator at every layer, but retrain the last layer with a new fully connected network to produce the 100 dimensional $\mathbf{z}$ vector. We then extend upon this simple transfer learning encoder by training an additional convolutional layer followed by two fully connected layers, with batch normalization and leaky ReLU non-linearities. These two models are illustrated in Figure 4. We freeze the transfer layers from the discriminator in both models and optimize with the L2 loss and the Adam optimizer with a learning rate of 0.0002. The training curve for the transfer learning model was remarkably smooth and fast to converge, as shown in Figure 6, even when compared to that of the baseline shown in Figure 5.

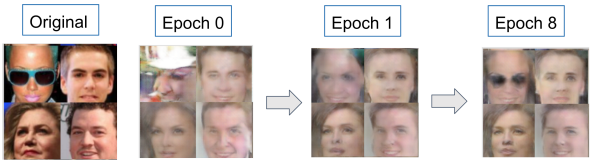

Figure 7. Training samples for the transfer encoder during various epochs.

We visualize the training process for the transfer encoder as shown in Figure 7. In particular, the model at first struggles for non-face attributes like glasses and hats, which completely disrupt the encoding of images, and the encoder outputs a similar face for every face. As training continues, the encoder learns to differentiate between different characteristics in the latent space, and by the 8th epoch represents glasses and various facial features well.

4.4.1 Similarity Metrics

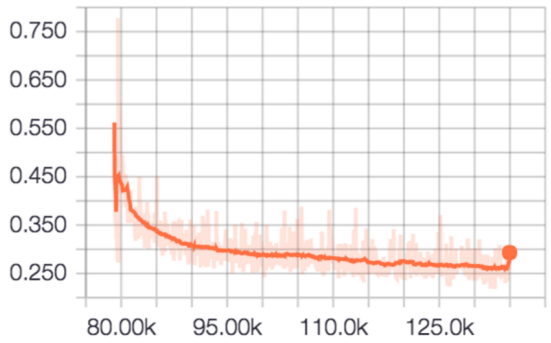

Figure 8. SSIM training loss curve for the transfer encoder.

After analyzing the output of our models, the sampled images often suffer from blurry patches, which did not seem to improve as L2 loss decreased. To address this issue, we experimented with different similarity metrics to improve qualitative encoding results for human observers. We evaluate the effect of different similarity metrics for the loss function on output results using the transfer learning encoder, since this was the fastest to train. As described by Zhao et. al, although the L2 loss is most commonly used in computer vision tasks, the L2 loss is a poor measure of image quality as measured qualitatively by human observers, because it does not pay enough attention to local characteristics of images [21]. We first use the L1 loss similarity metric, since the L1 loss has been show to improve splotchy artifacts that arise from using the L2 loss similarity metric [21]. We then try a state of the art image similarity metric that is motivated perceptually called the Structural Similarity metric, or SSIM, to better account for local structure [21].

Explicitly, we calculate the loss as a sum of the SSIM index and a weighted L2 norm as our regularization term as so:

$$Loss = 1 - \text{SSIM} + 0.00005L_2$$

where

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

is the perceptual difference between two windows x and y of size $N \times N$ and $c_1 = (k_1 L)^2$ and $c_2 = (k_2 L)^2$ are variables used to stabilize the denominator, with default values $k_1 = 0.001$, $k_2 = 0.03$, and $L = 2^{\text{bits per pixel}-1}$ The weighted L2 norm is used to better combat the problem of being stuck in local minima. For the SSIM loss, we use an increased learning rate of 0.001.

4.5. Latent Space

Motivated by the latent space manipulation for generating images with different attributes shown in the original DCGAN paper and the NVIDIA blog post, we apply our encoder to image editing. To do this, we first represent the attributes in the CelebA dataset as vectors in the latent space. In the CelebA dataset, each image is labeled with one of 40 attributes. We first compute the **z** vectors representing each attribute. To do this, we subtracted the average **z** vector of all images that do not have the specific attribute from the average **z** vector of all images that have the attribute in the training data. For instance, to compute the attribute for "Male" we encode all training male images and take the average vector **m**, then we encode all training non-male images into average vector **f** and we get the "Male" attribute vector by subtracting **f** from **m**. Simply put, **z = m - f**.

We then manipulate images by first encoding the image in the latent space with the encoder network, as described above, and then adding the attribute vector to the encoded image vector. We then explore manipulating images with different attributes.

4.6. Results

4.6.1 Encoder Quantitative Results

To report our quantitative results, we evaluated our trained encoder models on our withheld evaluation set representing 15% of the data. We then calculate the average L1 loss, L2 loss, and SSIM loss between the encoded image and original input image for the test set and report this for each image. Each of these loss functions captures different aspects of the performance of our autoencoder network with the generator. The quantitative values of the loss on our test sets are shown in Figure 9. Overall each encoder performed best on the loss it was trained to minimize with the exception being the Deep Conv Encoder which had the lowest losses all across the board.

	L2 Loss	L1 Loss	SSIM Loss
FC Encoder (L2 loss)	721.280	2192.837	0.305
Deep Conv Encoder (L2 Loss)	580.059	1949.630	0.266
Transfer Encoder (SSIM Loss)	1082.709	2586.792	0.268
Transfer Learning Encoder (L1 Loss)	744.001	2137.577	0.303
Transfer Learning Encoder (L2 Loss)	697.754	2140.374	0.305
Extended Transfer Learning Encoder (L2 Loss)	692.799	2110.645	0.302

Figure 9. Quantitative evaluation results on test set for encoders.

4.6.2 Encoder Qualitative Results

As we can see in the Figure 10 and Figure 11 below, our Deep Convolutional Encoder is able to successfully generate **z** encoding vectors that can later on be decoded by the generator into realistic images. However, some issues that arise are that the generated images may appear a bit blurry. As discussed before, this is likely due to the the use of the L2 loss which we attempt to ameliorate by also trying the SSIM loss in a different architecture. Another interesting result is that unwanted words in a picture appear filtered out in the generated image. We can explain this by considering how the encoding of the image is in a way a compressed representation that mostly preserves the features that conform a face. This can also explain why the generator has a hard time generating non-face articles such as sun-glasses and why sometimes hats may appear similar to hair.

Figure 10. Original test images for the Deep Conv Encoder

We then compare the qualitative results of using different loss functions to optimize the same transfer encoder in Figure 12. As can be seen in the sample images, the output of the L2 loss function seems to capture many more of the artifacts, such as glasses. However, as a result, there are patches of blur and dark spots in the generated images. The output of the L1 loss function seems to capture less detail, although the faces on a whole appear to be much smoother. Finally, the SSIM loss does an extremely good job of realistically encoding and rendering eyes and skin with few artifacts, but the images seem to be less similar to the original images.

4.6.3 Latent Space Manipulation Results

We encode a sample of test images into the latent space and manipulate them with different attributes in Figure 13.

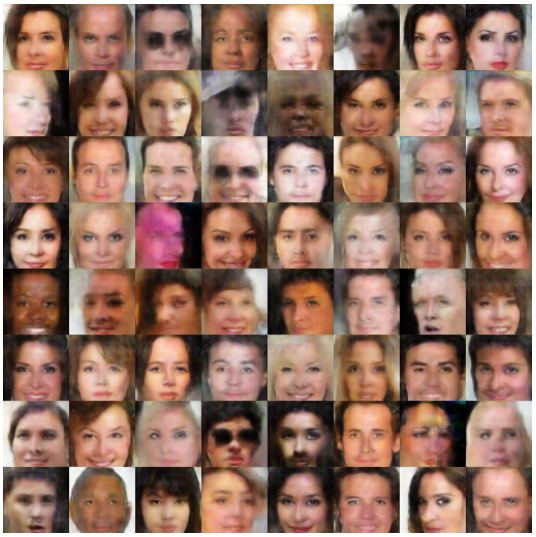

Figure 11. Generated images from Deep Conv encodings

Figure 12. Generated test samples with transfer encoder trained with L2 loss (left), L1 loss (middle), SSIM loss (right)

Below, we show the results for the addition of the smiling attribute to each image in Figure 14. Indeed, the images largely look similar to the originals when generated, and the addition of the smile vector can clearly be seen in the smiles of the facial expressions.

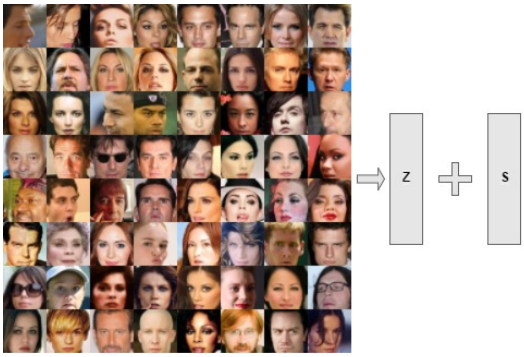

Figure 13. Addition of attributes in the latent space

Overall, we found that attributes with both sides well represented similarly are much more effective for latent space manipulation. In particular, the attribute "Male" did

Figure 14. Result of addition of "Smiling" attribute

not change the generated output much. The dataset is mostly composed of female pictures which likely fools the generator into adding feminine features to an image that was supposed to be male in effect canceling the effect of having added that attribute. Moreover, it's likely that this attribute is not well encoded in the latent space. The attribute for "Glasses" similarly is not well represented in the latent space. We hypothesize that glasses on the whole are rare in the dataset, and thus difficult to generate for the generator.

5. Conclusion

By training an encoder on top of the generator-discriminator architecture we were able to encode images and manipulate these encodings to edit photos by adding or removing specific attributes. We trained different encoder models and explored the use of different loss functions to encode qualitatively better images. We were then able to explore the semantic meaning encapsulated in the latent space and the potential for image manipulation with simple addition/subtraction algebraic operations in the latent space. In particular, we show that these manipulations work on encoded images and demonstrate a pipeline for image manipulation. Finally, we show results on unseen test images manipulated through our encoder and attribute vector architecture.

6. Future Work

Our model seems limited by inherent biases in the generator and dataset, which can perhaps be addressed with different generator approaches. For instance, the dataset is mostly composed of female faces which skews the generator into producing faces with feminine qualities (such as longer hair and eyelashes). The ability of the generator network to generate realistic and unbiased images is one of the major roadblocks to better image manipulation, and exploring different generator architectures to generalize better with skewed data and encode more information in the latent space $\mathbf{z}$ vectors could produce better results for the encoder and the image manipulation.

In addition, it could be effective to explore end to end architectures for image manipulation, rather than performing this image manipulation in two steps through the encoder and the attribute manipulation in the latent space. Moreover, exploring better quantitative metrics for generated image quality could be important to producing better image encodings and allow reporting of more precise quantitative results for image manipulation.

The latent vector space contains valuable information about an image, and allows a new approach to image manipulation. In the future, more complex operations in the latent space could lead to extraordinary results such as capture the full spectrum of emotional expression from one face to another.

References

[1] G. Antipov, M. Baccouche, and J. Dugelay. Face aging with conditional generative adversarial networks. *CoRR*, abs/1702.01983, 2017.

[2] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. *CoRR*, abs/1701.07875, 2017.

[3] A. Brock, T. Lim, J. M. Ritchie, and N. Weston. Neural photo editing with introspective adversarial networks. *CoRR*, abs/1609.07093, 2016.

[4] B. Dai, D. Lin, R. Urtasun, and S. Fidler. Towards diverse and natural image descriptions via a conditional GAN. *CoRR*, abs/1703.06029, 2017.

[5] E. L. Denton, S. Chintala, A. Szlam, and R. Fergus. Deep generative image models using a laplacian pyramid of adversarial networks. *CoRR*, abs/1506.05751, 2015.

[6] J. Donahue, P. Krähenbühl, and T. Darrell. Adversarial feature learning. *CoRR*, abs/1605.09782, 2016.

[7] J. Gauthier. Conditional generative adversarial nets for convolutional face generation. *Class Project for Stanford CS231N: Convolutional Neural Networks for Visual Recognition*, 2014.

[8] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. 2014.

[9] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. *CoRR*, abs/1704.00028, 2017.

[10] G. Heinrich. Photo editing with generative adversarial networks. `https://devblogs.nvidia.com/parallelforall/photo-editing-generative-adversarial-networks-1/`, Apr. 2017. Accessed: 2017-06-10.

[11] T. Kim. Neural face. https://github.com/carpedm20/DCGAN-tensorflow, 2016.

[12] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[13] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2014.

[14] A. B. L. Larsen, S. K. Sønderby, and O. Winther. Autoencoding beyond pixels using a learned similarity metric. *CoRR*, abs/1512.09300, 2015.

[15] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, Dec. 2015.

[16] A. Makhzani, J. Shlens, N. Jaitly, and I. J. Goodfellow. Adversarial autoencoders. *CoRR*, abs/1511.05644, 2015.

[17] M. Mirza and S. Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014.

[18] G. Perarnau, J. van de Weijer, B. Raducanu, and J. M. Álvarez. Invertible conditional gans for image editing. *CoRR*, abs/1611.06355, 2016.

[19] Y. Pu, Z. Gan, R. Henao, X. Yuan, C. Li, A. Stevens, and L. Carin. Variational autoencoder for deep learning of images, labels and captions. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2352–2360. Curran Associates, Inc., 2016.

[20] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2015.

[21] H. Zhao, O. Gallo, I. Frosio, and J. Kautz. Loss functions for neural networks for image processing. *CoRR*, abs/1511.08861, 2015.