Shoe Design using Generative Adversarial Networks

Hoang Le
Computer Science and Engineering
Pohang University of Science and Technology
`lemin1991@postech.ac.kr`

Hoang Nguyen
Management Engineering
Ulsan Institute of Science and Technology
`hoangnguyen3892@unist.ac.kr`

Abstract

In recent years, generative models have received a lot of attention in the scientific community. The ability of these models to generate samples that resemble the input data can be counted as a good indicator of models' capacity to learn the representation and underlying true distribution of the data. In this work, we explore the application of generative adversarial networks on the task of designing fashion items without any prior knowledge of that domain. Therefore, the goodness of the generative model is now evaluated not only in how exactly it can capture the real data distribution but also how realistic and inspirational the output is. Using available datasets or collecting more datasets with a slight modification, we will build and train a network that study design patterns and create new fashion items. Our models will also be evaluated and optimized by fine-tuning some hyper-parameters or changing the loss function.

1 Introduction

Since its recent resurgence, Deep Learning LeCun et al. (2015) has shown an excellent performance in many applications of computer vision, natural language processing, reinforcement learning, speech recognition, ranging from image classification Krizhevsky et al. (2012); Sermanet et al. (2013); Simonyan and Zisserman (2014); Szegedy et al. (2014); Zeiler and Fergus (2014); He et al. (2015) and image captioning Karpathy and Li (2015); Vinyals et al. (2015); Xu et al. (2015) to speech synthesis Zen et al. (2013); Wu et al. (2015) and acoustic modelling Hinton et al. (2012); Dahl et al. (2013).

Generative models are models that are able to generating new data points after being trained. In recent years, there has been a huge improvement of generative modeling in deep learning Goodfellow et al. (2014); Dosovitskiy et al. (2015); Radford et al. (2015a); van den Oord et al. (2016). Such models do not rely on explicit supervision and instead explain underlying distribution without it, in addition, it can determine intractable probabilistic computations caused by latent variables.

In recent work on unsupervised representation learning, *'adversarial nets'* framework has shown many promising results. This model is an alternative of inference network Jordan et al. (1999) and the Markov chain Monte Carlo (MCMC) method Neal (1993), which have been used widely by many researchers because of the ability of approximate posterior distribution. A subjective evaluation has been raising regarding to Generative Adversarial Networks as producing better samples than other methods Goodfellow (2017). Besides its attractive advantages, it has been known as being difficult and unstable to train.

In this work, we aim to explore an application of the existing deep learning model, namely *'Deep Convolutional Generative Adversarial Network'* Radford et al. (2015b), to the task of designing fashion - clothing, footwear, or accessories. In this task, the model should take on a role of a fashion designer who has a good fashion sense, a creative mind and a strong drawing skill. This is a very challenging task because the model does not just simply mimic the style. It should be able to capture the characteristics in terms of the shape, material and color of the fashion pieces, but also has to produce and visualize a mixed-and-matched style which is not similar with the one given in the dataset or at least the generated sketches should show some inspirational ideas. Obviously, it is difficult to achieve the ultimate goal, therefore, in this project, we attempt to train the model in such a way that it will be able to generate new valid samples.

2 Related Work

Two traditional approaches to generative models are parametric and non-parametric.

The non-parametric models usually make a match from a bunch of existing pictures. They are widely used in texture synthesis Efros and Leung (1999), super-resolution Freeman et al. (2002), and in-painting Hays and Efros (2007). For instance, the task of single image super-resolution, the model is required to impute the missing detail of the original data based on the information given by the sufficient input. This task can be done by exploiting the Markov network.

Parametric image-generating models have been focused extensively but they have not gained much success in generating natural images until recently. A typical example is an Auto-encoding Variational Bayes framework, where it learns an approximate inference model using the Stochastic Gradient Variational Bayes estimator. The output is generated from a random latent variable, which go through a trained decoder. This approach has had some moderate success since the sample outputs often suffer from being blurry Kingma and Welling (2013).

Another well-known approach is Generative Adversarial Networks (GANs) in which it applies value function of two-player minimax game to define loss functions for discriminative and generative models training at the same time. However, images generated from GAN suffer from being noisy and incomprehensible.

Especially, in the task where the goal is creating art, GANs seems to be taking the dominance among some recent work. Zhu et al. (2016) introduced iGAN, interactive Generative Adversarial Networks, helping users to create a new painting image from a few sketches through the interface. In addition, they did some experiences to manipulate the objects in terms of color and shape.

The Introspective Adversarial Network (IAN) Brock et al. (2016) is a hybridization of VAE and GAN which aims to enhance the power of adversarial networks while keeping the stability of variational inference method in order to apply into Neural Photo Editor. The output of this model should be photo-realistic and feature-aligned. The result shows that GANs can improve the quality of the output matching the user's desires.

3 Model

3.1 Generative Adversarial Networks (GANs)

GANs Goodfellow et al. (2014) is a composition of two separated networks, a generator (G) and a discriminator (D), which are trying to compete each other during the training time. The goal of the discriminator is distinguish between the real samples and the generated samples while the goal of the generator is generating the new samples trying to fool the discriminator. The following value function V(G, D) illustrates the theory behind GANs in which G and D are playing a minimax game:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[logD(x)] + \mathbb{E}_{z\sim p_z(z)}[log(1-D(G(z)))] \tag{1}$$

where $\mathbf{p_z}(\mathbf{z})$ is prior input noise variable distribution corresponding to latent variables $\mathbf{z}$ and $\mathbf{p_{data}}(\mathbf{x})$ is the distribution over the input data $\mathbf{x}$.

This target value function V(D,G) can be optimized by maximizing the two gradient ascent corresponding to G and D through two stages of backpropagation. It is shown that both D and G get stronger until G can sample $\mathbf{z}$ that follows training data distribution and D output the probability of $\mathbf{\frac{1}{2}}$ everywhere. The gradient ascent expressions for D and G are:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} [log D(x^{(i)})] + [log(1 - D(G(z^{(i)})))] \quad (2)$$

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} [log(1 - D(G(z^{(i)})))] \quad (3)$$

In the gradient ascent expression for the discriminator, the first term corresponds to optimizing the probability that the real data x is rated highly. The second term corresponds to optimizing the probability that the generated data $G(z)$ is rated poorly (Here we apply the gradient to the discriminator, not the generator).

Meanwhile, in the gradient descent expression for the generator, the term corresponds to optimizing the probability that the generated data $G(z)$ is rated highly (Here we apply the gradient to the generator network, not the discriminator).

The visual representation of GANs is given on Figure 1.

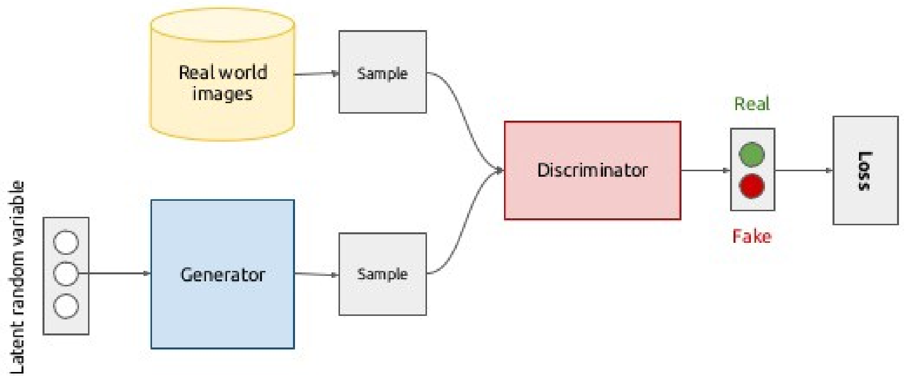


Figure 1: Generative Adversarial networks[1]

GANs has been known as yielding a sharper results compared to VAE while VAE often result in better log-likelihoods. Among the existing approaches, GANs appears to be a feasible and suitable for generating fashion since the new sample is generated in the way such that it is updated indirectly through the flow of gradients from the discriminator. Meanwhile, other methods which do not use maximum likelihood will lead to the exact recover of p_{data}. Obviously, in generative fashion problem, we do not want our model to access to p_{data}.

3.2 Deep Convolutional Generative Adversarial Networks (DCGANs)

The disadvantage that often results in nonsensial outputs using GANs is improved by feeding arithmetic vector in GANs. The architecture is proposed by Radford et al. (2015c) for a stable training procedure which appears to be attractive to representation learning. In their work, they propose the following core modifies to CNN architecture by adopting the idea of GANs:

- All pooling layers are replaced with strided convolutions (discriminator) and fractional-strided convolutions (generator), which allows the network to learn it own spatial during decode and encode stage.
- Batch normalization is applied to maintain the stability of network, the zero-mean center characteristic of features as well as to make sure the gradients flow deeper.

[1]Source: `https://hackernoon.com/how-do-gans-intuitively-work-2dda07f247a1`

- All fully connected layers are removed.
- ReLU is used for all layers of the generator, except Tanh is used for the output, and LeakyReLU is used for all layers of discriminator.

The normally-distributed variable z is initialized, then fed through several convolutional layers and activated by a tanh function. The final output of the generator will be passed to the transposed convolutions which arises from the desire to create the opposite output of a normal convolution, in another word, to create the output that has the similar form of z. At the end, there is a sigmoid activation function to classify into the real and the generated data.

The illustration of the encode stage is presented on Figure 2.

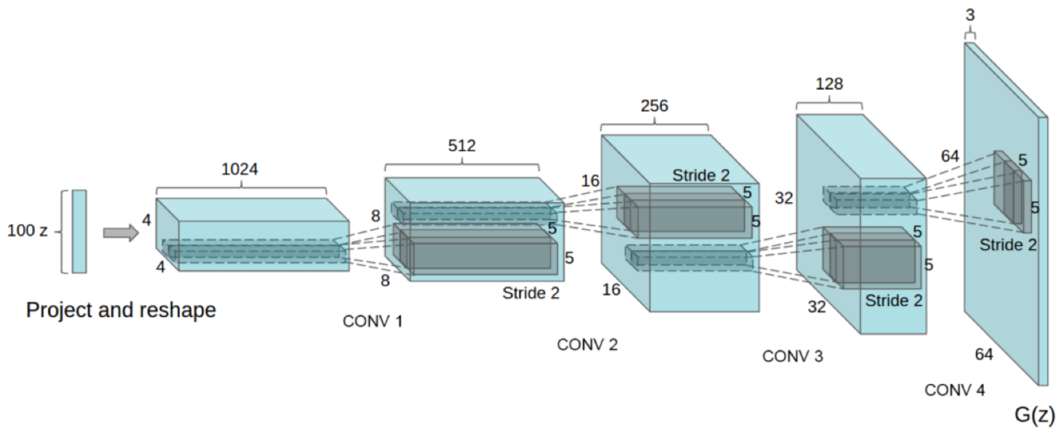


Figure 2: DCGAN generator. As can be seen, no fully connected layer is used. The architecture of the discriminator is the inverse of that of generator[3]

In our work, we are planning to adopt this idea into different dataset in the field of fashion design. As mentioned earlier, a fashion item consists of several colors as well as small detail pieces of design, which can lead to the difficulty in the process of learning features of both CNNs and GANs. Hence, in this project, we attempt to train a model that can learn a hierarchy of features.

4 Dataset

In our experiment, we would like to train and test our model using UT Zappos50K (UT-Zap50K) dataset Yu and Grauman (2014). It is a large shoe dataset made of 50,025 catalog images from **Zappos**[4]. The images fall into 4 major categories, sneakers, sandals, slippers, and boots. All the shoes are taken at the centered of a white background and pictured in the same orientation for convenience. However, before feeding these images into the training model, we need to do preprocessing steps on this dataset to make it more appropriate to our model, which is implemented by using some simple padding, cropping and convolutional layers at the beginning.

A sample of some pictures of the real data is shown in Figure 3. As can be seen, the majority of the data is sneakers, including sport and slip-on shoes, with a variety of colors from many well-known brands such as Adidas, Nike and New Balance. Because of the dominance of sneaker in the dataset, the generated results will be mainly sneakers. This will be observed in Section 5.3.

After that, if we still have enough time and our model perform stability in the available dataset, we will build our own dataset by collecting t-shirt and accessory images, which have identical background and structure with the shoes dataset, to train and test our model.

[3]Source: `https://arxiv.org/abs/1511.06434`

[4]`http://www.zappos.com/`

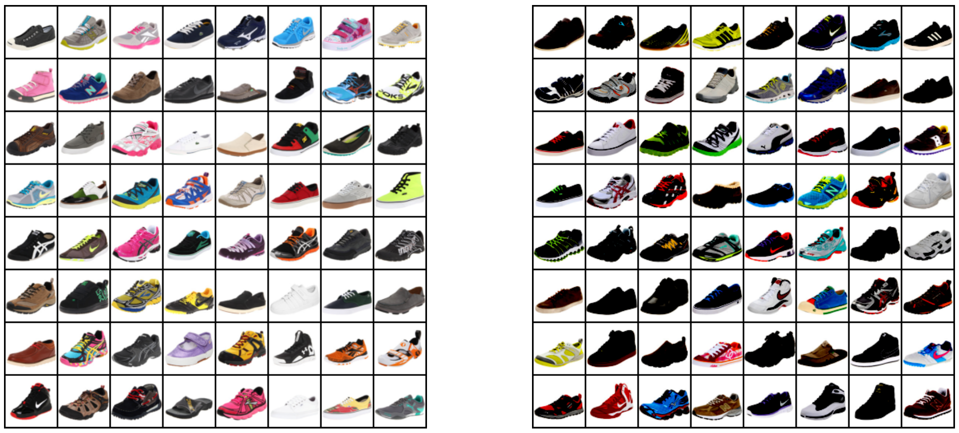


Figure 3: Samples of real data

5 Experiments

Following the acquisition of the appropriate dataset, we conducted several experiments and evaluated different approaches on it. It has been known that GANs is one of the most difficult model to train. Therefore, in this task, we also want to investigate the effects of some performance improvement techniques proposed in some technical blogs and how the fine-tuning of the hyper-parameters influences the outcome. Therefore, we will explain in detail the steps we performed as well as the results we got in the following.

The code was implemented in **Pytorch**[5]. All computations were conducted on a single machine with a NVIDIA GeForce GTX 1050 graphics card with 1.95GB of RAM. It took approximately two hours for running 100 epochs.

5.1 Methods

Our main method was described above in Section 3. Even though, DCGAN is an improvement of GANs, which is considered to be more stable in terms of training and generating higher quality samples, it is still hard to make it converge. Due to this reason, we adopted the tricks of **Soumith**[6] and **Guim Perarnau**[7], which we will briefly outline below, in order to increase the chance of having good model and better high-resolution images.

- Use a modified loss function: In GAN papers, the loss function to optimize G is $\min [log(1 - D)]$, but here we will use $\max [logD]$ during G training procedure to avoid vanishing gradients early on.
- Use a spherical noise: the noise will be sampled from a Gaussian distribution.
- Use one-sided label smoothing: make the discriminator target output from [0=fake image, 1=real image] to [0=fake image, 0.9=real image].
- Use *freezing* Salimans et al. (2016) , which means stopping the updates of D whenever its training loss is less than a certain upper threshold of the training loss of G.

We intended not to use all the tricks at a time, but used a combination of different tricks and applied one by one after each experiment.

Our proposed DCGAN model takes in the input of 64x64-pixel image. During training time, we can track the failure of model at early stage by observing these events:

- Discriminator losses approach zero: this means there is no gradient for G's optimizer.

[5]https://github.com/pytorch
[6]https://github.com/soumith/ganhacks
[7]http://guimperarnau.com/blog/2017/03/Fantastic-GANs-and-where-to-find-them

- D losses rise unbounded on generated images: similarly, this means there is no no gradient for D to update, and G stops training since the gradients it's reading suggest that it has achieved perfect performance.
- Generator losses steadily decrease: this means that what G produced can not fool D anymore.
- Divergent discriminator accuracy: D learns by either classifying everything as real or everything as generated. We can detect this by checking D's losses on generated images against D's losses on real images. In GANs proposed paper, the author evidenced that the probabilities of D classifying fake and real labels equal 0.5 when it converges.

5.2 Implementation Details

This dataset contains more than 50,000 images of sneaker. The original resolution of each image is 136x102x3. We transformed the data so that the final image dimension is 64x64x3. First, we added 24 white padding to the image border to get 136x136 image, further cropped back to 136x136 and finally resize to squared image size of 64.

One note on what we did while training is we tried with many different hyperparameters, especially learning rate to observe the performance of model. As we noticed, the generated images became worse or even nonsense images were got as we were decreasing the learning rate from 0.0002 to 0.00001. We hereby describe only the cases with useful outputs.

For the first training, the pure DCGAN architecture with the spherical noise. We trained with the learning rate of 0.0002, mini batch-size of 64 and optimized the parameters of D and G by using Adam Kingma and Ba (2014) with β_1 = 0.5 and β_2 = 0.999.

For the second training, we kept the learning rate of 0.0002, mini batch-size of 64, Adam optimizer with β_1 = 0.5 and β_2 = 0.999 intact. The input images were normalized, the weight decay of $1e-4$ was added and the loss function was modified.

Next, for the third training, we retained all hyperparameters and implemented an additional sided-label of 0.9 for real data. Finally, we used *freezing* with the threshold of 0.7.

5.3 Results

When it comes to the evaluation, the way how a generative model as well as a deep learning model is assessed has been an arguable topic. Until now, many researchers agreed that a meaningful training progress can not be merely reflected by the losses. In GANs, there would be a case that small loss on G, which is normally considered as a bad situation theoretically, results in better G in reality. For our generative model, we evaluated the model by printing out the pictures and looking at them. The model loss was used for tracking and reference.

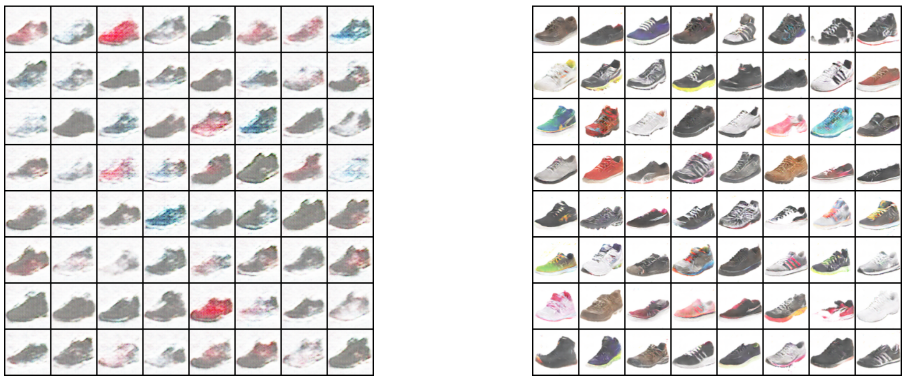


Figure 4: Visualization of generated samples of the first experiment. Left: the generated sample at the early state. Right: the generated sample at the late state.

In Figure 4, we showed samples drawn from the generator after training. From a very first iteration, the model was able to learn the general information of the objects such as shape or main color, as shown in the left of Figure 4. This is what exactly we expected before training because the dataset we used here is uniform in terms of those features. Meanwhile, the right shows that the results are quite promising. Unfortunately, the resolution of images is low.

200 In Figures 5, we show some samples drawn from the generator after the next training. Obviously, by applying some improvements, our models could obtain higher-resolution images.

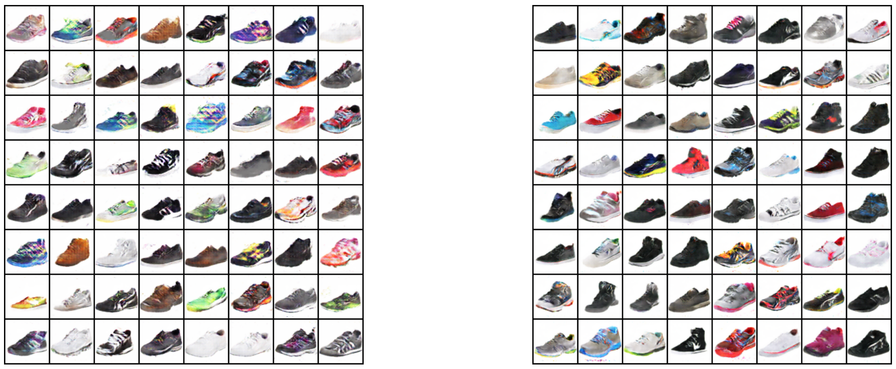


Figure 5: Visualization of generated samples of the second experiment. Left: the generated sample at the early state. Right: the generated sample at the late state.

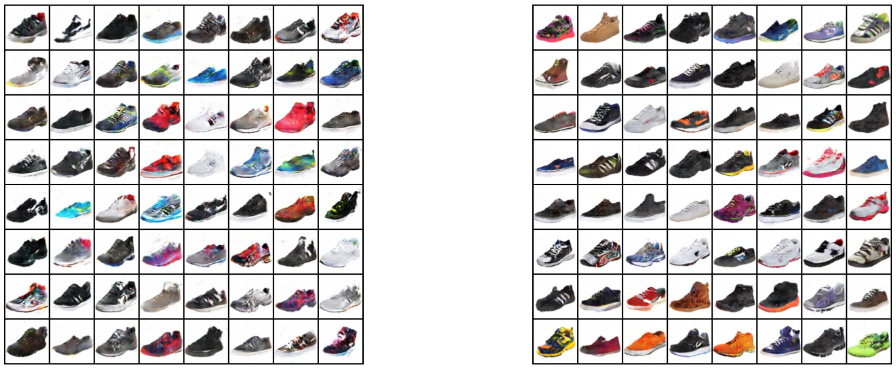


Figure 6: Generated samples of the third experiment. Left: the generated sample at the early state. Right: the generated sample at the late state.

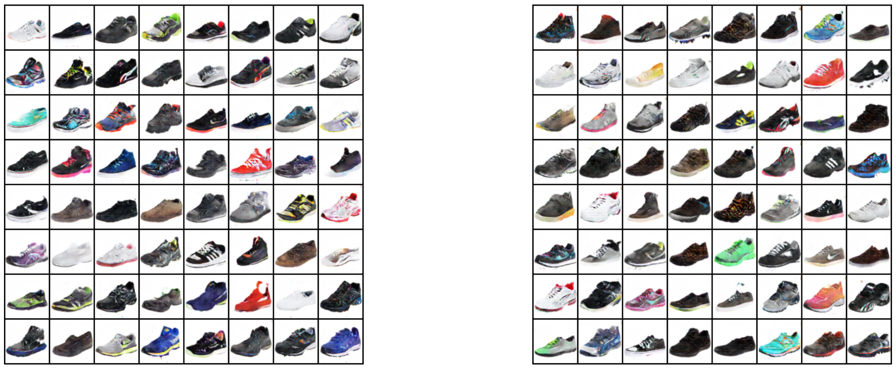
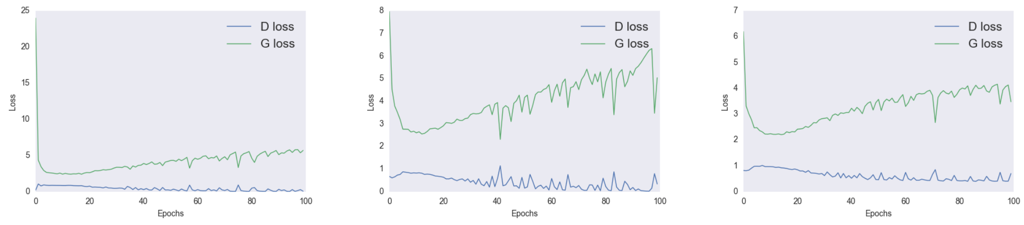


Figure 7: Generated samples of the last experiment. Left: the generated sample at the early state. Right: the generated sample at the late state.

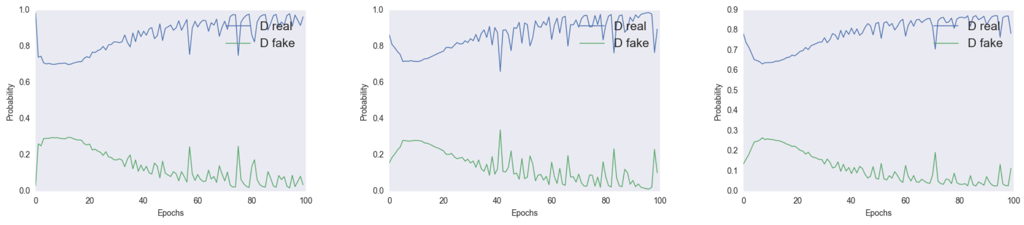
In Figures 6 and Figures 7, compared to the previous generations, the result got from these training is much better in term of the resolution and model stability over time. We cannot say which result is better in practice because we have no expertise in design. However, we can say that our model learned and generated some representations of the images.

Move to the performance of D and G, Figures 8 and 9 give the evidence that we were encountering the problem in which D became too strong compared to G in the three first experiments. It supports that G may be stuck in some local minimum while D is still learning well and is further and further outperforming G, so it cannot really learn anything and often coming back to random noise. Theoretically, both networks should be trained to become powerful together by taking in turn and the model converges when the probability should equal 0.5. Here, we must say GANs is extremely unstable and hard to train.



(a) Experiment 1 (b) Experiment 2 (c) Experiment 3

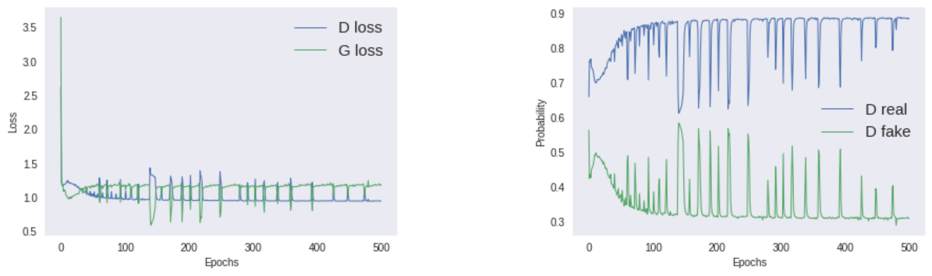
Figure 8: Model loss



(a) Experiment 1 (b) Experiment 2 (c) Experiment 3

Figure 9: Performance of Discriminator

Move to the loss of the final experiment in Figure 10, it seems that D and G were updating together, meanwhile D sometimes misclassified the labels. In summary, the model is able to tackle the problem mentioned earlier.



(a) Model loss (b) Discriminator probabilities 2

Figure 10: Model performance

6 Discussion and conclusions

As can be seen from all figures in Section 5.3, all the models were able to mimic the shape of items they needed to generate since the second epoch. For higher features, the models could learn to generate after epoch 20. With some methods proposed, the results from the generative model were improved and there were some artifacts that we can easily observe.

Even though we implemented many techniques to improve the training, we still encounter the problems of too-powerful discriminator. In order to tackle this, we applied *freezing* which is a very promising and efficient method compared to the previous approaches.

It has been known that even with the right hyper-parameters, network architecture, and training procedure, there is still a high chance that either the generator or discriminator will over-perform the other. The common case of this is when the generator has collapsed onto a single point, and therefore the output will not be improved.

From our experiment so far, we can see the case that the discriminator became too powerful and is able to easily make the distinction between real and fake images while the generator was still dumb is more likely to happen in reality. To achieve a good model, the Discriminator and Generator should learn and improve at the same time. It seems to be hard since we trained the whole model from the beginning and used no pre-trained model.

Generative Adversarial Networks is a promising class of generative models that has so far been held back by unstable training and by the lack of a proper evaluation metric. This work presents partial solutions to both of these problems. We propose several techniques to stabilize training which allow us to train models that were previously untrainable. In the future work, we hope to develop a more stable and practical model that can be applied in many different dataset, which eases the work of a designer. For more detail about the successful and failure cases, please refer to the **Appendix**.

References

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

Dahl, G. E., Sainath, T. N., and Hinton, G. E. (2013). Improving deep neural networks for LVCSR using rectified linear units and dropout. In *International Conference on Acoustics, Speech and Signal Processing, ICASSP*, pages 8609–8613.

Dosovitskiy, A., Springenberg, J. T., and Brox, T. (2015). Learning to generate chairs with convolutional neural networks. In *Conference on Computer Vision and Pattern Recognition, CVPR*, pages 1538–1546.

Efros, A. A. and Leung, T. K. (1999). Texture synthesis by non-parametric sampling. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1033–1038 vol.2.

Freeman, W. T., Jones, T. R., and Pasztor, E. C. (2002). Example-based super-resolution. *IEEE Computer Graphics and Applications*, 22(2):56–65.

Goodfellow, I. J. (2017). NIPS 2016 tutorial: Generative adversarial networks. *CoRR*, abs/1701.00160.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative Adversarial Networks. *ArXiv e-prints*.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. C., and Bengio, Y. (2014). Generative adversarial networks. *CoRR*, abs/1406.2661.

Hays, J. and Efros, A. A. (2007). Scene completion using millions of photographs. *ACM Trans. Graph.*, 26(3).

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(9):1904–1916.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., et al. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. (1999). An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233.

Karpathy, A. and Li, F. (2015). Deep visual-semantic alignments for generating image descriptions. In *CVPR*.

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

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, number 2014.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *NIPS*.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

Neal, R. M. (1993). Probabilistic inference using markov chain monte carlo methods.

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

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

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

Salimans, T., Goodfellow, I. J., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved techniques for training gans. *CoRR*, abs/1606.03498.

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. (2013). Overfeat: Integrated recognition, localization and detection using convolutional networks. *CoRR*, abs/1312.6229.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2014). Going deeper with convolutions. *CoRR*, abs/1409.4842.

van den Oord, A., Kalchbrenner, N., and Kavukcuoglu, K. (2016). Pixel recurrent neural networks. In *ICML*, pages 1747–1756.

Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2015). Show and tell: A neural image caption generator. In *CVPR*.

Wu, Z., Valentini-Botinhao, C., Watts, O., and King, S. (2015). Deep neural networks employing multi-task learning and stacked bottleneck features for speech synthesis. In *International Conference on Acoustics, Speech and Signal Processing, ICASSP*, pages 4460–4464.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A. C., Salakhutdinov, R., Zemel, R. S., and Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. In *ICML*.

Yu, A. and Grauman, K. (2014). Fine-Grained Visual Comparisons with Local Learning. In *Computer Vision and Pattern Recognition (CVPR)*.

Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *ECCV*.

Zen, H., Senior, A. W., and Schuster, M. (2013). Statistical parametric speech synthesis using deep neural networks. In *International Conference on Acoustics, Speech and Signal Processing, ICASSP*, pages 7962–7966.

Zhu, J., Krähenbühl, P., Shechtman, E., and Efros, A. A. (2016). Generative visual manipulation on the natural image manifold. *CoRR*, abs/1609.03552.

Appendix

The Figure 11 showed the cases that generative model successfully output some sensible images.

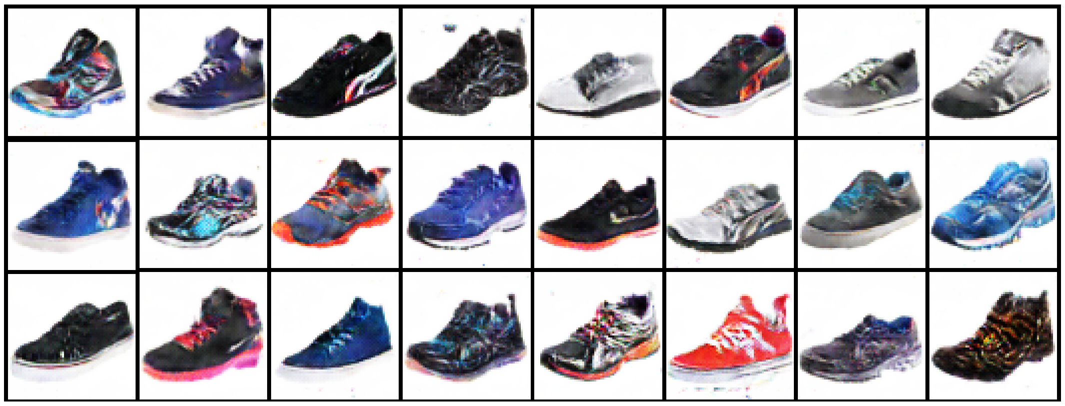


Figure 11: Successful cases

In the failure cases, the model can neither learn the shape or the color of the shoes. In some instances, the model even generated the item with the distinct shape as shown in the last picture of Figure 12.

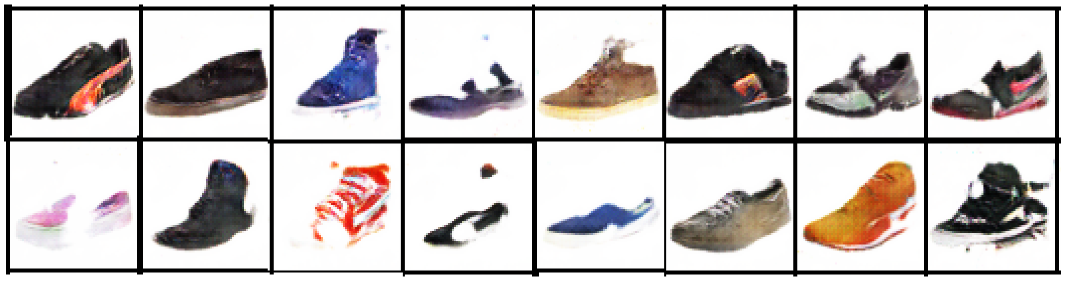


Figure 12: Failure cases