Variational Autoencoders (VAE) vs Generative Adversarial Network (GAN) vs VAE-GAN

Chirag Rana
Department of Computer Science
University of Toronto, Toronto, ON
`chirag.rana@mail.utoronto.ca`

Rutav Shah
Department of Computer Science
University of Toronto, Toronto, ON
`rutav.shah@mail.utoronto.ca`

Shree Khajuria
Department of Computer Science
University of Toronto, Toronto, ON
`shree.khajuria@mail.utoronto.ca`

Abstract

We analyze the performance, advantages, and disadvantages of Variational Autoencoder (VAE), Generative Adversarial Network (GAN), and their combined framework VAE-GAN [5] and why it works. Using the MNIST dataset we experiment with the three different frameworks and observe their losses, training times, and outputs. We also go look at ways to improve the VAE-GAN framework and propose the usage of Adam optimizer [12].

1 Introduction

GANS and VAE are very popular choices when it comes to generating images. Both these models have their pros and cons while VAE'S are easy to train as compared to GANS, GANS produce better quality images. So to get the best of both A. Larsen et al in his paper, Auto-encoding beyond pixels using a learned similarity metric, proposed the term VAE-GAN [5]. This project aims to compare these three frameworks, GAN, VAE and VAE-GAN, by making the three models' architecture as similar as possible, as well as perform experiments to examine their properties.

2 Related Works

Variational Auto-Encoder (VAE) was first introduced in 2013 by D. P. Kingma and M. Welling. Their main benefit was that it enabled the learning of smooth latent state representations of the input data. The encoder takes an input sample and converts it into a vector (a latent space representation of that data) and then the vector is passed to the decoder which reconstructs it. But due to minimizing the MSE based reconstruction error we tend to get more blurry images [4] [3].

GANS solve this issue by calculating the loss function at the end of the discriminator and this in return produces high-quality images. GANs or Generative Adversarial Networks were first introduced in 2014 by Ian Goddfellow et al [2]. Since then they achieved impressive results on image generation [14] [6]. In training real images from a dataset are given to the discriminator simultaneously with the fake images generated by the generator and the descriptor distinguishes the fake from the real. But However, GANs are very sensitive to training parameter tuning and suffer from unstable convergence and mode collapse. [7] They are also very difficult to train.

Codebase located at https://github.com/rutavshah/CSC413_Project

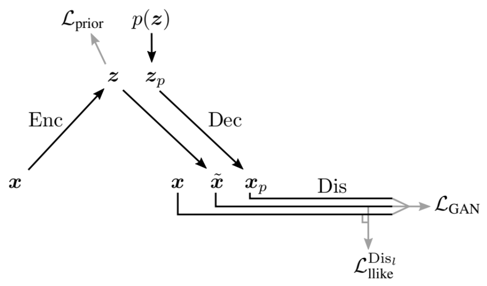


Figure 1: VAE-GAN

Keeping the pros and cons of both the models in mind A. Larsen et al in 2016 proposed VAE-GANs [5]. They proposed replacing the VAE reconstruction error term with the reconstruction error expressed in the GAN discriminator, as it is more informative.

3 Method

The method we use is to create three convolutional networks (Encoder, Decoder/Generator, Discriminator) using PyTorch [10][1]. With these three networks we can create all three frameworks of GAN, VAE, and VAE-GAN needed for comparisons. Appendix A contains the structure for the Encoder, Appendix B contains the structure for the Decoder/Generator, and the Appendix C for the Discriminator.

3.1 GAN

We will consider a traditional GAN with a Generator (which is the same as the Decoder for the VAE model) and a Discriminator [2]. We want to keep the training and architecture of the GAN as similar to the VAE and VAE-GAN as possible since our primary objective is to compare the three frameworks. We train it for 1 epoch (due to limited time and resources) with batch size of 64 with 938 training iterations. The training algorithm consisted of one update of the Discriminator, then one update for the Generator and repeat. With each update of a network, we sampled a new noise vector and every iteration we sampled a new batch of data [2].

3.2 VAE

The VAE framework used the Encoder and Decoder (which is the same as the Generator for the GAN model) networks [11]. This model is also trained with 1 epoch, batch size 64 and 938 training iterations. The VAE model uses Mean Squared Error (MSE) and the Kullback–Leibler divergence to calculate the reconstruction error [8]. We also use the reparameterization trick in the calculation of the latent variable for easier backpropagation [4].

3.3 VAE-GAN

From Figure 1 we can observe the structure of how we combine the VAE with the GAN to come up with the VAE-GAN framework. The Generator and Decoder are essentially the same network in the VAE-GAN. However, when we combine the models it is not simple as having the output of the Decoder/Generator as the input of the Discriminator, we need more signals to backpropagate through the networks to get better results hence the three inputs to the Discriminator [5]. From Algorithm 1 [5] we can observe the process of how the VAE-GAN is trained. The Discriminator is updated first and then we get outputs from the updated Discriminator to get the GAN loss. From the GAN loss we can compute a weighted loss prioritising the style and content errors based on a hyperparameter γ [5]. We can then update the Encoder with the GAN loss, thereby completing one backward pass through the entire frame.

Algorithm 1 Training the VAE-GAN model

```
1: $\theta_{Enc}, \theta_{Dec}, \theta_{Dis} \leftarrow$ Initialize network parameters
2: repeat
3:     $\mathbf{X} \leftarrow$ random mini-batch from dataset
4:     $\mathbf{Z} \leftarrow$ Enc($\mathbf{X}$)
5:     $\mathcal{L}_{prior} \leftarrow D_{KL}(q(Z|X)||p(Z))$
6:     $\tilde{X} \leftarrow$ Dec($Z$)
7:     $\mathcal{L}_{llike}^{Dis_l} \leftarrow -\mathbb{E}_{q(Z|X)}[p(Dis_l(X)|Z)]$
8:     $Z_p \leftarrow$ samples from prior $\mathcal{N}(0, I)$
9:     $X_p \leftarrow$ Dec($Z_p$)
10:    $\mathcal{L}_{GAN} \leftarrow \log(Dis(X)) + \log(1 - Dis(\tilde{X})) + \log(1 - Dis(X_p))$
11:    Update parameters according to gradients
12:    $\theta_{Enc} \overset{+}{\leftarrow} -\nabla_{\theta_{Enc}}(\mathcal{L}_{prior} + \mathcal{L}_{llike}^{Dis_l})$
13:    $\theta_{Dec} \overset{+}{\leftarrow} -\nabla_{\theta_{Dec}}(\gamma\mathcal{L}_{llike}^{Dis_l} - \mathcal{L}_{GAN})$
14:    $\theta_{Dis} \overset{+}{\leftarrow} -\nabla_{\theta_{Dis}}\mathcal{L}_{GAN}$
15: until deadline
```

Table 1: VAE-GAN Hypersensitivity Analysis - Learning Rate

Learning Rate	GAN Loss	Prior Loss	Discriminator Loss
0.00015	1.6904	0.3349	0.0930
0.0003	1.5981	0.3089	0.0841
0.0006	1.5820	0.2772	0.0986

4 Experiments

4.1 Dataset

We use traditional MNIST dataset with a batch size of 64. We resize, center crop the image. For the image translation experiment we do, we randomly translate the sample images by 10% vertically and horizontally.

4.2 Gradient Descent Optimizer

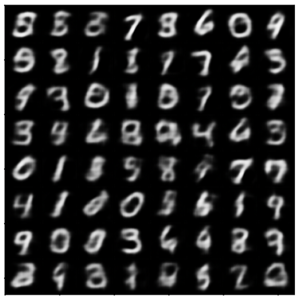
The traditional experiment performed with VAE-GANs [5] used the RMSprop gradient optimizer [1]. We decided to use the Adaptive Momemnt Optimization (Adam) algorithm for gradient descent updates. We observed that the training sped up using the all the same hyperparameters for all three frameworks (VAE, GAN, and VAE-GAN). The purpose for this experiment was to analyze how much the frameworks benefit with the addition of bias-correction and momentum [12]. We did not observe a significant difference in the losses.

4.3 Hypersensitivity Analysis

We performed a hypersensitivity analysis on the hyperparameter learning rate of the VAE-GAN framework (Table 1). The default learning rate used in the original experiment with VAE-GANs [1] was 0.0003. We checked how the losses differ when we doubled it (0.0006) and halved it (0.00015). As we can see the VAE-GAN model we trained appears to be quite stable as the three losses do not differ by too much and it seems to be stable to varying change in learning rate.

4.4 Image Translation

One of the advantages of the VAE-GANs was taking the discriminator of the GAN to learn the loss function. This compared to a VAE which uses Mean Squared Error (MSE) to calculate the reconstruction error is a significantly better approach when using a task domain like images [8].


(a) Not translated

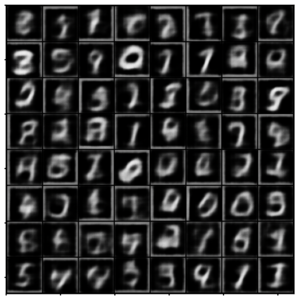
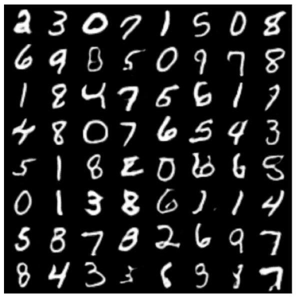

(b) Translated

Figure 2: VAE Image Translation Results


(a) Not translated

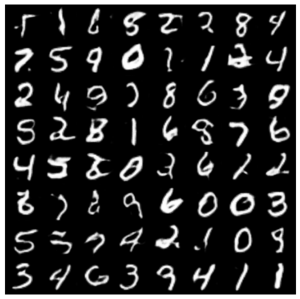

(b) Translated

Figure 3: VAE-GAN Image Translation Results

When an image gets translated, the MSE will have a large change even though the image is still the same. We wanted to verify this idea with our experiment.

The images in the MNIST dataset were randomly shifted by 10% vertically and horizontally. In Figure 2a (not translated) and Figure 2b (translated) we see clearly see the result for VAE models. As we can expect the images are blurry and the numbers are disfigured with a VAE model but the translated results are significantly more blurry and disfigured. We can also see the resultant translation with some of the images having white edges. For the VAE-GAN model, in Figure 3a (not translated) and Figure 3b (translated), we can see there is a bit of difference as some of the numbers appear to be more slanted than usual. In addition, some of the translated numbers on closer inspection appear to have disfigured parts.

Comparing the results from VAE-GAN model to the VAE model, there is a notable difference. Our expectation that the VAE-GAN model would not be as effected by the image translation compared to the VAE model is shown to be correct.

4.5 Discussion and Limitations

Training the models required a significant amount of time and computational resources. The GAN model in particular was difficult to train and unfortunately the Image Translation experiment could not include the GAN model as we did not have the resources or time to tune the hyperparameters for that model.

5 Conclusion

VAE-GANs allow us to take the best of both VAE and GAN models. We can see through the results of our experiments that the VAE-GAN model is easier to train than a GAN and more resistant to image translation compared to VAE models. We also explored how sensitive the VAE-GAN model is to changes in the learning rate and how we could expand the original VAE-GAN to use newer gradient optimization algorithms like Adam.

References

[1] A. (2016). andersbll/autoencoding_beyond_pixels. GitHub. https://github.com/andersbll/autoencoding_beyond_pixels

[2] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A.C., & Bengio, Y. (2014). Generative Adversarial Networks. ArXiv, abs/1406.2661.

[3] Kingma, D.P., & Welling, M. (2013). Auto-Encoding Variational Bayes. CoRR, abs/1312.6114.

[4] Kingma, D.P., & Welling, M. (2019). An Introduction to Variational Autoencoders. Found. Trends Mach. Learn., 12, 307-392.

[5] Larsen, A.B., Sønderby, S.K., Larochelle, H., & Winther, O. (2016). Autoencoding beyond pixels using a learned similarity metric. ArXiv, abs/1512.09300.

[6] Martin Arjovsky, Soumith Chintala, and Leon Bottou. Wasserstein generative adversarial networks. ´ In International Conference on Machine Learning, pp. 214–223, 2017.

[7] Mi, L., Shen, M., & Zhang, J. (2018). A Probe Towards Understanding GAN and VAE Models. ArXiv, abs/1812.05676.

[8] NPTEL-NOC IITM. (2020, November 11). Combining VAEs and GANs. YouTube. https://www.youtube.com/watch?v=0OgFW2W9LRY

[9] Pandey, P. (2018, June 3). Deep Generative Models - Prakash Pandey. Medium. https://medium.com/@prakashpandey9/deep-generative-models-e0f149995b7c

[10] R. (2020). rishabhd786/VAE-GAN-PYTORCH. GitHub. https://github.com/rishabhd786/VAE-GAN-PYTORCH

[11] Shafkat, I. (2018, April 05). Intuitively understanding variational autoencoders. Retrieved April 20, 2021, from https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf

[12] Shankhar, B. S. (2020, May 27). Adam Optimization Algorithm - Optimization Algorithms for Deep Neural Networks. Medium. https://medium.com/optimization-algorithms-for-deep-neural-networks/adam-optimization-algorithm-7ce202877eda

[13] Stewart, M. P. R. (2020, July 30). GANs vs. Autoencoders: Comparison of Deep Generative Models. Medium. https://towardsdatascience.com/gans-vs-autoencoders-comparison-of-deep-generative-models-985cf15936ea

[14] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. (2018). Progressive growing of gans for improved quality, stability, and variation. In International Conference on Learning Representations.

Codebase

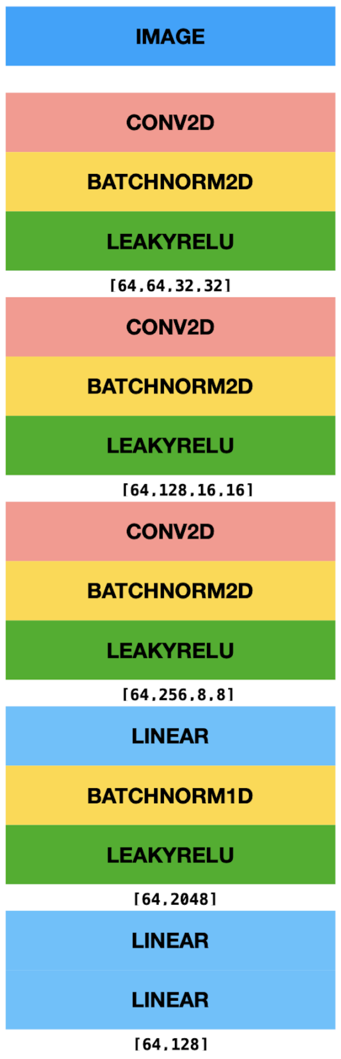
Codebase located at https://github.com/rutavshah/CSC413_Project

Contributions

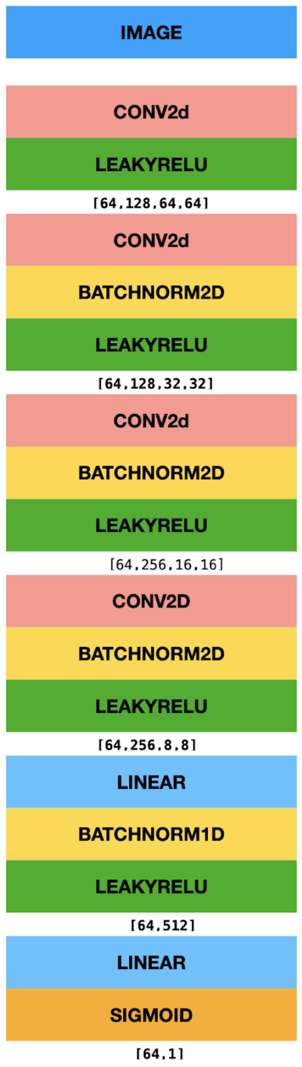
Everyone contributed equally.

Appendices

A Encoder Appendix



B Decoder Appendix



C Discriminator Appendix

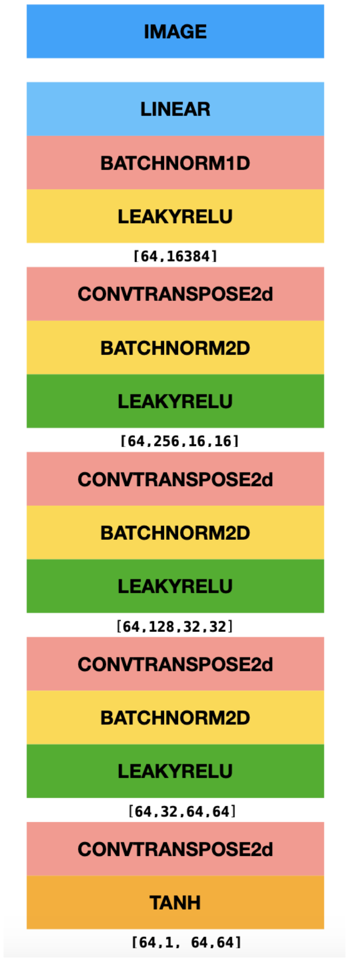


IMAGE
LINEAR
BATCHNORM1D
LEAKYRELU
[64,16384]
CONVTRANSPOSE2d
BATCHNORM2D
LEAKYRELU
[64,256,16,16]
CONVTRANSPOSE2d
BATCHNORM2D
LEAKYRELU
[64,128,32,32]
CONVTRANSPOSE2d
BATCHNORM2D
LEAKYRELU
[64,32,64,64]
CONVTRANSPOSE2d
TANH
[64,1, 64,64]