Intro in the Generative Adversarial Networks

Seminar

Image Based Biometrics 2020/21, Faculty of Computer and Information Science, University of Ljubljana

Matjaž Mav (63130148)

***Abstract*—In this short paper we review basics of the Generative Adversarial Networks (GANs) along with the a few early GAN implementations. Later we implement those models and practically evaluate them. Additionally, we explore how GANs can be used to generate photo-realistic images from a paint brush tampered images.**

I. Introduction

Generative models are interesting area in the computer vision domain and have various use cases, for example data augmentation, style transfer, data manipulation, anomaly detection, adversarial training and many more.

Essentially there are two bigger sub domains of generative models, Autoencoders (AE) and Generative Adversarial Networks (GANs) [1]. Both these models consist of two parts. AEs consist of the encoder that maps a high dimensional space input into a low dimensional latent space [2] and the decoder that maps a low dimensional latent space back to the original space. Each GAN consists of a two separate models, the generator and the discriminator; and both are competing with each other. The generator tries to generate real like data from the random noise vector and the discriminator tries to discriminate between real and fake (generated samples).

The rest of this paper organised as follows. In the rest of this section we will present GANs in more detail and the related work. In the second section we will review the implementation of three different GAN models and discuss their advantages and disadvantages. In the third section we will present learning result of the models form the second section. Additionally, we will present our experiments with image tampering with the state of the art GAN model.

A. Training Generative Adversarial Networks

As we drafted before, GAN training is composed of a two models, the generator and the discriminator, this is illustrated on the Figure 1. The generator maps the noise vector (sometimes also called latent vector or Z vector) to the generated image (fake) and the discriminator tries to discriminate between real and fake images.

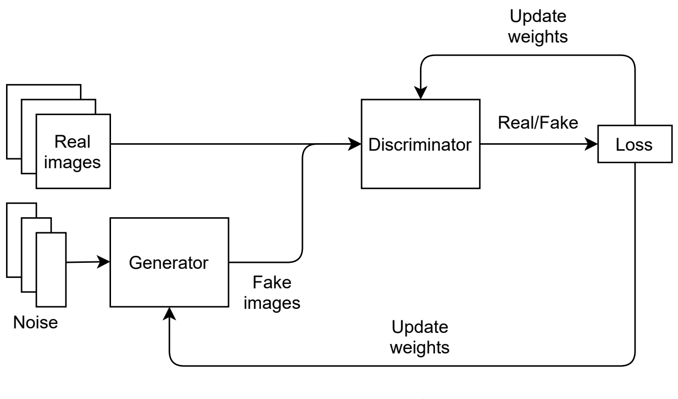


Figure 1: Basic architecture of GAN model training

Essentially these two models compete between each other, the generator tries to trick the discriminator, this is also called minimax problem.

Binary Cross Entropy (BCE) Loss is common loss function used in training GANs, because it is designed for two class problems (for example real vs fake). BCE loss is defined on interval between zero and one. For a positive outcome it returns value close to zero and in case of a negative outcome it returns value close to infinity. BCE loss is defined with Equation 1 where G stands for the generator, D for the discriminator, x sample of real images and z sample of noise vectors.

$$\min_G \max_D -[\mathbb{E}(log(D(x))) + \mathbb{E}(1 - log(D(G(z))))] \qquad (1)$$

Error back propagation is usually done in cycles, first on the discriminator and then on the generator.

B. Latent Space Manipulation

Latent space or Z space [2] is a vector space from which we sample random noise and pass it to the generator. Ideally we can find directions in the latent space to manipulate the generator into producing images whit certain features, this is also called controllable generation.

One of the common problems is so called latent space entanglement, which means that change in one direction would change two or more features. Additionally these features can also be correlated.

Another way of manipulating the generator into producing images with certain features is method called conditional generation. This approach takes a part in the GAN training stage, where generator receives as input a random noise and also one hot encoded class or embedding. This additional information is then used by the discriminator to guide the generator into learning how to generate images of certain class.

The simplest latent space manipulation is visualized on the Figure 2.

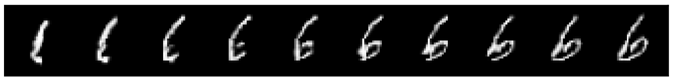


Figure 2: Linear interpolation between two random latent vectors, digit one and digit six generated with DCGAN.

C. Model Evaluation

Evaluating GANs is challenging because it is hard for a computer to measure how real the generate image is to the human eye. Good evaluation metric should encapsulate how real image looks like (fidelity) and how diverse are the generated images (diversity).

Frechet Inception Distance (FID) [3] is a most commonly used measure in the research literature. FID score utilises Inception V3 [4] model to obtain the embeddings for real and generated images. Then it compares the two distributions with

Multivariate Frechet Distance (Equation 2). Smaller FID score promise better results, however, there is no clear threshold. Also, FID score tends to get smaller for a larger samples. Additionally, Inception V3 is trained on natural looking photos and for example it is not appropriate to use on datasets like MNIST [5].

$$\|\mu_X + \mu_Y\|^2 + Tr(\Sigma_X + \Sigma_Y - 2\sqrt{\Sigma_X \Sigma_Y}) \quad (2)$$

Another common measures reported in papers are Inception Score (IS) [6], Perceptual Path Length (PPL) [7], Precision and Recall [8] and also human evaluation (HYPE [9]).

II. Methodology

In this section we will review tree different GAN models and discuss theirs advantages and disadvantages.

A. DCGAN

Radford et al. [10] proposed Deep Convolutional GAN (DCGAN) model that achieves stable training. They also visualized learned convolutional filters and shown how specific filters draw specific objects. Additionally they explored latent space and shown that generator have interesting vector arithmetic properties allowing easy manipulation of semantic qualities of generated images.

Architecture of the DCGAN model adopts few interesting ideas: (1) Only convolutional layers are used, there is no dense layers. (2) Pooling functions are replaced with strided convolutions. (3) Batch normalization is used on all layers (except the output layer of the generator and the input layer of the discriminator). Batch normalization is essential for deeper neural networks. (4) Generator uses ReLU activation for all layers, output layer uses Tanh activation. Discriminator uses LeakyReLU ($slope = 0.2$) activation for all layers. (5) Learning is done in mini-batches, using Adam optimizer ($\beta_1 = 0.5$).

B. WGAN-GP

Gulrajani et al. [11] addressed problem of training instability. They propose improvement over Wasserstein GAN (WGAN) [12] that uses weight clipping to enforce a 1-Lipschitz continuity (Equation 3) of the critic.

$$\|\nabla f(x)\| <= 1 \quad (3)$$

In WGAN derived models, discriminator is replaced with the critic. Discriminator is essentially a classifier that return probability that given image is real or fake. Thus, if the discriminator converges and always predict that given image is fake, generator can not learn further. On the other hand critic (C) can return any real value as long the norm of its gradient is at most one (Equation 2) and this allow the generator to learn better.

Instead of weight clipping authors of the WGAN-GP propose soft method called Gradient Penalty (GP) to enforce 1-Lipschitz continuity. The final loss function is composition of the Earth-Movers distance and with regularization term (Equation 4), where λ is hyper parameter and $\hat{x}$ is a random interpolation between real and fake image.

$$\min_G \max_C \mathbb{E}[C(x)] - \mathbb{E}[C(G(z))] + \lambda \mathbb{E}[(\|\nabla C(\hat{x})\| - 1)^2] \quad (4)$$

They demonstrated greater learning stability across a variety of architectures (including DCGAN).

C. Simple GAN *

This GAN model is our simplification of the DCGAN, using only dense layers and batch normalization. The generator uses ReLU activation in the hidden layers and Sigmoid in the output layer; the discriminator uses LeakyReLU. We decided for this architecture so we can compare dense and convolutional models.

III. Results

A. Model Evaluation

In this section we present our training results of previously defined GAN models (DCGAN, WGAN-GP and Simple GAN*) and our findings.

In the Figure 3 we can observe how learning progresses between different epochs. We trained all three models on 50 epochs. Unfortunately, results are not as good as we expected, but we can clearly observe some useful properties. Simple GAN and DCGAN are more likely to generate 1/7 or 6/8/9, this is also called mode collapse. On the other hand, WGAN-GP seems to produce more diverse results.

In the Figure 4 we visualized training loss for the generator and the discriminator/critic. This values are not directly related to quality of produced images, but are still useful to see the dynamic between two parties. We can also observe that one models tens to converge quicker the others.

The WGAN-GP model uses different loss function and can not be directly compared to two others. Additionally we see that critic overfits faster then the generator, causing the loss to increase gradually over time. Authors of the WGAN-GP shown that validation loss keeps dropping although training loss is increasing.

B. Photo-realistic Image Tampering *

Here we explore how the current state of the art GAN, named StyleGAN2 [7], [13], [14] can be used to generate photo-realistic images from a paint brush tampered images. We used official implementation provide by the authors with pre-trained model.

Our experiment was performed on three original images. First we used paint brush and childishly tampered images. Later we performed StyleGAN2 projection and observed intermediate results. Our experiments are visualized on Figure 5 and more detailed projections are visualized on Figure 6. We observed that projection overfits to the given image in this use case is best to use some intermediate projection.

IV. Conclusion

In our short paper, we review basic ideas behind GANs. We revisited theory behind the two early GAN models (DCGAN and WGAN-GP) and provided our implementation and evaluation results. Additionally, we reviewed StyleGAN2 model and explored how projection can be used to generate photo-realistic facial images from tampered originals. In future work we should explore different settings to obtain better results and evaluate our implementations with measure similar to FID or IS, to do so we will need good embedding model for MNIST dataset.

References

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, pp. 2672–2680, 2014.

[2] B. Everett, *An introduction to latent variable models.* Springer Science & Business Media, 2013.

[3] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," in *Advances in neural information processing systems*, 2017, pp. 6626–6637.

[4] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[5] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[6] S. Barratt and R. Sharma, "A note on the inception score," *arXiv preprint arXiv:1801.01973*, 2018.

[7] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019, pp. 4401–4410.

[8] M. S. Sajjadi, O. Bachem, M. Lucic, O. Bousquet, and S. Gelly, "Assessing generative models via precision and recall," in *Advances in Neural Information Processing Systems*, 2018, pp. 5228–5237.

[9] S. Zhou, M. Gordon, R. Krishna, A. Narcomey, L. F. Fei-Fei, and M. Bernstein, "Hype: A benchmark for human eye perceptual evaluation of generative models," in *Advances in Neural Information Processing Systems*, 2019, pp. 3449–3461.

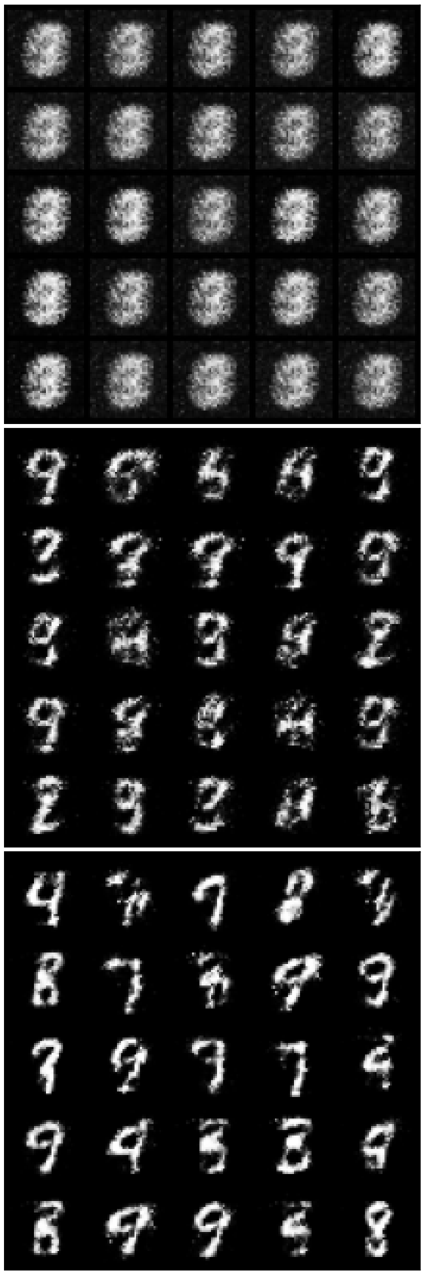
[10] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

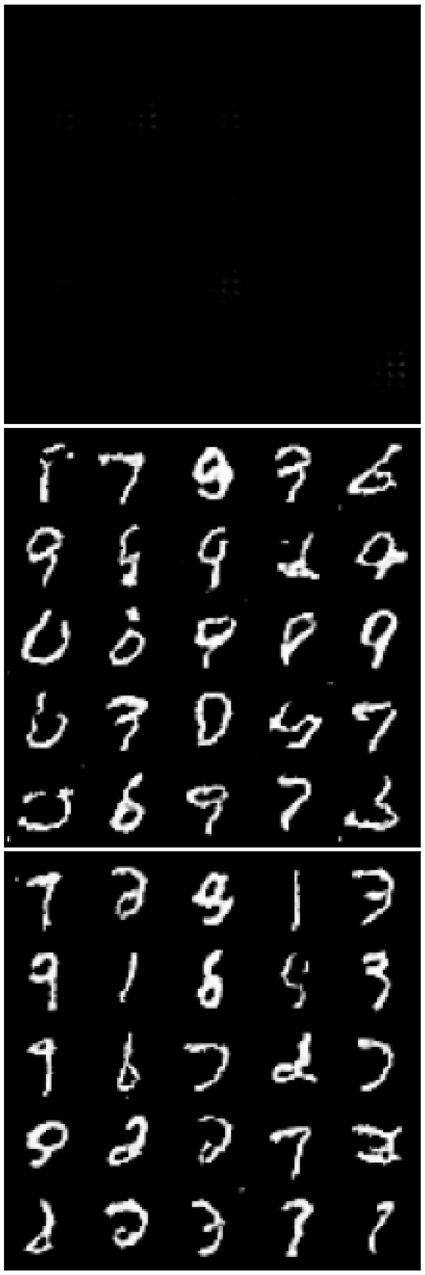
[11] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *Advances in neural information processing systems*, 2017, pp. 5767–5777.

[12] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017.

[13] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of stylegan," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 8110–8119.

[14] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila, "Training generative adversarial networks with limited data," *arXiv preprint arXiv:2006.06676*, 2020.


(a) Simple GAN


(b) DCGAN

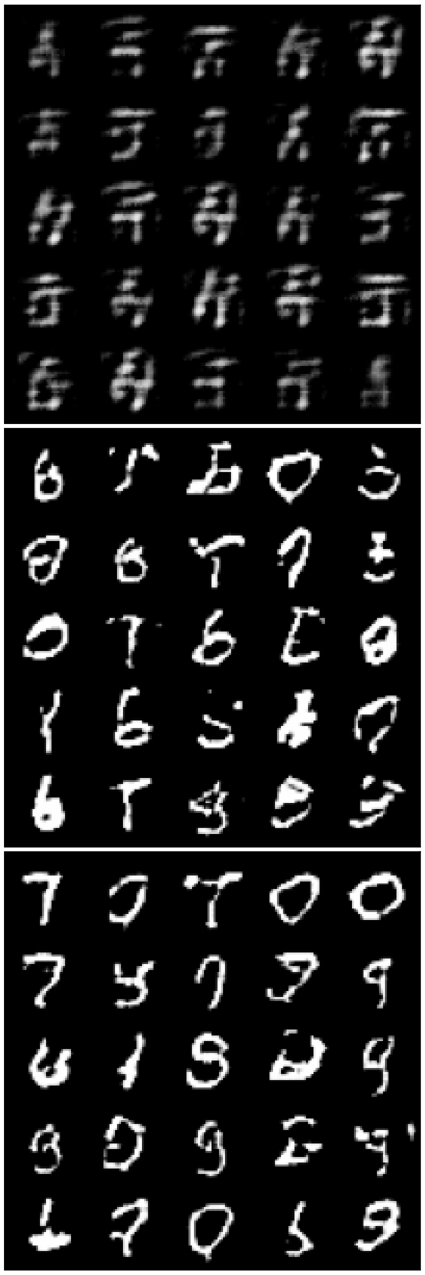
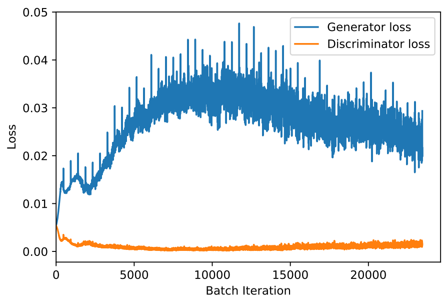
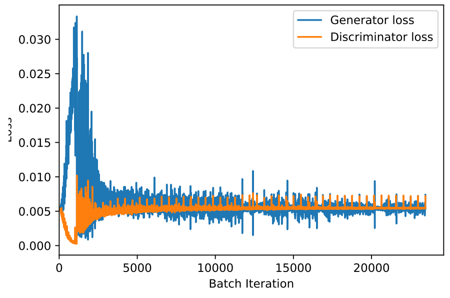

(c) WGAN-GP

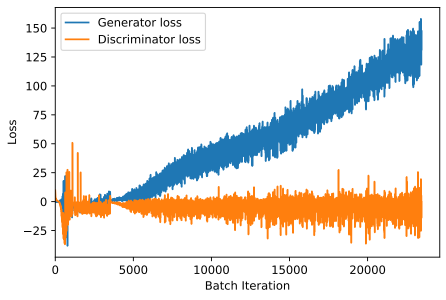
Figure 3: Example of a few randomly generated digits at epoch 1, 25 and 50 (top to bottom).



(a) Simple GAN



(b) DCGAN



(c) WGAN-GP

Figure 4: Comparison between generator and discriminator loss between training.

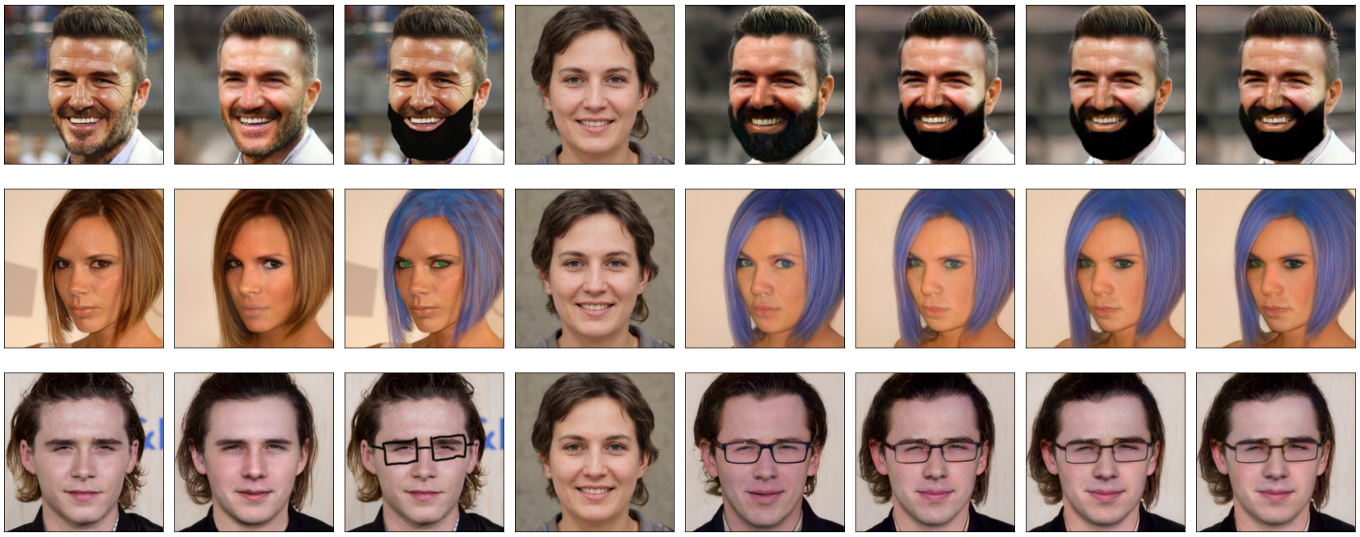


Figure 5: Example of StyleGAN2 projections of the Beckham family and its tampered pairs. Columns from left to right represents: original image, projection of the original image, tampered original, random initial projection (step 0), tempered projection (step 200), tempered projection (step 400), tempered projection (step 600), tempered projection (step 800) and tempered projection (step 1000). Zoom in for more details.

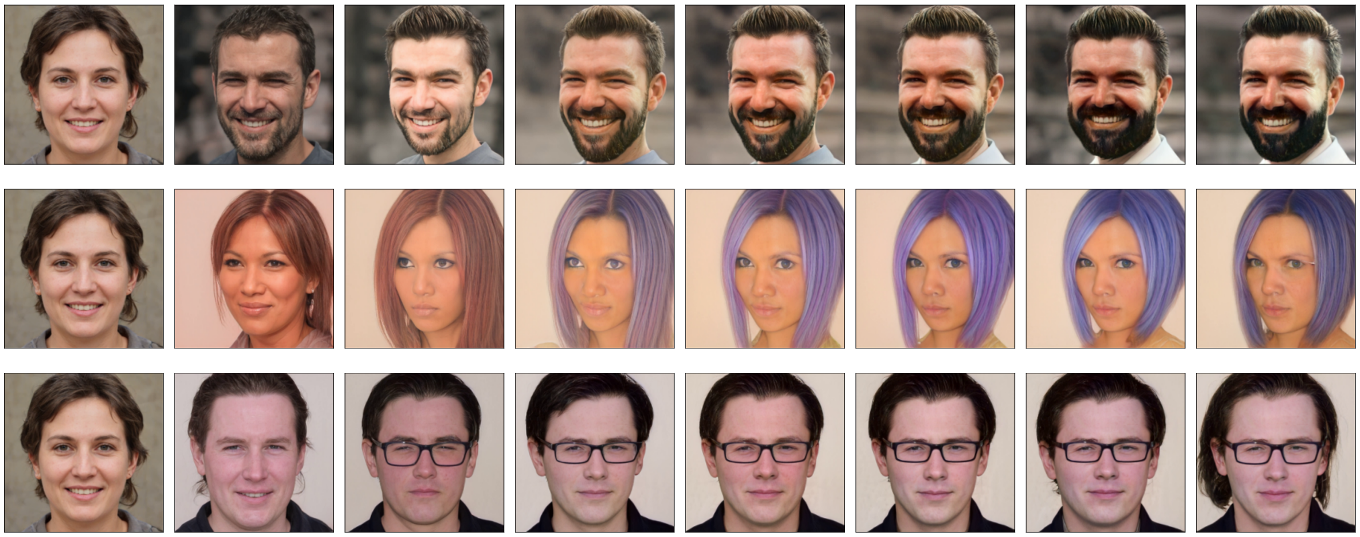


Figure 6: Addition to the Figure 5. Detailed projection path between the random initial projection at step 0 and tempered projection at step 200. Zoom in for more details.