Deep Learning Project 4: Generative Adversarial Networks

Elliot Greenlee

December 12, 2017

1 Generative Adversarial Networks

Generative Adversarial Networks are an unsupervised technique for image generation, utilizing two competing networks. A generative model G is trained to capture some data distribution, and a discriminative model D is trained to estimate the probability that a sample came from the training data or G. The networks are trained simultaneously and iteratively in competition [3]. This formulation offers a large amount of flexibility in the choice of objective function, but is unstable due to the careful balance between generator and discriminator [2].

Instability arises most often in the case of mode missing, where a mode of the probability distribution is not present, or in the case of vanishing gradients, where the apparent error for gradient descent trends to zero too early. The KL divergence objective function punishes unrealistic samples where the probability distribution has an extra peak more than samples with a mode missing, which leads to this problem. As the discriminator and generator are trained, typically the discriminator improves too quickly unless controlled, and as it learns to be a perfect discriminator the calculated error change for backpropagation becomes zero, causing the vanishing gradients instability [3].

GAN is not robust to architectural changes, and many generators produce nonsensical outputs. Deep Convolutional GAN solves this with specific constraints to the architectural topolgy of GAN. Replace pooling layers with strides so that the network can learn its own downsampling. Use batch normalization to stabilize by normalizing the inputs to have zero mean and unit variance, which helps with gradient flow and bad initializations. Remove the fully connected hidden layers, and use leaky relu all to prevent vanishing gradients. Figure 1 shows the structure of DCGAN However, this solutions still has problems with mode collapsing when training for a long time [6].

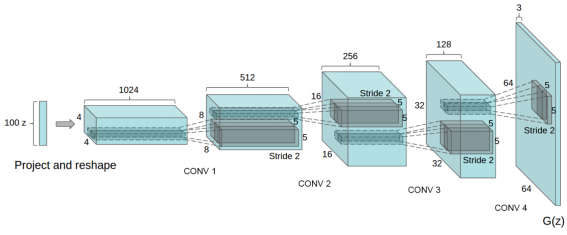

Figure 1: The structure of DCGAN.

In traditional GAN the generator is fed noise in order to generate new images, and there is no way to control the modes of the data being generated. It becomes challenging to scale such models to accommodate an extremely large number of predicted output categories. In Conditional GAN, an additional piece of information such as a class label or data from another modality is fed into the generator and the discriminator in order to produce a certain type of image [5]. This structure can be seen in figure 2.

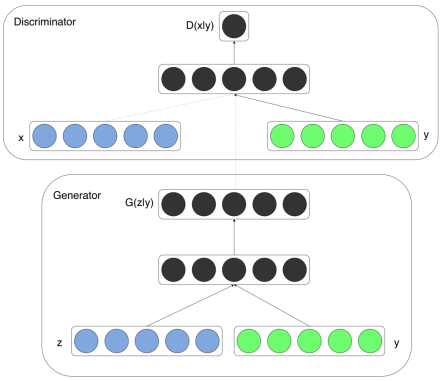

Figure 2: The simplified structure of CGAN.

2 Results

The code for this project was implemented in Tensorflow [1] and is available at https://github.com/carpedm20/DCGAN-tensorflow. Both datasets were run on a GeForce GTX 1070 GPU. The MNIST dataset consists of handwritten digits from 0 to 9 [4]. The CelebA datasets consists of celebrity faces [7]. The network used 25 epochs, a learning rate of 0.0002, 0.5 momentum for the Adam training, and a batch size of 64.

For the MNIST dataset, the input and output heights were 28 pixels. After 25 epochs with 1093 iterations per epoch and a total time of 22 minutes, the network reached a discriminative loss of 1.32 and a generative loss of 0.67. Figure 3 shows generated images at various epochs while training.

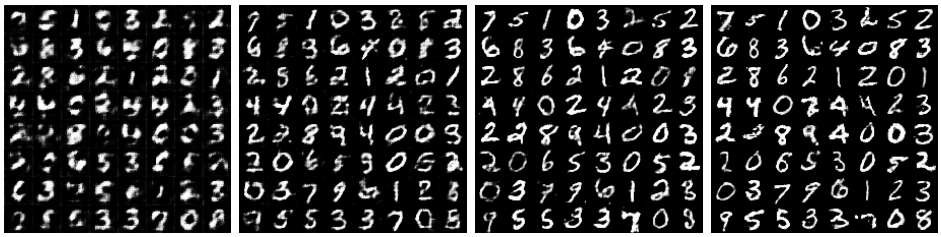

Figure 3: From left to right, generated digits at epochs 0, 1, 7, and 24.

For the celebA dataset, the input height was 108 pixels. After 25 epochs with 3165 iterations per epoch and a total time of 9.5 hours, the network reached a discriminative loss of 0.56 and a generative loss of 0.4. Figure 4 shows generated images a various epochs while training.

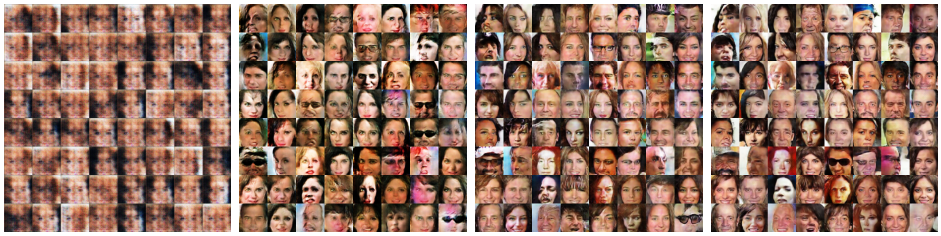

Figure 4: From left to right, generated faces at epochs 0, 1, 7, and 24.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein GAN. *ArXiv e-prints*, January 2017.

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

[4] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[5] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014.

[6] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2015.

[7] Shuo Yang, Ping Luo, Chen Change Loy, and Xiaoou Tang. From facial parts responses to face detection: A deep learning approach. *CoRR*, abs/1509.06451, 2015.