DLLAB WEEK 12 Pre-Report : Generative Adversarial Nets

Hyunsoo Cha[1], Kwanghoon Sohn[2]
[1]Yonsei University. [2]Yonsei University.

1 Generative Adversarial Nets

1.1 Introduction

Deep generative models are subjects that have been studied but have not been well used due to various difficulties. The adversarial nets framework is a proposed framework, divided into discretionary models and generative models[1]. The discretionary model is a learning model that determines. Generative models, similar to counterfeitors' teams, are key to producing fake currencies and preventing detection. The discretionary model makes imitations that are indistinguishable as if they were real. These frameworks are optimization algorithms and training algorithms used in many types of models. Researchers study the generative model for generating samples through random noise via multilayer perceptron, and the discretionary model is also implemented using multilayer perceptron. Researchers explore the special cases of adversarial nets and demonstrate training using high successful backpropagation and dropout algorithms. There is no need for inference or Markov chain.

1.2 Adversarial Nets

The adversarial modeling framework is the best model for applying multilayer perceptron. The mapping data space can then be expressed as $G(z;\theta_g)$. G is represented as a differentiable function by multilayer perceptron with parameter θ_g. The researchers output a single scalar with a second multilayer perceptron $D(x;\theta_d)$. $D(x)$ means probability. The researchers train to maximize the probability allocated as a correct label[7] for samples and training examples from G. In other words, value function $V(G,D)$ allows two players to play mini max games[3]. This is expressed in a formula as follows:

$$\min_G \max_D V(D.G) =_{x\ p_{\text{data}}(x)} [\log D(x)] +_{z\ p_z(z)} [\log(1-D(G(z)))] \tag{1}$$

The theoretical analysis of the Adversarial net can be done as follows. Training criteria are *GandD*, which allow data generating distributions to be restored to sufficient space. This means a non-parametric limit. Optimizing D can result in overfitting with computationally significant and finite datasets. Therefore, the researchers proposed a method to optimize G and optimize D for k steps. The above formula provides sufficient gradients to optimize G. The researchers trained G and optimized it in a way that maximizes $\log D(G(z))$.

1.3 Theoretical Results

Suppose the distribution of $G(z)$ samples when $z \sim p_{(z)}$ is a probability distribution with p_g. Algorithm 1 is a good estimator which $p_{\mathbf{data}}$ can be assumed. Given sufficient capacity and training time, the results can be represented by studying the coverage of infinity. Given that $p_g = p_{\mathbf{data}}$ has a global optimum, the desired results were obtained.

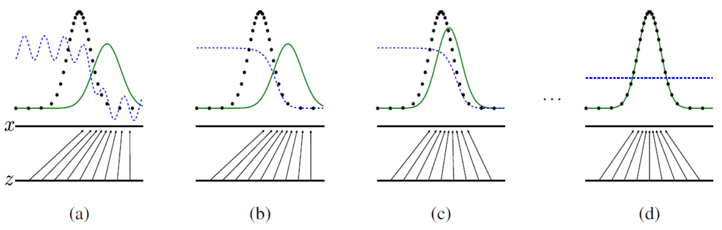

Figure 1: The figure above shows a diagram for the GAN updating the blue dashed line, discriminative distribution D.

They show how generative distribution p_g, (G) is created through (a) through (d). They show the process of being sampled as x domain for z domain. Thus, the mapping process is $x = G(z)$.

1. D is a partially accurate classifier and p_g is similar to $p_{\mathbf{data}}$. Researchers suppose that they have an adversarial pair similar to near convergence.
2. In the inner loop of Algorithm D, they train a discriminate sample from the data. They can see that it converges to $D^*((x)) = \frac{p_{\mathbf{data}(x)}}{p_{\mathbf{data}(x)}+p_g(x)}$.
3. After updating to G, the slope of D becomes $G(z)$ in the flow to the region for classifying the data.
4. After a few steps of training, when G,D have sufficient capacity, $p_g = p_{\mathbf{data}}$ makes it difficult to find a better spot. The discriminator cannot be divided by $D(x) = \frac{1}{2}$.

Algorithm 1 It stands for minibatch in the stochastic gradient description of GAN. It is designed as a dual for loop. First of all, the inner for loop for training treatment consists of k steps. It consists of sample minibatch of noise sample m which is $\{z^{(1)},\cdots,z^{(m)}\}$ from noise prior $p_g(z)$. Sample mini-batch from data generating distribution $p_{\mathbf{data}}(x)$ is $\{x^{(1)},\cdots,x^{(m)}\}$. The discriminator can be raised to the following stochastic gradient.

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

The generator should be updated by descenting the stochastic gradient. It can be represented by the following formula:

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

Gradient-based learning rules are standardized on gradient-based updates.

Global Optimality of $p_g = p_{\mathbf{data}}$ An optimality discriminator is represented as D and can be represented when G is fixed as follows.

$$D^*_G(x) = \frac{p_{\mathbf{data}}(x)}{p_{\mathbf{data}}(x)+p_g(x)} \tag{4}$$

Theorem 1 According to $p_g = p_{\mathbf{data}}$, virtual training criterion $C(G)$ can be global minimum. In this part, $C(G)$ is $-\log 4$.

Proposition 2 Assume that G,D has a lot of capacity, then discriminator can be reach the given G. Also, p_g can be updated to the direction of improvement on criterion.

$$E_{x\sim p_{data}}[\log D^*_G(x)] + E_{x\sim p_g}[\log(1-D^*_G(x))] \tag{5}$$

At this point, p_g can be p_{data} at the end on convergence.

1.4 Experiments

Researchers trained adversarial nets using CIFAR-10, MNIST[4], and Toronto Face Database[5]. Dropout was added to the generator and input was selected as the bottomomost layer of the generator network. The researchers calculated the probability of test set data at p_g.

1.5 Conclusion

Learned approximate inferences can be created by an auxillary network that assumes z for a given x. For all cases where $p(x_S|x_\$)$, a subset of all index of x is called S. Semi-supervised learning is performed by the GAN. Finally, there was an effective calculation improvement.

Model	MNIST	TFD
DBN [3]	138 ± 2	1909 ± 66
Stacked CAE [3]	121 ± 1.6	$\mathbf{2110 \pm 50}$
Deep GSN [5]	214 ± 1.1	1890 ± 29
Adversarial nets	$\mathbf{225 \pm 2}$	$\mathbf{2057 \pm 26}$

Figure 2: Researchers have shown above the log-likelihood measurement table based on Parzen's Windows. MNIST shows the average of log-likelihood in the test set. In TFD, the researchers calculated the standard error as the across fold of the dataset.

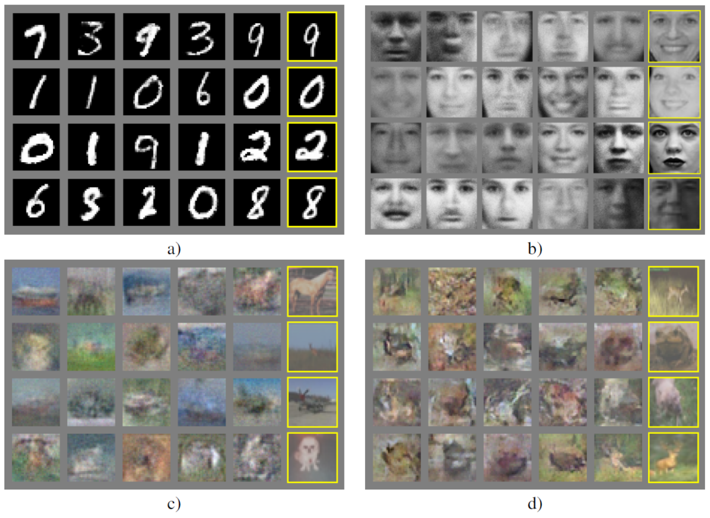

Figure 3: Through the above pictures, samples from the model were visualized. The rightmost column shows an example of training in the neighboring sample. These samples are uncorrelated because they are not dependent on Markov chain mixing. a) is MNIST. b) is TFD. c) and d) are CIFAR-10.

2 DCGANs

2.1 Introduction

Because there are no labels, these datasets can be used a lot and variously for problems such as image classification. Researchers propose a way to generate good images through GAN. They also use constructors and discriminators for supervised learning problems to extract features. GAN replaces the maximum likelihood method. This learning process and the lack of heuristic cost functions can be said to be more suitable for expression training. The researchers implemented and utilized DCGAN with the following facts.

1. Set constraints for Convolution GAN architecture. They also make learning stable through these structures and proposed conditions. They express these structures as Deep Convolutional GANs (DCGANs).
2. Using training discriminators, they use the problem of image classification. It shows some advantages over other unsupervised learning.
3. Researchers visualize learning fields by GAN. Specific filters are shown about learning specific objects.
4. Show vector technology to easily control Semantic Quality.

2.2 Details of Adversarial Training

Guidelines for designing DCGAN reliably are as follows.

1. Replace Pooling layers with strided convolution. It replaces the constructor with a fractional-strided convolution.
2. Use Batch Normalization for constructors and discriminators.
3. Remove the hidden layer of the fully connected layer through deep architecture.
4. Use activation function as ReLU and Tanh activation function only for output layer.

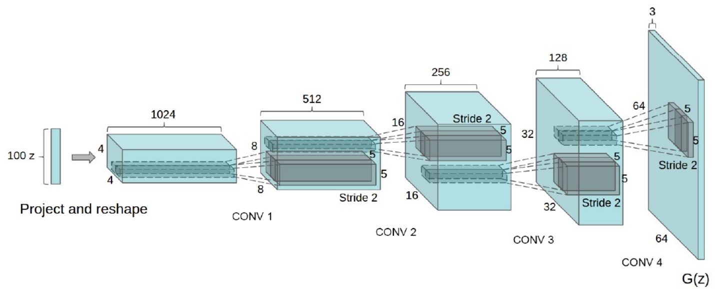

Figure 4: The figure above uses the DCGAN generator for LSUN[6] scene modeling. First, the uniform distribution Z in 100 dimensions is project and reshape. Finally, they change to a 64×64 pixel image. In other words, no fully connected layer or pooling layer was used in the above structures.

5. Use LeakyReLU on all layers of the discriminator.

Images for learning were not preprocessed separately. The Tanh activity function also sets the interval from -1 to 1. stochastic gradient descent method is trained at 128 minibatch size. All of weights are initiated at 0.02. LeakyReLU is set to slope in 0.2. Hyperparameter uses Adam optimizer and the learning rate is 0.0002.

LSUN Concerns about overfitting and memorization of training samples have emerged from researchers. To prove that much of these data scale and generate high resolution, they learned LSUN bedroom dataset model with 3 million learning samples. Researchers found that a sample from a single epoch tended to mimic time training.

Deduplication An image duplication remove operation is carried out to reduce the likelihood that the constructor will memorize input samples.

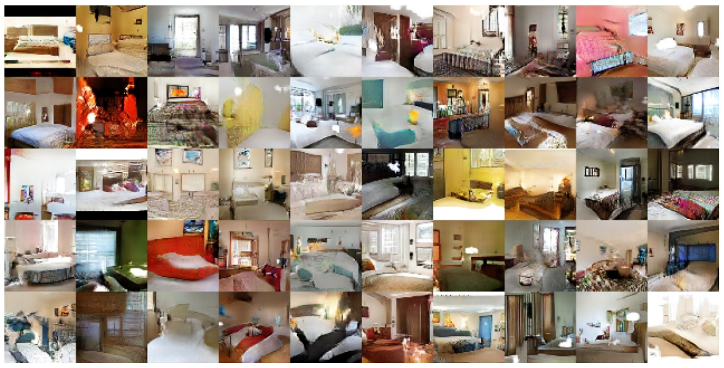

Figure 5: The figure above shows a bedroom image after 5 epoch training. They can see that noise texture is repeatedly visual under-fitting on the baseboard of some beds.

Empirical At CIFAR-10 dataset, most powerful baseline task is K-means feature training algorithm through feature extraction pipeline. Large feature maps, they show an accuracy of 80.6%. Since DCGAN has not been trained in CIFAR-10, strength can be demonstrated through various experiments.

Model	Accuracy	Accuracy (400 per class)	max # of features units
1 Layer K-means	80.6%	63.7% (±0.7%)	4800
3 Layer K-means Learned RF	82.0%	70.7% (±0.7%)	3200
View Invariant K-means	81.9%	72.6% (±0.7%)	6400
Exemplar CNN	84.3%	77.4% (±0.2%)	1024
DCGAN (ours) + L2-SVM	82.8%	73.8% (±0.4%)	512

Figure 6: The table above shows the results of the CIFAR-10 classification through a pretrained-model.

2.3 Conclusion

Researchers have found that GAN trains the image expression in generative modeling and supervision when compared to other models. They also propose a stable architecture for learning GANs[2]. The instability of the model is when the filter's subset collapses due to occilling mode.

3 Reference

[1] Yoshua Bengio. *Learning deep architectures for AI*. Now Publishers Inc, 2009.

[2] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

[3] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[4] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[5] Joshua Susskind, Adam Anderson, and Geoffrey E Hinton. The toronto face dataset. Technical report, Technical Report UTML TR 2010-001, U. Toronto, 2010.

[6] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

[7] Junbo Zhao, Michael Mathieu, Ross Goroshin, and Yann Lecun. Stacked what-where auto-encoders. *arXiv preprint arXiv:1506.02351*, 2015.