Learning To Generate Rich-content Images Conditioned On Semantic Labels

Tsun-Hsuang, Wang
102061210
johnsonwang0810@gmail.com

Chan-Wei, Hu
102061247
huchanwei1204@gmail.com

Ping-Yen, Lin
102061130
leonlin1275@gmail.com

Abstract

In this final project, we try to build a model constructed by CNN (Convolutional Neural Network) to generate authentic real-world images. We present a state-of-the-art method, Stacked GAN[1], aiming to improve the generative model performance. In previous works, there are lots of works focus on utilizing more training tricks to stable generative adversarial network. Instead of just adding more convolution-transpose layers and utilize more training tricks, we put our attention on the model structure. By breaking the generative process into several stacks and adding more supervision provided by an auxiliary encoder, we expect to build a relative stable and diverse network capable of generating higher resolution images with comparatively stable training process.

1. Introduction

There has been a large interest in generative models recently, DCGAN (deep convolutional generative adversarial network)[2] is a typical example. Mainly composed of CNN, DCGAN wants to generate images that captures distribution of real-world images. These are models that can learn to create data that is similar to data that we give them. The intuition behind this is that if we can get a model to generate high-quality image, then it must have also learned a lot about images in general (means that the model can really learn the details of data). In other words, the model should also have a good internal representation of images, so it becomes some kind of "understanding" the whole content of its target instead of just "recognizing" the target. We can then hopefully use this representation to help us with other related tasks, such as classification, localization, segmentation...etc.

Generative Adversarial Networks(GANs) are an interesting idea that were first introduced in 2014 by Ian Goodfellow et al. [3]. The main idea behind GAN is to have two competing neural network models. One takes noise as input and generates samples (and so is called the generator). The other model (called the discriminator) receives samples from both the generator and the training data, and has to be able to distinguish between the two sources. These two networks play a competing game, where the generator is learning to produce more and more realistic samples, and the discriminator is learning to get better and better at distinguishing generated data from real data. These two networks are trained simultaneously, and the hope is that the competition will drive the generated samples to be indistinguishable from real data.(See Figure.1)

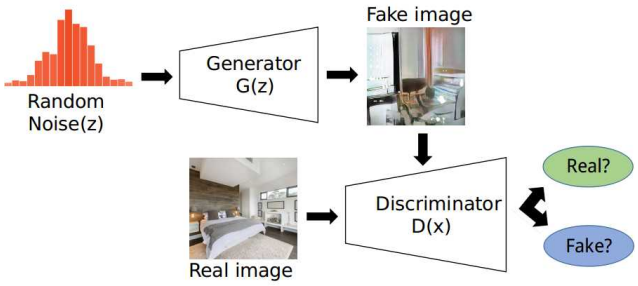

Figure 1. General system block

2. Related work

2.1. Generative Adversarial Network

Generative Adversarial Network is first introduced by Ian Goodfellow et al.[3]. They introduced neural networks that are trained in an adversarial manner to generate data mimicking some distribution. In the model, there are generative model and discrminative model. A discriminative model tries to distinguish the real image or the generated image while generative model tries to learn the joint probability of the input data and labels simultaneously and fool the discriminative model.

2.2. Deep Convolutional GAN

Deep Convolutional Generative Adversarial Networks is introduced by Radford et al. [2] They proposed a class of CNNs called deep convolutional generative adversarial networks (DCGANs), that have certain architectural constraints, and demonstrate that they are a strong candidate

for unsupervised learning. They showed convincing evidence that our deep convolutional adversarial pair learns a hierarchy of representations from object parts to scenes in both the generator and discriminator.

2.3. Info-GAN

Info-GAN, which is introduced by Xi Chen et al.[4], is an information-theoretic extension to the Generative Adversarial Network that is able to learn disentangled representations in a completely unsupervised manner. They derive a lower bound to the mutual information objective that can be optimized efficiently, and show that our training procedure can be interpreted as a variation of the Wake-Sleep algorithm.

2.4. StackGAN

StackGAN is introduced by Han Zhang et al.[5]. They propose stacked Generative Adversarial Networks (StackGAN) to generate photo-realistic images conditioned on text descriptions. Importantly, their StackGAN for the first time generates realistic 256 x 256 images conditioned on only text descriptions, while state-of-the-art methods can generate at most 128 x 128 images.

3. Method

3.1. Overview

In the following section, we will elucidate the entire workflow of SGAN, including how the network is constructed, several losses for different objectives along with some intuitive explanation over corresponding formula, and how to train a SGAN. In addition, we will further elaborate details of how we extend the vanilla SGAN to a deeper network so as to cope with more challenging problems.

3.2. Basic SGAN

As mentioned in previous sections, one of the notorious characteristic of GAN is its extremely unstable training process. To come the rescue, SGAN brings up a brilliant concept as providing extra supervision during training. In comparison to DCGAN which end-to-end trains cascaded generator and discriminator, SGAN splits them into several stacks, accompanied with pretrained encoder guiding each of them. Such enlightening idea can be concluded as:

1. Use the existing deep convolution encoder structure to guide the generator on feature space.

2. Perform adversarial training process on different feature level.

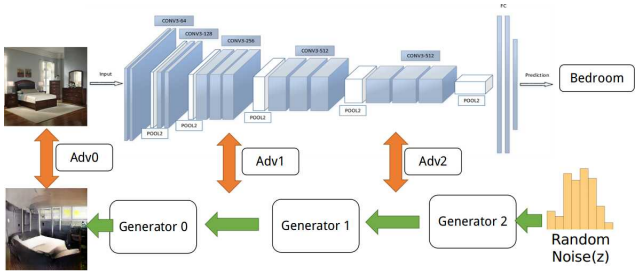

Figure 2. SGAN general idea.

Pretrained encoder: For each experiments, we need to pretrain the CNN encoder in advance. For MNIST dataset, we use the classical LeNet as our encoder[6]. For CIFAR10 dataset, we adopt the modified Lenet, adding more channels in convolution layer to obtain richer feature encoding. The encoder is trained to extract different level feature representation of images, and provides those encoded representation as oracle for the learning process of generator.

Stacked Generator: Generator's purpose is to invert the manner of the CNN encoder. Assume the encoder perform the task $y = E_\phi(x)$, where x is the input image, y is the output prediction and ϕ is the network parameters. Then, each intermediate encoding process, i.e. each stack, can be viewed as $h_{i+1} = E_{\phi_i}(h_i)$, where i indicates the i-th stack of layers with nonlinear mapping functionality.

In this manner, the goal of each generator stack is to perform the invert upsampling. That is $\tilde{h_i} = G_{\phi_i}(\tilde{h}_{i+1})$ where $\tilde{h_i}$ and h_i is in the similar feature level representation but not exactly the same. Through the stack-wise independent training, each generator stack can learn respective feature level mapping along with the guidance of corresponding encoder stacks. Meanwhile, adversarial training is performed between each stacks of generator and discriminator.

Stacked Discriminator: Discriminator stacks is built to perform supervision on learning of different generator stacks. On different feature level, the discriminator perform similar task to the encoder but predicting the realness of the generator's output instead. Aside from the adversarial learning, a second branch is forked to reconstruct the input noise fed to the generator, formulating the entropy loss which will be later-on mentioned.

Loss Function: Each generator stack G_i is trained subject to three different loss function: adversarial loss, conditional loss and entropy loss:

$$L_{G_i} = \lambda_1 L_{G_i}^{adv} + \lambda_2 L_{G_i}^{cond} + \lambda_3 L_{G_i}^{ent}$$

where $L_{G_i}^{adv}$, $L_{G_i}^{cond}$, $L_{G_i}^{ent}$ denote the adversarial loss, conditional loss and entropy loss respectively. λ_1, λ_2 and λ_3 are the weights associated with different loss terms. In practice, the weight can be adjusted to fit different generative tasks which will be elaborate in the experiment section. In the following section, we will elaborate the three loss terms in detail:

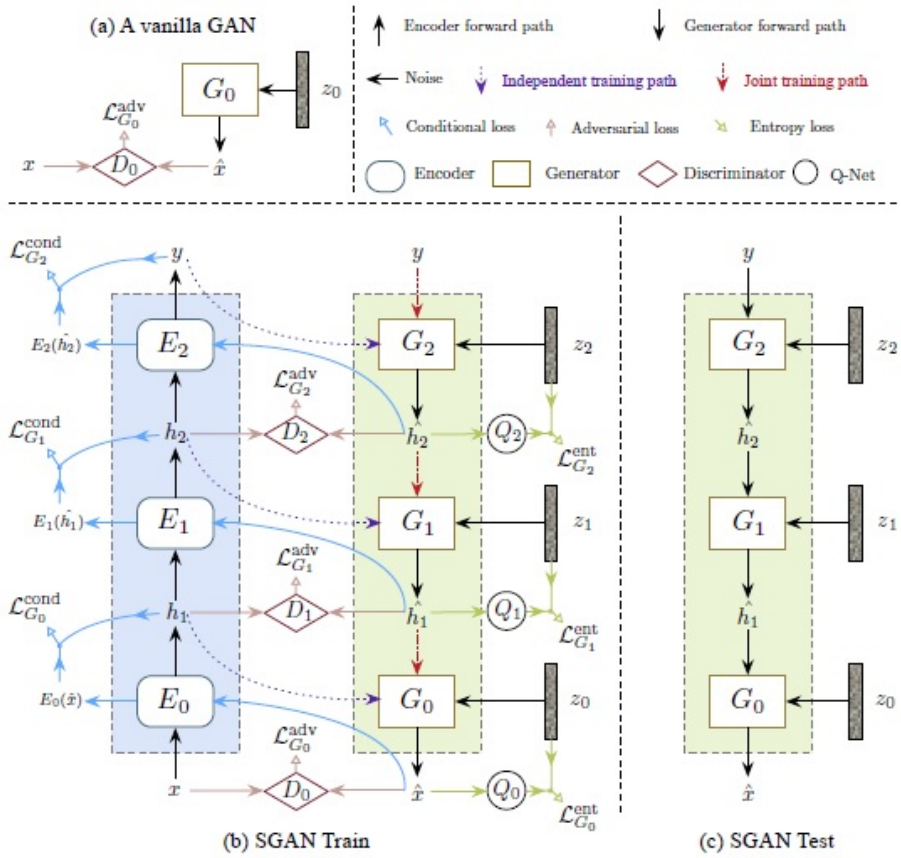

Figure 3. (a) vanilla GAN mentioned in [3], (b) SGAN workflow during training, (c) SGAN workflow during testing

1. **Adversarial Loss:** Inheriting the essence of adversarial learning, adversarial loss is simply referred to as a two-player min-max game to train G_i and D_i. At i-th stack, $G-i$ strives to generate images that can fool D_i by minimizing the loss,

$$L_{G_i}^{adv} = \mathbb{E}_{h_i \sim P_{data,\mathbb{E}}}[-logD_i(h_i)]+$$

$$\mathbb{E}_{z_i \sim P_{z_i}, h_{i+1} \sim P_{data,\mathbb{E}}}[-log(1 - D_i(G_i(h_{i+1}, z_i)))],$$

and D_i tries to distinguish real images from generated images, playing against generator with minimizing the loss,

$$L_{D_i}^{adv} = \mathbb{E}_{h_{i+1} \sim P_{data,E}, z_i \sim P_{z_i}}[-log(D_i(G_i(h_{i+1}, z_i)))],$$

where z_i is the random noise sampled from Gaussian distribution P_{z_i}, P_{data} is the distribution of real-world images, and stack's index i is sorted from low to high as low-level representation (close to image) to high-level representation (close to final semantic meaning) of images.

2. **Conditional Loss:** At each stack, generator is trained to capture the distribution of lower level representations conditioned on higher level representations. This is where encoder stack provide oracles as guidance. We feed the generated lower-level representations back to the corresponding encoder stack to encode it back to the corresponding feature space and force the recovered representation to be similar to the representation on which current generator stack conditioned:

$$L_{G_i}^{cond} = \mathbb{E}_{h_{i+1} \sim P_{data,E}, z_i \sim P_{z_i}}(F),$$
$$F = [f(\mathbb{E}_i(G_i(h_{i+1}, z_i)), h_{i+1})],$$

where f is the distance measure. We use Euclidean distance for the intermediate distance measure and cross-entropy for the labels.

3. **Entropy Loss:** Simply adding conditional loss raises another issue, i.e. G_i ignores random noise z_i and collapse to a deterministic model. This is a common phenomenon in conditional GAN, called*conditional model collapse*. To avoid such issue, as mentioned before, we force the discriminator to reconstruct the latent noise, enforcing D_i to internalize z_i, meanwhile, prospering the diversity of generative model.

$$L_{G_i}^{ent} = \mathbb{E}_{z_i \sim P_{zi}}[\mathbb{E}_{\hat{h_i} \sim G_i(\hat{h_i}|z_i)}[-\log Q_i(z_i|\hat{h_i})]]$$

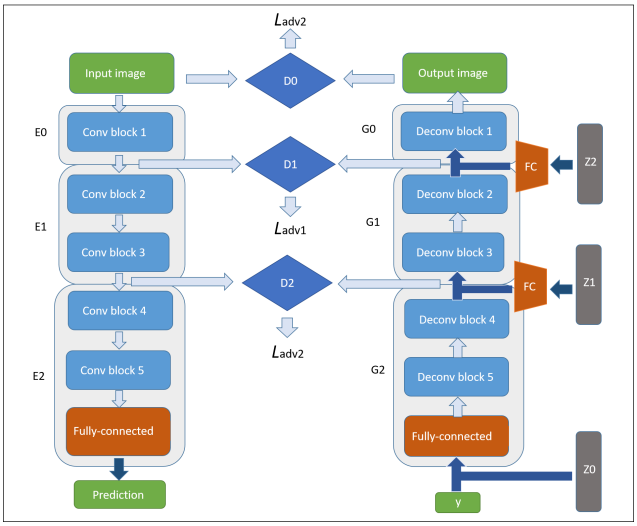

Figure 4. VGG16 encoder + generator + discriminator

In our implementation, we simply calculate L2 loss for the reconstruction of z_i.

Training Process: There are two stage of training SGAN, independent training stage and joint training stage. At independent training stage, higher level representations, which G_i is conditioned on, come from the oracles h_{i+1}, guided by corresponding encoder and the real images. At joint training stage, higher level representations come from $\tilde{h}_i$, output of previous stack generator G_{i+1}. Joint training stage is simply regular training strategies of GAN, and independent training stage is like a step-by-step teaching process that, hopefully with some guidance, generator can learn better.

3.3. Going deeper with SGAN

In the original SGAN submission, they only try simple network structure[6] on relative simple task[7, 8]. The performance and the inception score is quite impressive including the diversity, conditional control and stability of the proposed model. Therefore, the idea came to us that whether we can generalize this concept to a more complicated task and a deeper network architecture. We move our attention to the well-known deep VGG16 convolution neural network[9]. With the deeper encoder structure and luxurious feature, perhaps we can build a deeper and stabler generator architecture.

We try our idea on the Place2 dataset[10], which contains images with much higher resolution, forming a much challenging task compared to MNIST or CIFAR10. We use VGG16 model as encoder and design corresponding generator and discriminator stacks according to network architecture of VGG16.

Design of discriminator is similar to that of vanilla SGAN. For generator G_i, there may be some differences

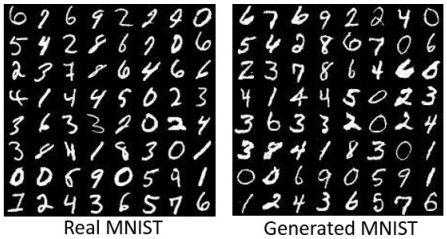

Figure 5. results on mnist

from the very simple SGAN architecture used for MNIST and CIFAR10. At each stack of generator, latent noise z_i is concatenated to higher level representation. However, if the entire generator network goes deeper, those intermediate representation may be a feature map. In our implementation, we use fully-connected layers to extend dimension of z_i, reshape to 2D size (width and height) of intermediate feature map, and finally concatenate it along channel dimension. The overall structure is shown in Figure 4.

4. Experiments

We have done experiments of SGAN on different dataset, including MNIST, CIFAR10, and Place2. In the following section, we will elaborate respectively results and associated analysis.

4.1. SGAN on MNIST

We conduct our a series of experiments, starting from the simplest dataset: MNIST[7]. We use LeNet[6] as encoder, which is trained for 25 epochs and reach accuracy = 99% on validation set. For generator, we invert the structure of LeNet: concatenate the conditional one-hot label, indicating digit number 0∼9, and latent noise as input, feed the input to a series of fully-connected layer to produce a intermediate representation vector, subsequently passing to 2 deconvolution layers for upsampling, finally output images with target resolution 28 x 28.

4.2. SGAN on CIFAR10

As mentioned above, there are two stacks of generator, and so does discriminator. Initially, each stack is trained independently with the 3 objective loss and the oracle provided by corresponding encoder stack. After independently trained for a while, we alter to joint training stage. Now each stage condition on the output of the previous stage, not the encoder oracle. Note that joint training is not required for discriminator. During the training process, we use the Adam optimizer with learning rate = 0.0002 and momentum = 0.5. For the 2 generator stacks, we both utilize latent noise z with 50 dimension.

The result is shown in Figure 5. We can find that digit number between real and generated images, implying that

Figure 6. results on CIFAR10

the generated images are actually conditioned on given semantic labels. Besides, diversity of generated images is manifest, as digits generated have different styles and fonts in comparison to the real digits. Our experiment on MNIST is pretty successful.

After achieve amazing results on MNIST, we move on to a more complicated dataset with natural images and higher resolution: CIFAR10[8]. Since the images in CIFAR10 is just slightly bigger than which in MNIST (32 x 32 v.s. 28 x 28), we adopt the similar encoder structure to LeNet, with a little change on the kernel size and number of channels of convolution layer and number of units in fully-connected layers, providing the network with adequate capability to capture distribution of low-resolution natural images.

We train the encoder for 50 epochs and it reaches the accuracy of about 95 % on validation set. Then, similar to the experiments on MNIST, we freeze the encoder and construct the generator and discriminator counterpart. We train both the independent part and the joint part for 50 epochs. During the training process, we use the Adam optimizer with learning rate = 0.0002 and momentum = 0.6. For the 2 generator blocks, we both utilize latent noise z with 50 dimension.

The result on CIFAR10 is displayed on Figure 6. The generated images are not bad, but they do not condition on given semantic labels (there is no correspondence on object categories between real image and generated image given the same labels). We have tried several methods to fix this problem but unfortunately none of them works. We have tried to enhance, in different degree, the conditional loss at different stacks, but it results in conditional model collapse and suppress diversity of generated images. After time of struggles, we found that SGAN author post their work over CIFAR10 very recently, and we figured out that the objective loss being used are different from the description of their paper. We guess that maybe that is the point, but we fail to implement it before presentation.

4.3. SGAN on Place2

After two pilot experiments, we try our idea on the Place2 dataset [10] in the end. We take only 10 categories out of total 365 categories, simplifying our task. We use VGG16[11] pretrained model and finetune it on our simplified 10-class indoor scene dataset, which contains 50,000 images with resolution 256 x 256. In order to fit the size of VGG16, input images are resized to 224 x 224.

We confront abundant of problems over experiment on Place2. While training VGG16 encoder, we first utilize the Adam optimizer with learning rate = 0.001 and momentum = 0.5. It ends that the model can't converge at all under such setting. Then, we change the optimizer to momentum optimizer with stochastic gradient decent, learning rate = 0.001 and momentum = 0.9. Under this setting, we can train the network successfully and obtain the competitive top-one accuracy of around 69% on the sampled testing set (top-one accuracy of whole 365 categories classification is about 55% for VGG16).

After finishing the encoder training process, we start to construct the generator and discriminator part. Different from the previous setting, to invert the VGG16 structure is much more challenging. We divide entire SGAN architecture into 3 stack, shown as Figure 4. In our setting, each de-

Figure 7. VGG16 result on Place2

convolution block is constructed by a series of operations: A convolution-transpose with stride 2, for upsampling, followed by batch normalization and leaky ReLu, and subsequently connected to 2 convolution layers with 'SAME' padding to do refinement in feature space. In each stage, the latent space noise and the conditional label (only fed in G2) will be fed into the fully-connected layers with batch normalization layers to produce the tensor with the desire shape.

As for the discriminator, we simply perform the series of convolution with stride 2 to shrink the spatial resolution. On each step, batch normalization and leaky ReLu is applied after the operation to stabilize the training process.

Empirically, we feed latent noise with higher dimension in the lower level stacks due to the larger spatial resolution. In such manner, we can ensure that the entropy loss still has valid functionality in spite of the growing spatial dimension of the feature map. In practice, we use $dim(z_0) = 100, dim(z_1) = 1000, dim(z_2) = 3000$.

Actually, among our numerous experiments, our model mostly failed to converge to a good state. Often, the jointly-inference generated images kept looking better but stop improving at certain points and start degenerating. For results in Figure 7, it took about 1$\sim$2 days for training and unfortunately, the realness of generated images declined after then and we stop the training process. We have thought of some reasons like inadequate capability of our model and make efforts to increase richness in channels of deconvolution layer and fully-connected layer. Besides, we have tried different weights for the 3 loss. Modified version of encoder guidance over generator is also carried out. However, none of them works. Learning from our CIFAR10 experiment, maybe we have to try different formulation of objective loss, yet its effectiveness remains to be further discovered.

5. Conclusion

We have implemented vanilla SGAN on MNIST and CIFAR10. The former results are amazing, generating digit images with diverse styles conditioned on given numbers, but the latter results fall short of our expectation, i.e. the conditioning power seems not working well. In addition, we further extend SGAN to a deeper and more powerful network structure utilizing the well-known VGG16. Our task becomes much more challenging and we bring up some unsuccessful but worth-seeing results on discovering SGAN. There are still a lot to be worked on, and hope we could keep taking further strides.

References

[1] Omid Poursaeed John Hopcroft Serge Belongie Xun Huang, Yixuan Li. Stacked generative adversarial networks.

[2] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In ICLR, 2016.

[3] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In NIPS, 2014.

[4] R. Houthooft J. Schulman I. Sutskever X. Chen, Y. Duan and P. Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In NIPS, 2016.

[5] Hongsheng Li Shaoting Zhang Xiaolei Huang Xiaogang Wang Dimitris Metaxas Han Zhang, Tao Xu. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. 2016.

[6] Yann LeCun Leon Bottou Yoshua Bengio and Patrick Haner Ab. Gradient-based learning applied to document recognition. IEEE 1998.

[7] Yann Lecun and Corinna Cortes. The MNIST database of handwritten digits.

[8] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. technical report, 2009.

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

[10] J. Xiao A. Torralba B. Zhou, A. Lapedriza and A. Oliva. learning deep features for scene recognition using places database. advances in neural information processing systems 27. NIPS, 2014.

[11] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Antonio Torralba, and Aude Oliva. Places: An image database for deep scene understanding. *arXiv preprint arXiv:1610.02055*, 2016.