Experiment on class-conditioned DCGAN

Tuesday, May 5th, 2017

1 GAN Model

1.1 Model

We experiment with two different types of GANs for this assignment, including DCGAN (deep convolutional GAN), WGAN (Wasserstein GAN). A DCGAN uses a convlutional network for both the generator and the discriminator instead of using MLP for the tasks. The deep convolutional structure gives the model a better ability to perform image recognition, while WGAN uses a different distance between the real data distribution and the fake data distribution for the loss function; this has given GAN a more stable way for training according to the loss function.

1.2 Dataset

The dataset we used for this assignment are MNIST and cifar10. The reason is that both datasets contain ten classes, which is much easy for us to switch between datasets using the same model architecture setting.

1. MNIST hand-written digits, 10 classes: 0 - 9
2. cifar10 small 32×32 images, 10 classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck

1.3 Task

In the vanilla GAN setup, images created by the trained generator are controlled or constrained by the data that we feed into the model; a good generator may be obtained from fine-tuning, however, the generated images may be very random, especially when an input data has a broad range of classes. In many circumstances, generating reality-like images may help train the discriminator for better classification, but the generated images do not seem to give much interest for any purpose. As a result, we chose to perform the conditional GAN task.

We tried conditional GAN by providing true label information for model training. Given such information, we hoped to see that the trained model can generate pictures which is not randomly from what it learned from its training data, but pictures which are directed by the condition information.

We implemented this task using the DCGAN architecture, while the original paper uses the vanilla GAN [2] and compared the results with that from the default DCGAN [3] and WGAN [1].

As for the context representation, we have two approaches. One of them is to just use a one-hot vector for the ten classes; the other one is to use a 300-dimensional word embedding which is obtained from GloVe.

1.4 Experiments

Shared parameters for the results are:

- batch size: 64
- image size 64×64 (note that, the cifar10 dataset is originally set to have image size 32×32, thus the generated results from the model is blurry as they are being zoomed out).

1.4.1 default DCGAN

DCGAN, inferred from its name, it uses deep convolution for both the generator and the discriminator.

- Discriminator : Four-layer CNN followed by a final conv layer which transform input to a signal value for classification, where the classification function is a sigmoid non-linear function to give the probability of whether the input data is a real image or not.
    1. nonlinear function at each layer: Leaky ReLu
    2. batch normalization at each layer except the very first layer (since we may lose a lot of useful features if we do the normalization at such an early stage)
- Generator : Four-layer decov NN to createa 2d pictures from a 1d noise vector
    1. nonlinear function at each layer: ReLu, except the final nonlinear layer to be a tanh function
    2. batch normalization at each layer

In DCGAN setup, we update the generator whenever the discriminator is updated.

1.4.2 conditional-DCGAN with additional MLP layers after context concatenation

- Discriminator: Similar architecture as the default DCGAN except in the penultimate layer, we concatenate the context C vector. This concatenated layer is then undergone an additional three-layer MLP network for training a model which includes label information; and a final sigmoid function is used for calculating the probability of a real image.
- Generator: Same as the default DCGAN except now it takes a concatenated vector of z and context C instead of just a random noise z.

1.4.3 conditional-DCGAN with additional decov-conv layers after context concatenation

- Discriminator: Very similar to the architecture of the conditional-DCGAN with additional MLP layers. However, instead of using a simple three-layer MLP network, here we explore to use deconvolution after the context concatenation, and do another convolution for the finally classification task. For each of the deconvolution and convolution, we have built four layers for both.

- Generator: Same as the generator used in the conditional-DCGAN with additional MLP layers.

1.4.4 default WGAN

In addition to the aforementioned experimental GANs, we have also tried using WGAN, which is assumed to generate better images.

All setup for the WGAN is nearly the same as the DCGAN setup, except that WGAN uses a different loss function - Wasserstein distance. Furthermore, the generator in WGAN is updated every five updates of the discriminator.

1.5 Results

Figures 1 - 5 are generated using conditional DCGAN, thus the arrangement of the images is that, 5 images for each class in a row-based manner.

Figures 6 and 7 are generated using default WGAN and default DCGAN which do not have the condition setting, the generated outputs are random images.

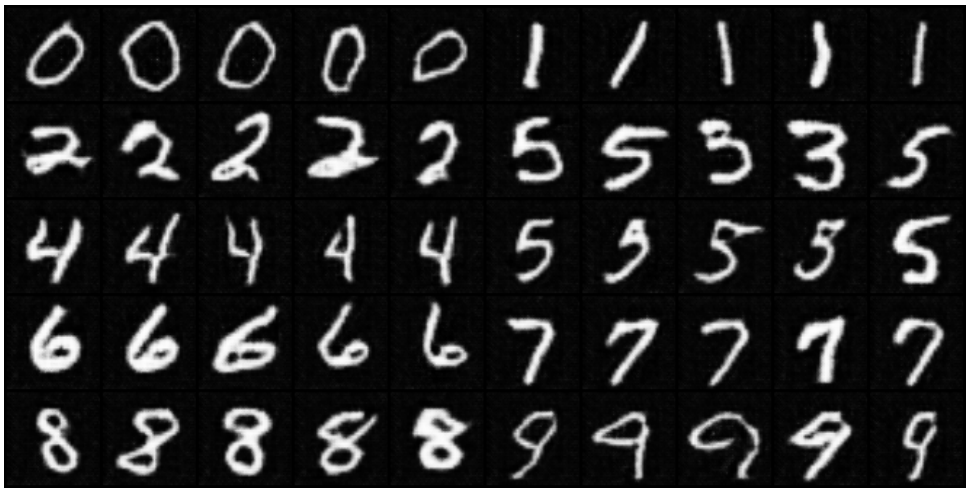

Figure 1: Generated MNIST images using conditional-DCGAN with one-hot embedding of context vector

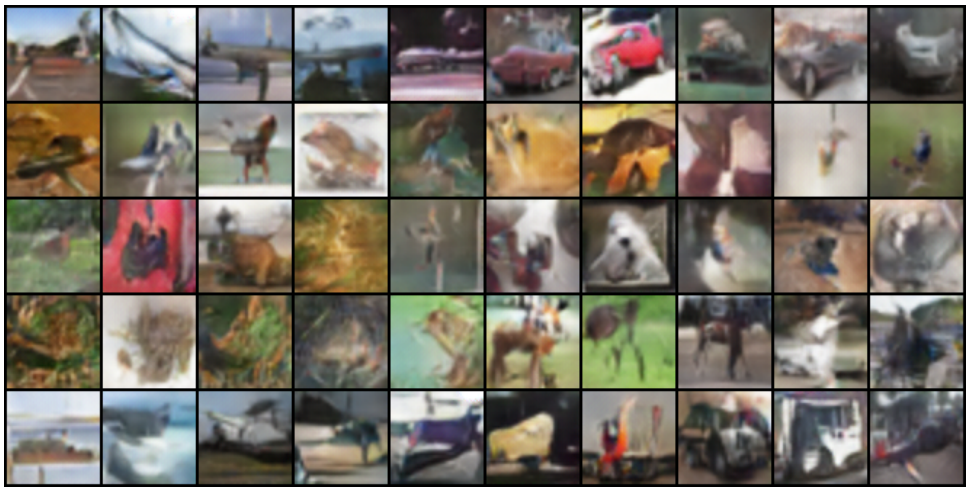

Figure 2: Generated cifar10 images using conditional-DCGAN with one-hot embedding of context vector (MLP additional layers)

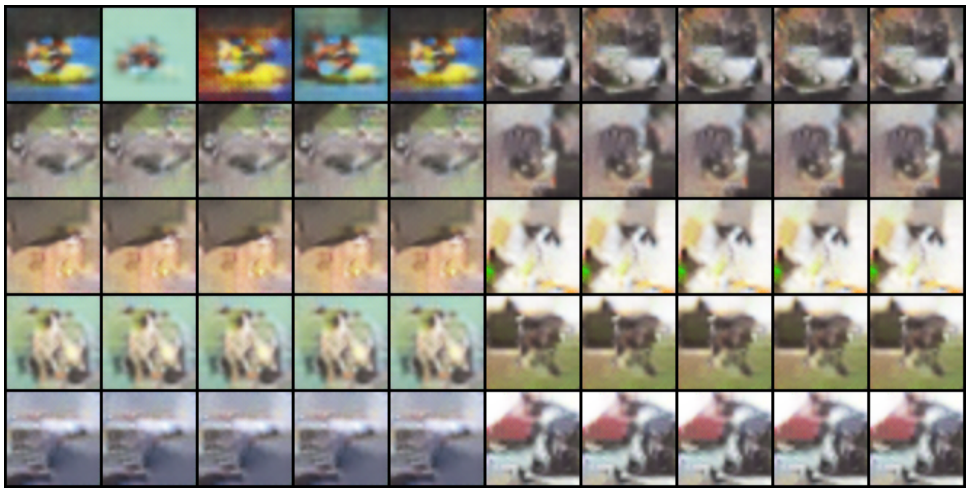

Figure 3: Generated cifar10 images using conditional-DCGAN with 300-dimensional embedding of context vector (MLP additional layers)

Figure 4: Generated cifar10 images using conditional-DCGAN with one-hot embedding of context vector (Decov-cov additional layers)

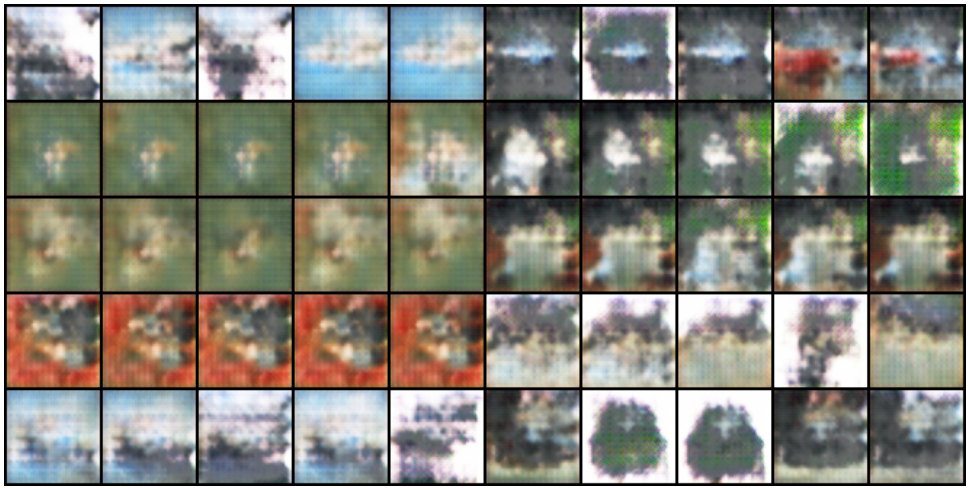

Figure 5: Generated cifar10 images using conditional-DCGAN with 300-dimensional embedding of context vector (Decov-cov additional layers)

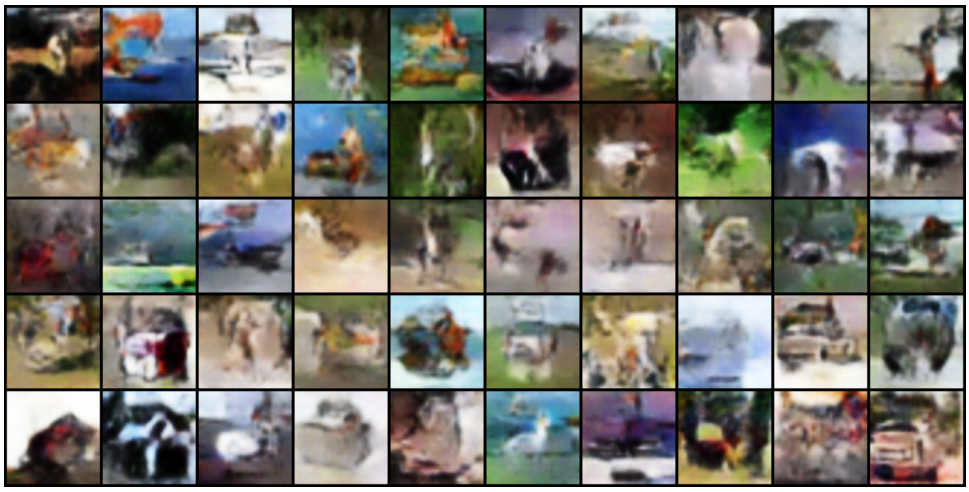

Figure 6: Generated cifar10 images using WGAN

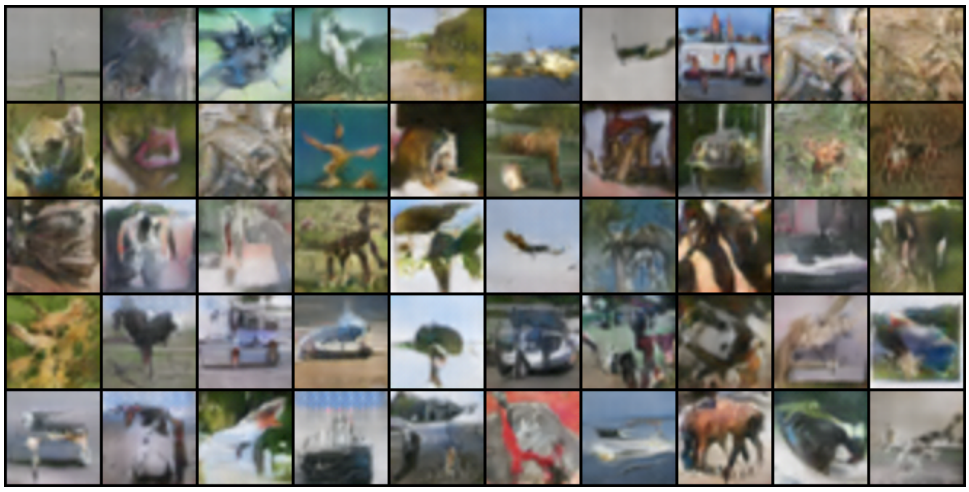

Figure 7: Generated cifar10 images using DCGAN

1.6 Analysis

Figure 1 is generated by a conditionial-DCGAN with one-hot embedding for the labels, results are pretty good, except the model makes some wrong image generations in digits 3 and 5. However, this seems reasonable, since sometimes even humans cannot recognize terrible writtend digits 3 and 5.

In general, as for the conditional GANs, context vector using one-hot embedding is better than the ones using 300-dimensional word embedding. This can be seen by comparison: (Figure 2 and Figure 3) or (Figure 4 and Figure 5).

First of all, the terrible results from the high-dimensional embedding may be due to the fact that we increase a lot of complexity to the model, while we haven't yet applied any regularization to the model. This has suggested that we should try dropout in further experiments.

Secondly, we can see that the high-dimensional embedding does not have a lot of variation in the five generated images in one class. This shows that the high-dimensional vector constrains the nearby pixels to have similar context information, again, another suggestion for regularization in the high-dimensional model.

We have obtained all models at around epoch 200, except the one for MNIST (MNIST is a very simple dataset, too many training epochs does not help). We use Adam as our optimizer in DCGAN, while WGAN uses RMSprop. We can see that WGAN generate clearer images than DCGAN. For further research, we would like to add the condition setting to WGAN, and explore more combination of different hyper-parameter for fine-tuning of the model.

References

[1] M. Arjovsky, S. Chintala, and L. Bottou. "Wasserstein GAN". In: *ArXiv e-prints* (Jan. 2017). arXiv: 1701.07875 [stat.ML].

[2] Mehdi Mirza and Simon Osindero. "Conditional Generative Adversarial Nets". In: *CoRR* abs/1411.1784 (2014). URL: http://arxiv.org/abs/1411.1784.

[3] Alec Radford, Luke Metz, and Soumith Chintala. “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”. In: *CoRR* abs/1511.06434 (2015). URL: http://arxiv.org/abs/1511.06434.