# Report to the First Research Turn

1800012969 Haoxiang Yang
Advised by Zhouhui Lian

## Abstract

Before this semester, I knew little about machine learning and had never tried any programs in this field. In the five-week first tern rotation, I am now confident to say that I am now a beginner, with the ability of reading papers, testing and altering models, and doing basic classification or generation tasks with my own network design.

## Description of the experience and what's learned and achieved

### 1. Preparation

Professor's Advise: Try to learn about the rationale of CNN; construct a shallow CNN classifier on MNIST by yourself and achieve correctness greater than 99% on test data; set up GAN and VAE generators, first on MNIST, then on general photos (if GPU resources available).

First, I downloaded and installed Tensorflow (CPU version)[1], Anaconda, Pytorch and VSCode on Win10. Later, in order to make use of my GPU, I uninstalled all these CPU version and re-installed the GPU version. It took me quite a lot of time to allocate the environment variables and have the CUDA matched, and the seemingly trivial task was a challenge for me, since I was not operating on an already-configured machine, which is the usual case, but on my own outdated computer. Now I am using Python 3.6 and CUDA 10.0 with VSCode to be an editor.

Second, I searched on the Internet some articles and blogs to acquire some basic knowledge about classification and generalization, including CNN[2][3], MNIST[4] and GAN. To verify my understanding of the blogs and check my configuration, I copied and ran some sample codes of CNN[5] and GAN[6] on my computer, and tried to tell myself the meaning of each line in the codes. I didn't gain the results of GAN using CPU, but did got them later.

During this time, the problems are that the rationale of GAN is still confusing as I haven't read the original paper, my GPU only has 2GB memory which does not support high-end large-scale researches, and my VPN was outdated and blocked. These problems are all solved later.

### 2. Paper reading

During these time I have read these following papers. (1) to (6) are some classical models in machine learning and deep learning. (7) to (9) are some latest researches in related fields. If you are not interested in this section, jump to the third part: Experiment.

(1) Reducing the Dimensionality of Data with Neural Networks [Hinton et al., 2006]

This paper mainly introduced a type of Autoencoder to do data dimension reduction (like PCA but act better than that) and the initialization of a neural network. From my view, the insight

of auto encoder intrigue further researches and provide a basis for the papers in the following decade. As the first paper read, I found some extra articles[7][8] on the Internet to gain more details and ran the code on the second article.

(2) Pixel Recurrent Neural Networks [Oord et al., 2016]

The main idea of the paper is PixelRNN. I didn't get the idea at the first sight so I found the corresponding presentation[9] on the Internet. The Row LSTM and Diagonal BiLSTM method sounds very attractive. Yet the Multiscale is seemingly more like human intelligence. Also I wonder how they get the functions (input, forget, output and context gate) and why they chose them. Are theses functions simply results of trials? Or there are some techniques and insights behind them?

(3) Some articles about Gradient descent and Back propagation

I knew some basic concepts but didn't know about details. Definitely, Gradient descent[10], Back propagation[11], De-convolution and Up-sampling[12][13] are fundamental in machine learning, for which I checked them online.

(4) U-Net: Convolutional Networks for Biomedical Image Segmentation [Ronneberger et al., 2015]

The paper use autoencoder to do classification in biology. It gave its network structure in its beginning, from which I can see a clear insight: connect the network from left-hand side (encoder) to right-hand side (decoder) so as to further pass the information and features while at the same time accelerate the training process.

The framework also utilize data augmentation. I searched for some articles[14] on the Internet to get some basic ideas, but I had no time to read more papers about data augmentation in this short rotation period. I also checked the meaning of momentum[15].

(5) Auto-Encoding Variational Bayes [Kingma and Max Welling, 2014]

The article is extremely abstruse. I had searched for more than five articles and listened to a presentation[16] before I knew what was going on. It is clear that VAE is an improvement of AE with its middle part detached. The encoder generate the mean and variation of a distribution, while the decoder does the sampling from the distribution. Using some analysis the author made the back propagation works.

From the Youtube video, I also learned that by adding a super global parameter we can control the network and force it to generate different pictures. Also it says that VAE generate pictures less clearly than GAN, which I assume is one of the major cause of the opaqueness in my experiment. The Youtube video is so educative that I was attracted to watch several other videos about reinforce learning[17][18] by the same author on it.

(6) Generative Adversarial Nets [Goodfellow et al., 2014]

The rationale of GAN is clear. It is surprising for me that the author proves the convexity of its function and the singleness of its solution. I guess it's related to game theory. Some questions remains: How is the size, channels decided? Do we have some methods other than trial one by one? And GAN is a simulation to two-player games. What about more players? After all, two-player

games are still limited (see adversarial samples).

(7) SinGAN: Learning a Generative Model from a Single Natural Image [Shaham et al., 2019]

The results in this paper is beautiful. What? It's ICCV 2019 best paper? Well, alright.

The paper is mainly about how to use GAN to accomplish tasks like image style Image migration, demosaicing, harmonization, image editing, drawing pictures by sketches, and transferring a simple image to a video, or to say, animation. It's approach is interesting: it used several GAN of various size to learn the features at different scales. Intuitively, this approach will succeed.

It is astounding that neural networks, although comprised of only mathematical formulas and calculations, act quite the same as humans' brains. That is to say, if we can realize other intuitive ideas like these, we would possibly further the networks' performance in the future. Yet it remains unknown whether we really understand its rationale, and what the boundary of the algorithms can be.

(8) Semantic Image Synthesis with Spatially-Adaptive Normalization [Park et al., 2019]

This paper is consistent to what I have said in the last paragraph. It repeatedly feed the images (characteristic, embedding) to the network (for four times) and gain better results, quite similar to feed a person some knowledge repeatedly to have me understood. It also use some other techniques, such as introducing full layers to represent means and variations.

(9) Large-Scale GAN Training for High Fidelity Natural Image Synthesis [Brock et al., 2019]

This article, to some extent, answered my question about how to design a GAN network. It claims that it makes sense to increase the width, depth and batch size of the network, use share embedding, and add a skip-z optimization (which means to connect the first layer embedding z to several consequential layers so that it will be able to 'control' them in the training). It also suggest a truncation of distribution when sampling, and normalization to avoid collapsing, especially when training a large scale of networks--with height and width doubled, as stated in the paper, the possibility of training failures increases significantly.

In sum, these papers are well-qualified and inspired me a lot in my learning process. The problem here is that I have no time (and probably no need) to run the codes one by one to testify the results and analysis the performance. And as can be seen in the following section, limitation on GPU resources can be really confusing for me.

**3. Experiment**

Note 1: All researches down here is completed with CPU: Intel(R) Core(TM) i5-7200U CPU@2.50GHz (RAM Memory 8G), and GPU: NVIDIA GeForce 940MX (RAM Memory 2G). I mainly use Pytorch when coding, but also use Tensorflow when no other choices are available.

**Note 2: Experiment 4 is the core task.**

(1) MNIST classification

Using a traditional CNN we can classify images of handwriting numbers from 0 to 9. The code is assembled by myself. After some trials, the network can now classify the ten digits with

correctness 99%+.

```
Finished Training
correct = 9935 total = 10000
Accuracy of the network on the all test images: 99 %
```

Figure 1: The output of the MNIST classification code.

(2) MNIST generation

Using GAN we can generate numbers similar to those in MNIST. The code is partly borrowed from the Internet.

One problem known here is that the output is sort of simplified, converging on three numbers: 3,7 and 9. This may be attributed to the fact that these numbers are relatively easier to generate and hard to discriminate. To overcome this, one has to add other penalty function to penalize convergence (not tested yet).

Figure 2:Visualization of generated numbers. From left to right is training results at 1/99, 10/99 and 99/99 of the epochs.

Note that number 3, 7, 9 occurs significantly more often than other numbers.

(3) Human Faces generation

Using GAN we can also generate faces with celebA (CelebFaces Attributes Dataset).

One problem is that the network is quite large, so the training process can be time-costing and annoying. Running a code takes me nearly 24 hours. Reflecting on that, I shall either save the model in the middle, or run the network with a shallower setting. These will be mentioned TBD in the next section.

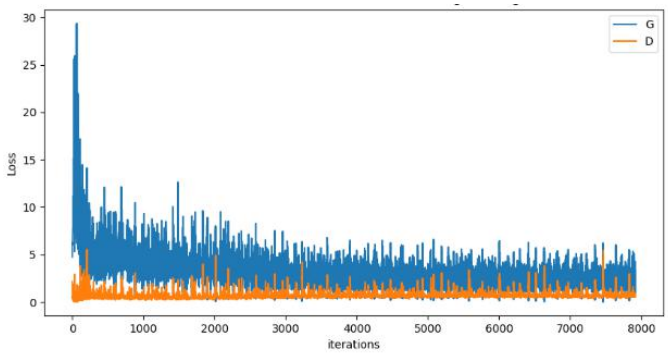

Figure 3: generator and discriminator loss function values during training

Figure 4: generated faces during training.


Figure 5: comparison between real faces and fake faces.

## (4) Zi2zi model (Core Task)

Zi2zi can transfer a Chinese character (zi) to another style or do more tasks. The original model is written by python2 with input 256*256, which I edited to python3 and 64*64 input. I spent about a week in testing the model.

The model is mainly comprised of a generator and a discriminator, and the generator is composed of an encoder and a decoder.

The encoder reads an input image as pixels, reshape it through several convolutional layers (for more details like the functions used, check my code on github), and transferring to a 128*1*1 tensor. The decoder reads the input from both encoder and the embeddings, and tries to restore what the encoder has seen. The encoder and decoder together forms a VAE, which is the input of the discriminator. It is well-worthy to note that the decoder borrow some part from the encoder--the decoder's layers are concatenated to the decoder layers, as explained in detail in Figure 8. This approach is borrowed from U-net. It improves and stabilizes performance since it link the information and enable encoder's greater control.

After the generator (the encoder and the decoder) generates an image, the discriminator trys to distinguish the fake images from the real ones. Then True/Fake Loss is calculated. (see Figure 6.) Category Loss is introduced to avoid style mixtures (not tested in this experiment). L1 loss (in Figure 6) is also calculated through a comparison.

The original model (256 * 256) is too large to be placed on my GPU. I deleted some layers

and so as to run the code. See more details in Figure 7, 8, 9. To gain better results I tried to make the network deeper, but there was little improvement.

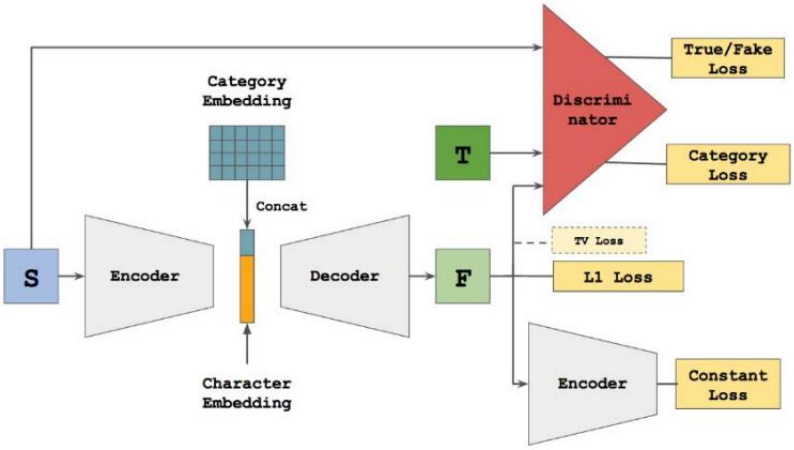


Figure 6: Network design details. First picture indicates the abstract model. Following pictures demonstrates some details.

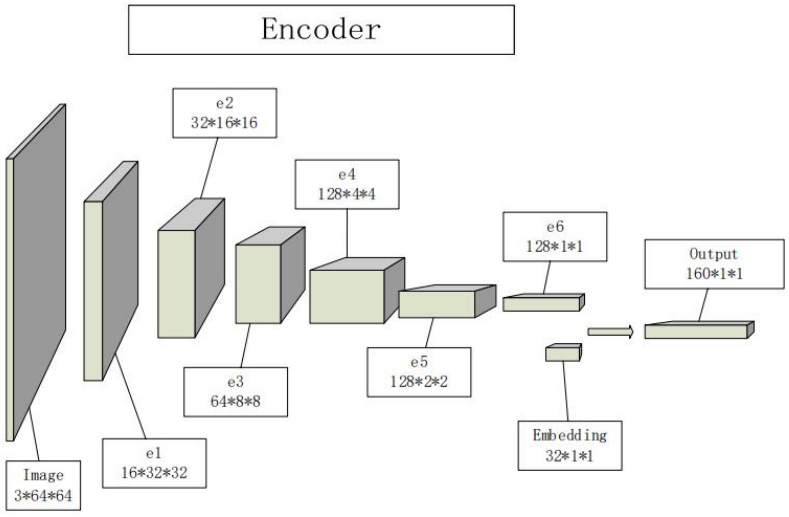


Figure 7: Encoder details. The input is 3*64*64, which means 3 color and 64*64 pixels.

The e6 ad embedding are concatenated directly in the last part to get the output.

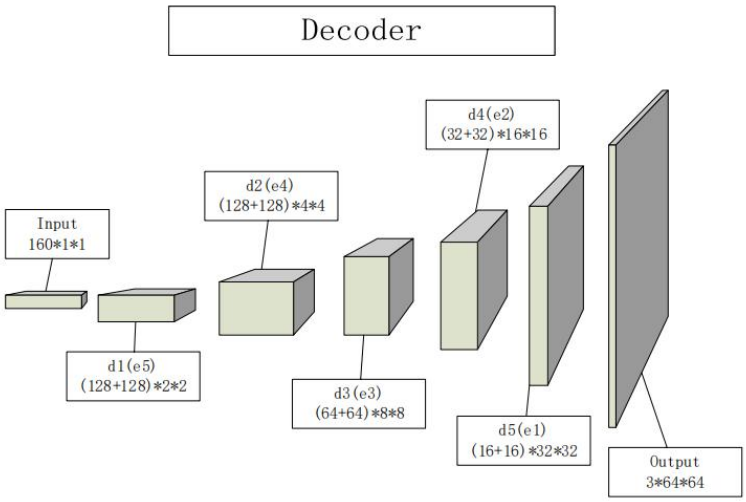


Figure 8: Decoder details. Each layer except the last one is concatenated to a former layer.

For example, after a convolution on the Input layer, d1 (old) is concatenated with former layer e5 to get the output d1 (new). Note that the size of d1 and e5 (height and width) match. The old output is 128 * 2 * 2, and is to be (128 + 128) * 2 * 2 now.

This framework is borrowed from (4) U-Net: Convolutional Networks for Biomedical Image Segmentation.

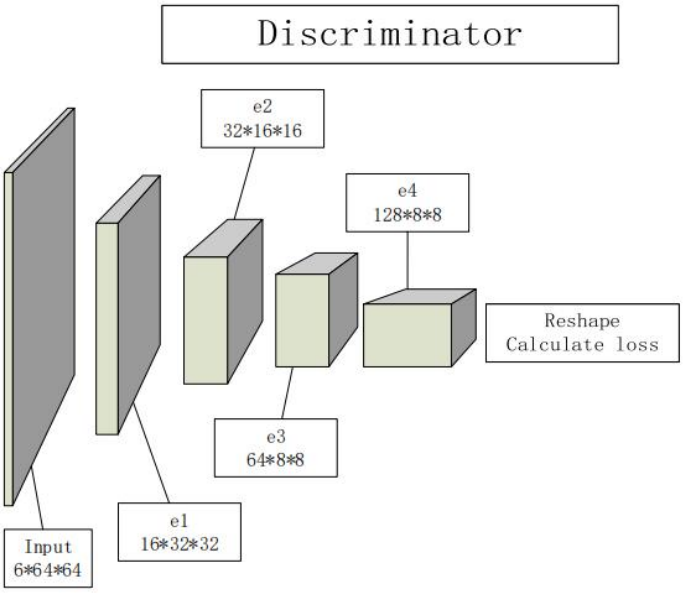


Figure 9: discriminator details. Similar to those of discriminator.

Here are some alternation worth mentioned. Pictures are shown after brief description.

1) L1 penalty: As L1 penalty increases from 10 to 10000 (multiplied by 10 each time, 100 is the default setting), the performance turned out to be better. I alter this so that the penalty applies to the 64*64 model. Other parameters is also tested but no improvements was found.

2) Epoch: As the epoch is increased from 100 to 300, only a little improvement is detected. It seems that the network converges very fast.

3) Style: As the typeface of character is changed, the results and performance is not changed intrinsically. You can see these various results in Figure 12. For bolder characteristics the

results seems to be worse.

4) More samples: as the sample number increase from 1000 to 3000, the results remain nearly the same. The outcome is still kinds of opaque and is not satisfactory, which means that the model has not learned more from more examples, but may be confused at the same time.

5) More diverse examples: use 3 types of characters as the source, each 1000 characters. Different from that stated in the passage, the results remain nearly the same.

6) Deeper layers: As layers become deeper, only a little improvement is detected. Therefore, I deem that the opaque parts in the results indicated a limitation of the network. It may be due to its size reduction or other factors like the nature of VAE and GAN.

**Code:**

In order to gain the results in these report, visit:

https://github.com/yanghaoxiang7/Yang-Haoxiang-s-First-Research-Turn

To find the original version, visit:

https://github.com/kaonashi-tyc/zi2zi

Figure 10: From left to right is L1=10, 100, 1000, 10000

Figure 11: Left is epoch=100, Right is epoch = 300

Figure 12: Four different types of characters (L1=10000) Some characters generated are still opaque.

Figure 13: Sample number=1000 (left), 3000 (middle), 1000*3 (right)

Figure 14: Left is the origin one. Right is when the layer channels is more than doubled. (all layers in encoder, decoder and discriminator)

It seems hard to discriminate these results, and the right-hand characters is still kinds of opaque,

which can be attributed to the relatively small size (256*256 -> 64*64)

## Reflection on the experience

(1) Experience and lessons

It is crucial to push myself in scientific researches. From my experience and some of my classmates' rotations, it is true that professor may not interfere too much when the student is doing researches. Yet with years in high school being monitored, it is urgent for me now to have myself accustomed to the atmosphere here.

Saving time is important. Treasuring my time does not mean having a person working all the day. Quite the opposite, I have to arrange the harder tasks and the easier ones so that I am doing them alternately. It's a pity that I cannot do more researches in this first stage of rotation, partly because I have wasted some time. Hopefully with more experience it will better next time.

Communication plays a indispensable role. By communicating with other a person correct its mistakes and find the shorter paths. Also, if I really want to achieve my final goal, I would have to convey my ideas, persuade others and demonstrate my competitiveness.

Additionally, machine learning is a fast-changing field. To follow up I must improve my English reading skills and learn to read faster and grasp the main idea of a passage.

(2) Things to be done

The following should be done when I came back to machine learning later, or when needed:

To study more types of GAN and their performance. Testify more layer-built-in functions.

To follow the latest result, and search for some recursive references and papers which refer to these papers.

Read the remaining papers about data augmentation and 3D generation.

(3) Some Ideas about GPU

Here's some ideas about the usage of GPU resources, as it has become a major obstacle when I do researches.

First, why can GPU accelerate the calculation? Parallel computing units. So now I have some more questions. GPU resources is very expensive. If we rely on the improvement in computing power to gain better results, GPU resources can be the limitation in our future. Can we split the network and calculate them separately, as different layers and the convolution in different positions can be calculated independently? Can we utilize distributed computing? It's well-worthy to note that Stanford is using GPU distributed computing power to fight against Coronavirus. Also, to compute something complex as a brain (maybe this is what we have to do in the future), we would have to make these approaches. That is to say, we still have a long way to go.

Second, no matter how large the network we can compute, we will always be annoyed by the too-long training process. Could we train a network at a small scale to a decent result, and then transfer the network to larger scale? Is this method valid? I have noticed that for some network, like zi2zi, no matter what kind of improvement I did, it remain kinds of opaque. For some earlier researches it's the same--No matter how we adjust the parameters, the performances only improves a little. Can we prove theoretically that if a small network perform well on small data, it will not perform too badly with larger version? Is the scale really important, or it can enhance the

outcome only in a limited term, as can be seen in my experiment?

In sum, I would claim that GPU resources is not something like people told me, ‘Oh, you cannot do any machine learning without GPU resources’. The GPU in the lab may run 100 times faster and have 8 times as much memory than my personal computer. Yet 100 and 8 are constants. If an algorithm itself does not perform well, nothing will be changed by an excellent GPU.

## Acknowledgement

My adviser, Lian Zhouhui, led me to this machine learning and character generation field. He provided me with lists and catalogs so that I could choose what I wanted to do. In this specific time period (Coronavirus), he chatted with me online regularly, viewed my report, made suggestions about my direction and corrected my mistakes in time.

My senior, Gao Yue, helped me a lot in my programming. He aided me in solving problems when I downloaded related software, and taught me the rationale of some codes online on TeamViewer for several continuous hours.

## References

[1]https://towardsdatascience.com/setup-an-environment-for-machine-learning-and-deep-learning-with-anaconda-in-windows-5d7134a3db10
[2] https://www.jianshu.com/p/9afc598fda2d
[3] http://cs231n.github.io/convolutional-networks/#overview
[4] https://www.cnblogs.com/xianhan/p/9145966.html
[5] https://www.jianshu.com/p/a652f1cb95b4
[6] https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html
[7] https://blog.csdn.net/qq_39521554/article/details/80697882
[8] https://www.jianshu.com/p/eacb36e201df
[9] Available on bilibili. https://www.bilibili.com/video/av73995520
[10] https://segmentfault.com/a/1190000019862084
[11] https://www.jianshu.com/p/39d6711430ff
[12] https://blog.csdn.net/nijiayan123/article/details/79416764
[13] https://blog.csdn.net/weixin_41012399/article/details/94406997
[14] https://blog.csdn.net/lanmengyiyu/article/details/79658545
[15] https://www.cnblogs.com/jiaxblog/p/9695042.html
[16] Available on Youtube. https://www.youtube.com/watch?v=9zKuYvjFFS8
[17] https://www.youtube.com/watch?v=JgvyzIkgxF0
[18] https://www.youtube.com/watch?v=Ol0-c9OE3VQ