Report of the Internship

Introduction

If you want to implement machine learning algorithm by yourself, you have to be familiar with the conventional framework, and the most popular frameworks available nowadays are : Pytorch and Tensorflow.
Furthermore, Having a good command of machine learning framework can also help understand the idea of the paper better, because it is easier for you to understand the source code and relate the main idea of the paper to the very detailed implementation in the source code, which might help you to have a better understanding of the paper.
In this internship, my main job is to convert the source code that is written in Tensorflow to Pytorch, which is not easy at the very first glance as I was given this task (Because at that time I was not very familiar with the modules in both frameworks). In the following paragraphs, I am going to talk about how I reimplement the code in the Pytorch framework.

Method

Read through the paper and try to use my own language to write down the pipeline

Firstly, I tried to read through the paper (not only the theoretical part but also the experimental part), in order to have a rough idea of what is the structure of the neural network, how does it differ from a traditional VAE model and what is name and the dimensions for the input data etc. I drawed a pipeline to intuitively explain what kind of components I have to have to reimplement the model. (See the pdf “Pipeline” attached in the folder)

Find out the run.py to execute the source code line by line

At the very first glance of the source code, I was really shocked by the scale of it: within the project folder, there are nearly 20 py files, and each might contain hundreds of lines of codes (if not thousands). How can I reimplement the code of such a large project? Before this project I am not good at programming and all that I have done is just several rather small programming tasks (none of them is longer then 100 lines of codes). After a week’s time struggling understanding the code, I figure out a good way to read through the code and

understand which part of the code corresponds to which part of the paper: find out the run.py and execute the code one line after another.

While doing this, I also figure out that I can use Pycharm's debugging mode to watch the variable and see the results of certain operations applied to the variable. By doing so, I can interact with the code as the code is running and I don't have to use print to print out the variable that I am interested in, which is totally not practical for a relatively large project like this.

Understand the main difference of Pytorch and Tensorflow and tried to find the equivalent/similar functions in Pytorch

When executing the source code line by line using pycharm's debugging mode, I drawed a mindmap (see pdf Function-dependencies_Mindmap attached) about the exact order of code's execution and tried to figure out which part is purely written by tensorflow framework and how I can reimplement that in Pytorch, which works well in micro-level and helps a better understanding of both frameworks but might later on lead to problem, because there is an essential difference between Pytorch and Tensorflow when it comes to this source code: In the source code, the author builds a static graph at first, and thus placeholders are needed. After executing sess.run() method, the graph computation can then be executed. However, for Pytorch the graph is built dynamically, which means we have to identify all the learnable parameters (neural network structure) when we call a certain model, and only then we can feed input data to the model.
That is to say, for tensorflow: we need placeholders and define static graph before hand. While for Pytorch, we should explicitly define neural network structure (building block) in the __init__ method, and in the forward method we then use these building block to build up the neural network and build up the graph (flow of computation).

Polish the code

Because of the difference between Pytorch and Tensorflow, it is impossible to reimplement the code in a 1-to-1 correspondence way, I have to think over some technical details such as: for Tensorflow we can directly define how to regularize the weight parameters without adding a regularization term to the final loss, what kind of constraint we want to put on the weights/ bias of the layer and how we want to initialize the parameters of each layer individually as we define each layer. However, in Pytorch we can not directly do that. I have to initialize the parameters after I instantiate the model and calculate the regularization term and put constraints on the bias after I run a single batch of data.

Save results and model parameters, record training process using tensorboard

The final step would be to save the output and parameters of the model and use tensorboard to visualize the training process. There are slight difference for both framework.

Results

See the reimplemented code in the folder “vi-hds-master_Pytorch”

Conclusion

I spent around 50 days doing this internship, which is the best practice ever for me to get familiar with the both framework and how to transfer the theoretical mathematical expression and plain description in the paper to the practical code. I also have got familiar with the concept such as modularity, and importance of the class (Python is an object oriented programming language). I firmly believe that such experience is going to lay a solid foundation for my practical skill and will be helpful for my practical course in the coming semester and master thesis. I also want to thank Carlos, my supervisor of this internship, for his kindness and patience and all the instruction he gave me. It is a great pleasure for me to have him as my supervisor. I wish all the best to his PHD career and his academic career.