Improving YANN toolbox with Jupyter notebooks, test coverages

Anil Kumar Motupalli, Buddha Puneeth Nandanoor, *Masters(MCS) Computer Science*

Abstract—Training Deep Convolutional Neural network became practically feasible with the adoption of GPU computing which gave birth to number of neural network frameworks. Among these frameworks, yann is a simple, feature rich and easy to adopt framework for training Convolutional Neural Networks as it is developed for Image Processing with Convolutional Neural Networks. As a part of this project we want to enhance its stability and usability with unit test coverage and Jupyter notebooks as tutorials.

1 Introduction

YANN is a python centric neural network toolbox which is highly centered towards Convolutional Neural Networks which is developed by Ragav Venkatesan working in Image analysis for Image Analysts. This makes it one of the best toolboxes for supervised and unsupervised Image processing with Convolutional Neural Networks. YANN is developed using theano which is a stable, open source project with an active developer community. Therefore, YANN inherits theano's features mentioned above along with it's ability to train Neural networks with GPU. GPUs are made of thousands of lightweight cores. Though these GPU cores are not as powerfull as CPUS, they can perform floating point arthematic in highly parallel fashion which is ideal for training Neural Networks.

Python is the most adopted language in data science because of it's highly expressive syntax and ease of learning. Python enjoys a middle ground between highlevel and low level languages enjoying advantages from both. It is easy to use like high level languages and it can be compiled to low level languages which makes it as performant as low level languages with the hastle of strong typing. Jupyter is a tool that makes learning python lot easier and effective. It is the perfect tool which combines both documentation and IDE in a single package unlike online tutorials which lack this flexibility. Users can read the required documentation and implement it right there which makes the learning more effective. Therefore, we want to convert the current YANN tutorials into Jupyter Notebooks to ease the learning curve for new adopters.

Deep Learning toolkits use a complex nested coding paradigm. Therefore, they need to be properly tested because small mistakes can wreck havoc at the end. Neural Network are a highly dynamic field with a huge number of researcher proposing new techniques every other day which should be incorporated into the tollbox without breaking the existing functionality.

2 BackGround

2.1 Jupyter Notebook

Jupyter notebook is a free and open source web application that allows users to create files that contain live code, explaination and equations in a single page accessible by web browser. It uses the python environment and libraries from the local machine and shows the output right there in the document. The whole document is divided into different cells where each cell can be an explaination or the live code. Creator of the notebook must choose the type of cell for each cell so that the notebook will change the functionality of that cell.

Jupyter notebooks provide rich markdown functionality to present the explaination in a consumable format. Creators can use markdown tags to emphasize the important words, code, show images, videos and write equations using latex.

Jupyter's code cells can be used to write and execute code from the notebook. If the user has multiple python environments on the local machine, he can explicitly select One; which will be used to execute code.

• *Anil Kumar Motupalli ASURITE:amotupal E-mail: amotupal@asu.edu.*

• *Buddha Puneeth Nandanoor ASURITE:bnandano E-mail: banandano@asu.edu.*

While editing code in the livecode cells, jupyter provides code suggestion when a user presses tab. When document is created, it saves all the explainations, code and results after executing that code. When the user downloads and opens that document again, he can edit the given code and experiment with it which gives both curated and independent learning experiences.

2.2 Unit tests and coverage

The functional correctness of a code can be evaluated using unit tests instantantly and in effective manner. It is difficult to ensure that new code is not breaking excisting code. Also, to ensure functional correctness of the new code. This will help developers to easly track what changes will cause what effects and where to fix them.

We have a complex code, there might be chances that certain portions of code won't be executed at any given scenario. There is no point to have such code, to make sure we don't have any such code, we need to run test suite on our code and need to calculate code coverage. In this project, we are planning to achieve maximum possible code coverage for all the functional code.

3 Tasks involved

Righnow yann's documentation is in nepolean style docs that are automatically generated. We are planning port them to Jupyter notebooks with proper explainations and figures. Along with these tutorials we will write a tutorial that demonstrates how to use jupyter notebooks.

YANN is having only few test cases covering activation functions and package imports. We are looking forward to develop unit tests for all the functional code which covers core, layers, modules and special packages of YANN. We are using pytest to write unit tests. We will be using coverage.py to perform coverage.

4 Project Timeline

Task	*Deadline*
Midterm Report	April 8
Converting tutorials to notebooks	April 15
Test cases	May 28

5 Task Distribution

We are planning to divide jupyter notebooks and unit tests between both of us. Once Jupyter notebooks are done, together we will work on remaining unit tests and code coverage. We will we interacting with Ragav Venkatesan on weekly basis and will take continuous feedback from him.

6 Execution

YANN is using Travis CI for code build. All the unit tests and coverage are liked with to Travis. So, whenever there is a commit to Git, Travis will run unit tests and coverage and generates report immediately.

7 Extensions

If we finish the above tasks before the targeted time we are planning to write a tutorial that demonstrate using pretrained networks with YANN.Once we have unit tests in place, we will be comfortable to make changes to code. At that point, we would like port the code to Python 3.x from 2.7.

8 Relevancy with coursework

As part of this project we are going to visit all the tutorials we worked during this semester as well as we are covering every aspect of YANN. This includes almost all the topics that was discussed in the class as well few new topics.

9 References

RAGAV VENKATESAN. Yann. .

ARBUCKLE, D. 2010. Python testing. .

BERGSTRA, J., BASTIEN, F., BREULEUX, O., LAMBLIN, P., PASCANU, R., DELALLEAU, O., DESJARDINS, G., WARDE-FARLEY, D., GOODFELLOW, I. AND BERGERON, A. 2011. Theano: Deep learning on gpus with python. In *NIPS 2011, BigLearning Workshop, Granada, Spain,* Anonymous Citeseer, .

RAGAN-KELLEY, M., PEREZ, F., GRANGER, B., KLUYVER, T., IVANOV, P., FREDERIC, J. AND BUSSONIER, M. 2014. The Jupyter/IPython architecture: a unified view of computational research, from interactive exploration to communication and publication. In *AGU Fall Meeting Abstracts,* Anonymous , 07.