

Understanding Deep Learning Through Visualization

Xuan Zou



Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2017-84

<http://www2.eecs.berkeley.edu/Pubs/TechRpts/2017/EECS-2017-84.html>

May 12, 2017

Copyright © 2017, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

University of California, Berkeley College of Engineering
MASTER OF ENGINEERING - SPRING 2017

EECS

Data Science & System

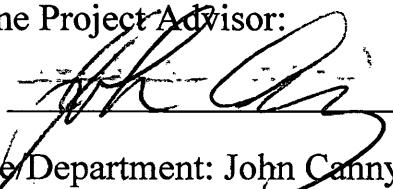
Understanding Deep Learning Through Visualization

Xuan Zou

This **Masters Project Paper** fulfills the Master of Engineering degree requirement.

Approved by:

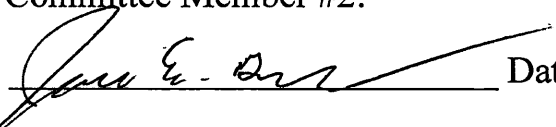
1. Capstone Project Advisor:

Signature: 

Date 5/11/17

Print Name/Department: John Canny/EECS

2. Faculty Committee Member #2:

Signature: 

Date 5/8/17

Print Name/Department: Joseph E. Gonzalez/EECS

Understanding Deep Learning Through Visualization

Xuan Zou
zouxuan@berkeley.edu

Department of Electrical Engineering
and Computer Science

Advisor: Prof. John Canny, canny@berkeley.edu

Executive Summary	3
Chapter 1: Technical Contribution	4
1.1 Inspiration of Project	4
1.2 BIDMach's Architecture	5
1.3 Advantage Over Other Platforms	8
1.4 Work Breakdown	8
1.5 BIDViz's Architecture	9
1.6 Libraries and Tools	10
1.7 User Interface	12
1.8 Future Improvement	18
Chapter 2: Engineering Leadership	19
2.1 Context Introduction	19
2.2 Project Management and Software Engineering	19
2.3 Industry Analysis	21
2.4 Marketing Strategy	23
2.5 Engineering Leadership Summary	25
Conclusion	26
Reference:	27

Executive Summary

The project our team worked on is called “Understanding deep learning through visualization”, which constructed an interactive platform that uses charts to visualize the deep learning training process and allow users to customize their own metric.

Our team developed a BIDViz platform, which integrates Play server with BIDMach library, meanwhile we construct a web page as the user interface, on which the users could view the charts and interact with the model. Han’s paper will explain the detailed architecture of our server, Jingqiu’s paper will illustrate the Messenger part of our platform and my paper will talk about the user interface part of our product.

In chapter 1, I am going to discuss the inspiration of our work, the related work, the work breakdown of each team member, the interface of the system, my contribution in the web front end part of the project and the future improvement I plan to do.

In chapter 2, the paper talk about the the context of our project, which consists of three parts, project management, industry analysis and marketing strategy.

Chapter 1: Technical Contribution

1.1 Inspiration of Project

Deep learning, which is also called Neural Networks, is a powerful tool to explore knowledge and rules from data and has been applied to various fields like natural language processing, image recognition and so on. However, because of the complexity of the training strategy as well as the limit understanding of the loss function and optimization method, tuning the parameters for neural networks has become a “black art” (B. Jiang and J. Canny 2017). Moreover, for deep neural networks, it is common to take several hours or days to train the model. Therefore, it is extremely expensive to stop the training model and restart it if we want to add a new metric statistics or change the parameters of the model. All in all, there exists plenty of problems of current deep learning practice. Firstly, it is time and energy consuming to find the best and most appropriate parameters for current problem. Secondly, it is hard to tell the trend and effect of changing any parameters, which means we usually lose our direction when tuning the model. At last, once we get a satisfying result we cannot know which parameter helps during the training and cannot give a reasonable explanation to the result.

Our project is called “Understanding deep learning through visualization”, which is intended to help to solve the above problems. We developed a system called BIDViz, which was based on the existing deep learning library BIDMach and the matrix library BIDMat. BIDViz provides a web page that contains statistics charts of the training model and allows user to customize their own charts. Meanwhile, it allows user to dynamically change the parameters of the model and pause or resume the training process. Our project tends to help data scientists and researchers by providing following functions:

1. Visualize statistics as streaming and dynamic charts. The major part of our web page is the charts that represents the statistics of the model. By looking at the training statistics history,

researchers could conclude whether the model is correctly implemented, what is the trend of the change of one hyper-parameter could lead and whether the training has converged.

2. Customize the charts by using simple and various API. Besides providing several predefined charts, the system also allow users to program their own charts to see other statistics. For instance, the training loss and prediction accuracy is the most commonly used charts for people to see whether the model is on the right track, while for some specific problems, training may stall for plenty of reason, like vanishing gradient or stoping at the local maximum. In this case, researchers may define new query and statistics charts to locate and diagnose the problem.

3. Tune the parameters and variable of the model without stopping the training process. Because the training of neural networks could be time-consuming, it is unrealistic to stop the training every time when we want to change the model parameters. BIDViz provides the function to dynamically adjust the model variables and create new dependency for parameters, while the training process will not be affected.

1.2 BIDMach's Architecture

BIDMach is a deep learning library developed in UC Berkeley BID Data laboratory, it consists of some efficient model optimizers as well as various strategies. BIDMach is currently the fastest framework for a lot of commonly used deep learning algorithms. According to the benchmark result, on a single GPU node, BIDMach outperforms other cluster systems which runs on hundreds of nodes (github.com/BIDData/BIDMach 2017) . Moreover BIMach streams data from disk and has no limited regarding to memory. Therefore with a large RAID, BIDMach can run topic models with hundreds of topics on several terabyte of data.

BIDMach data model use BIDMat to provide an efficient and interactive matrix layer. BIDMat is a matrix library as well as an interactive environment that intends to support large-scale exploratory data analysis. It is a sibling system of BIDMach and by working together, user could easily build and run their model on GPU.

BIDMach has a modular design intended to make it easy to create new models, run with different data sources, and take advantage of performance measures that are optimized in training. A graphic of BIDMach's architecture appears in Fig. 1 (github.com/BIDData/BIDMach/wiki/BIDMach's-Architecture 2017).

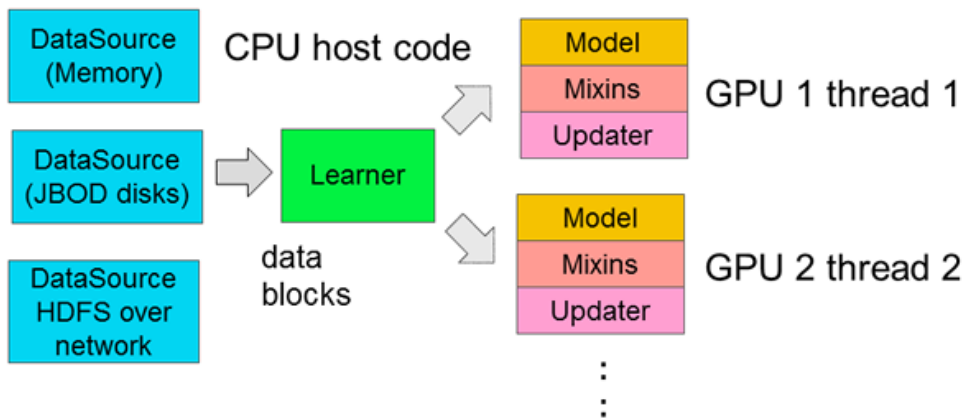


Figure 1. BIDMach's architecture

The elements of the architecture are:

1. Datasources. Datasources provides mini batch of data for processing. The datasource could have various format like an array in memory, file system on disk, or an HDFS source, which means it can be easily integrated with different system.

2. Learner. Learner mainly takes charge of data movement, training and synchronization of distributed models.

3. Models. Models implement particular learning algorithms. Models support an update method, which often produce a model gradient from the current model and input data block, and a likelihood method which calculate the likelihood of a data model.

4. Updaters. Updaters implement optimization strategies. They consist of different mini batch batch updaters. Updaters take advantage of dynamic windows and decreasing weights to connect the gradient updates with a model.

However, the current BIDMach library does not have any visualization part, which means the scientists and researcher can only interact with the library by the command line, and they cannot see the training process through charts. This is definitely one big drawback for BIDMach compared with its other mature competitors like TensorFlow, which has a visualization part called TensorBoard.

Therefore, we develop our deep learning visualization framework, BIDViz, an extensible toolkit to help machine learning practitioners to interact with the model training process. To implement the functions mentioned above, we use BIDMach as the back end deep learning library, leverages support for just-in-time compilation, code serialization and multi-threading in BIDMach/Scala. We also utilize the Scala Play framework to support high-level HTTP client/server communication with web visualization clients.

1.3 Advantage Over Other Platforms

Currently the most popular machine learning visualization library is TensorBoard based on Google TensorFlow (M.Abadi et al. 2016). TensorBoard is useful to visualize TensorFlow computation graph, which helps debug if the model is built as expected. In addition, users could use TensorBoard to visualize some statistics to monitor machine learning training process. We want to argue some key features that our system outperforms TensorBoard:

1. TensorBoard is a separate process from the running TensorFlow training process. Those two processes do not share the memory space, which makes interaction between these two system limited. For example, as mentioned above, such system cannot intervene machine learning training process to fix suboptimal training performance. However, our system puts training and visualization in the same process which gives visualization platform direct access to training system.
2. TensorBoard has the computational graph structure, which is hard to add additional evaluations, such as debugging metrics, into the graph once training starts. Although TensorBoard takes care of the plotting functions, users must explicitly write code to add those calculations into summary operators before training. Our system allows users to add and delete metrics debugging information anytime during the training processes.

1.4 Work Breakdown

Our project can be divided into three parts. Jingqiu takes charge of the Messenger part, which is how the metric statistics system goes and the server connect to the existing deep learning system. Han builds the web server via Play framework on Scala, and establishes the communication protocol between the front end system and the server. These two parts is the back end system of our project, which is essential for a robust deep learning system.

I am in charge of the front end part of our visualization system, which is a web page that provides the charts of statistics and the interface for users to interact with the system. For the front end part, I need to communicate with the server which provide interface for front end to get and send data. Meanwhile it should allow new users who are not familiar with BIDMach or deep learning to easily use this tool and help their work.

1.5 BIDViz's Architecture

A BIDViz application consists of 3 modules, illustrated in Fig 2. First is a deep learning library that provides the training model for our application, in here we choose BIDMach. Second is the server, which consists of channel that observes the current training state, and a web server that interacts with the user. The last is the web page that user can interact with.

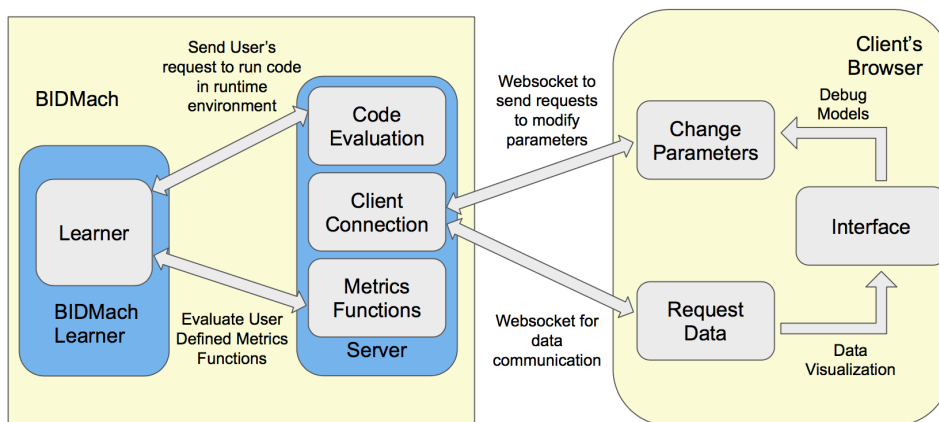


Figure 2. BIDViz application

In the web page part, the first thing is to build connection with server through Play framework, the code is like:

```
this.endPoint = "ws://" + window.location.host + "/ws";

if (this.websocket !== null) {
```

```
        this.websocket.close();
    }

    this.websocket = new WebSocket(endPoint);

    this.websocket.onmessage = this.onmessage.bind(this);

    this.websocket.onopen = this.onopen.bind(this);

    this.websocket.onclose = this.onclose.bind(this);

    this.websocket.onerror = this.onerror.bind(this);

    console.log("websocket connected");
```

Then I create a function call *addPoint*, which is the callback function of *websocket.onmessage*, *addPoint* can set the new point in to the charts, so that it shows the data streaming in the charts. For different types of charts, they need to provide the same interface of *addPoint*, so that other functions could update the charts via the uniform API. For other parts of the system, Han's paper provides a detailed illustration for the back end architecture and designed of the system. Jingqiu's paper talk about code evaluation and message part of the system.

1.6 Libraries and Tools

In order to construct a elegant and useful web page, I utilize plenty of open source libraries. Firstly, I use Bootstrap as the CSS library. Bootstrap a widely use framework which provides a uniform and modern style for HTML elements, which save time for the developers to stylize the web page.

Secondly, I use jQuery to add interactive functions to the web page. jQuery is a fast and small JavaScript library which offer plenty of functions. jQuery provides a wrapper API and make it much easier to traverse the HTML document, manipulate DOM object as well as provide ajax support compared with native JavaScript (api.jquery.com 2017).

Thirdly, for our project, the most important feature is the charts that visualize the training process. Therefore, I need to use a visualization library for our project. After doing some research of the existing visualization library, I found there are plenty of candidates, such as Chart.js, D3, Highcharts, Google chart etc. Finally I decide to choose between D3 and Highcharts. Highcharts is a charting library written in pure JavaScript, which offers the feature to add interactive charts to web application (www.highcharts.com/docs 2017). D3 is a JavaScript library that provides powerful visualization techniques by directly manipulate DOM objects. Finally I decided to use Highcharts as our visualization library, because of several advantages over D3. Firstly, in Highcharts a chart object can be completely expressed as a Json object. To create different kinds of chart, all you have to do is borrow a config object from one of the demos and change it according to your needs. Secondly, Highcharts provides a lot of default setting and parameters which is convenient to use and meanwhile, if you want to customize the charts, it is also easy to modify the parameters. However, regarding to D3, even though it is more powerful to build some highly customized charts, it require us to manipulate SVG in a very low level, which is difficult to learn and use (github.com/d3/d3/wiki 2017). To sum up, Highcharts is more aligned with one principle in software engineering, “Just Enough”.

Another important function is to dynamically modify the model during training. I provide a function in the web page to view the parameters as well as changing them. Showing the parameters is quite easy, Bootstrap provides table component to build a beautiful table. However, the table is not editable and is hard to listen the change of table item. Therefore I utilize another JavaScript library called EditableTable. By utilizing EditableTable component, the web page can listen to the change of the parameters table and check the validation of the change.

The above are the libraries I used during development. Thanks to the open source community, nowadays the programmers have so many frameworks and libraries to choose and avoid building the same wheel again.

1.7 User Interface

Now I have finished the user interface, the web page of our system. Once BIDViz is set up in the training script and the training script starts running, users can use web browsers to access the system interface through server IP address with a pre-defined port number. An example interface is in Fig 3.

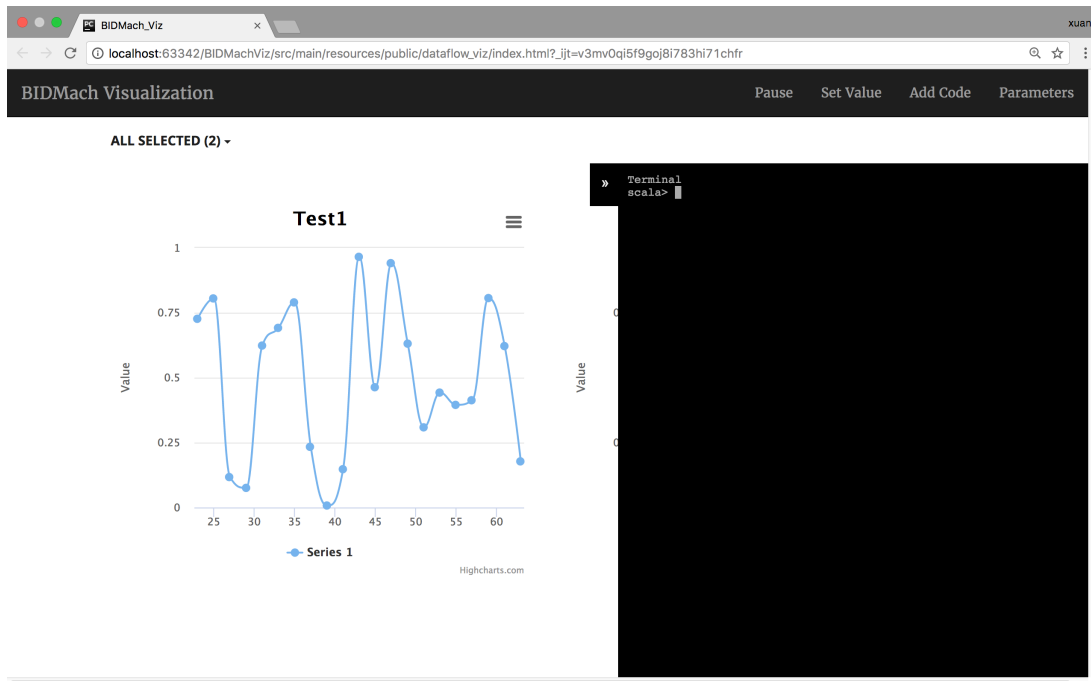


Figure 3. BIDViz main screen

The interface provides simple working environment for the user.

1. Basic charts. As demonstrated in Fig. 4, most of the space on the web page is for rendering plots for user-specified metrics. The page has two figures per row which update simultaneously. In addition, there is a selection bar at the top left of the page which allows users to select the charts they want to see in the current browser while hide others. Specifically, it is very convenient to only show two figures when users like to compare two different metrics.

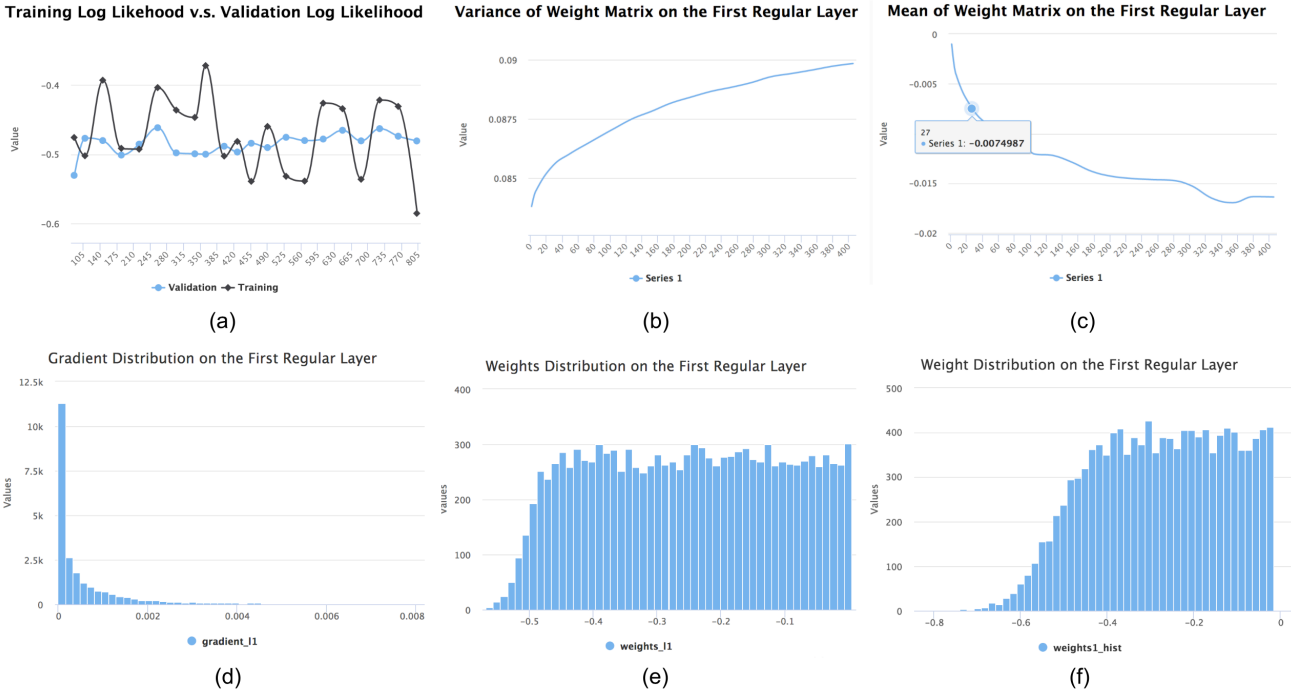
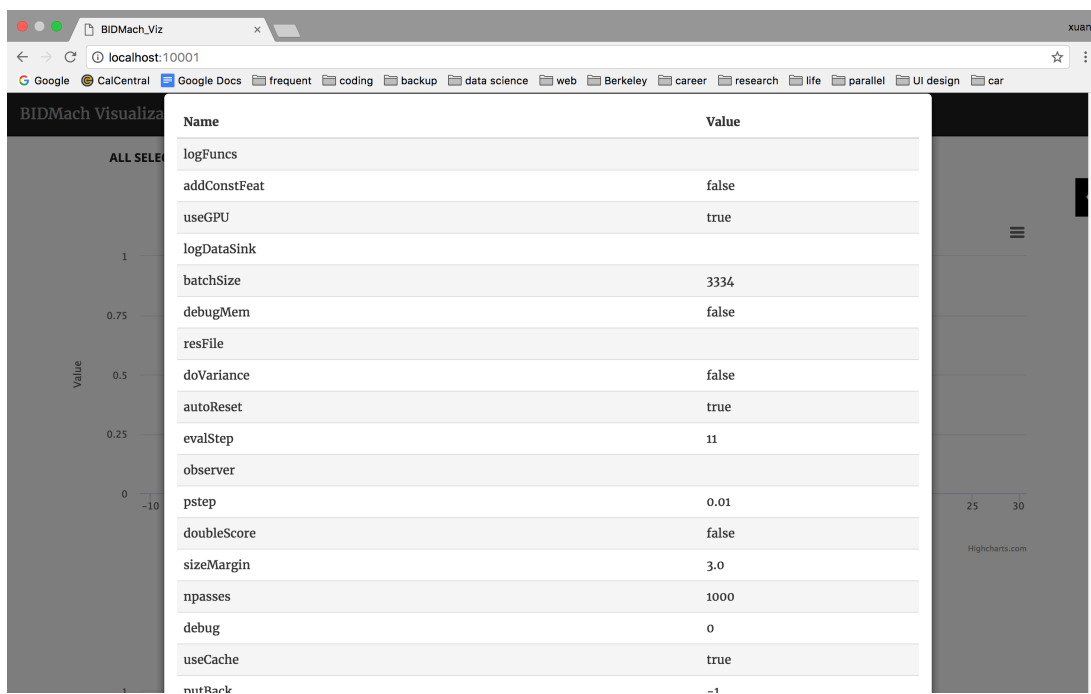


Figure 4. Charts defined by user

2. Built-in terminal. As showed in Fig. 3, the right side of the page, we create a floating window that can slide out to become a terminal. It is often the case that users know what they want to look at, but they don't actually remember which fields in the model that they should use. Having a terminal which evaluates users' command in real-time training environment can help users better write customized code for metric logging, and find other information in the

model that cannot be captured through logging and plotting. In addition, such a terminal is also able to capture run-time errors in the training process.

3. Pause/Resume training button. The Pause/Resume button is used for directly controlling the back-end training process. It can temporarily pause the training progress, or resume the current training progress. By stopping the current process and taking a close look at the current model parameters, statistics and performance through our in-browser terminal, users can effectively make analytical decisions on the next step.



Name	Value
logFuncs	
addConstFeat	false
useGPU	true
logDataSink	
batchSize	3334
debugMem	false
resFile	
doVariance	false
autoReset	true
evalStep	11
observer	
pstep	0.01
doubleScore	false
sizeMargin	3.0
npasses	1000
debug	0
useCache	true
numBark	-1

Figure 5. Change parameters dialogue

4. Changing Parameters of model. As shown in Fig. 5, the button “Show Parameter” is to print all parameters in the current training model and allows users to change them. As discussed early, after identifying the problem, it is also very useful to fix the model configuration by changing model parameters. For example, when users identify slowly changed training accuracy, they may need to dynamically change the learning rate in the training

environment. Such feature creates freedom for users to update the current model during the training process. Currently, users can modify all model parameters in the model. However, users need to be aware of the consequence of changing each parameter. For example, changing the size of minibatch may not create any difference as minibatch was only read once at the very beginning of the training process.

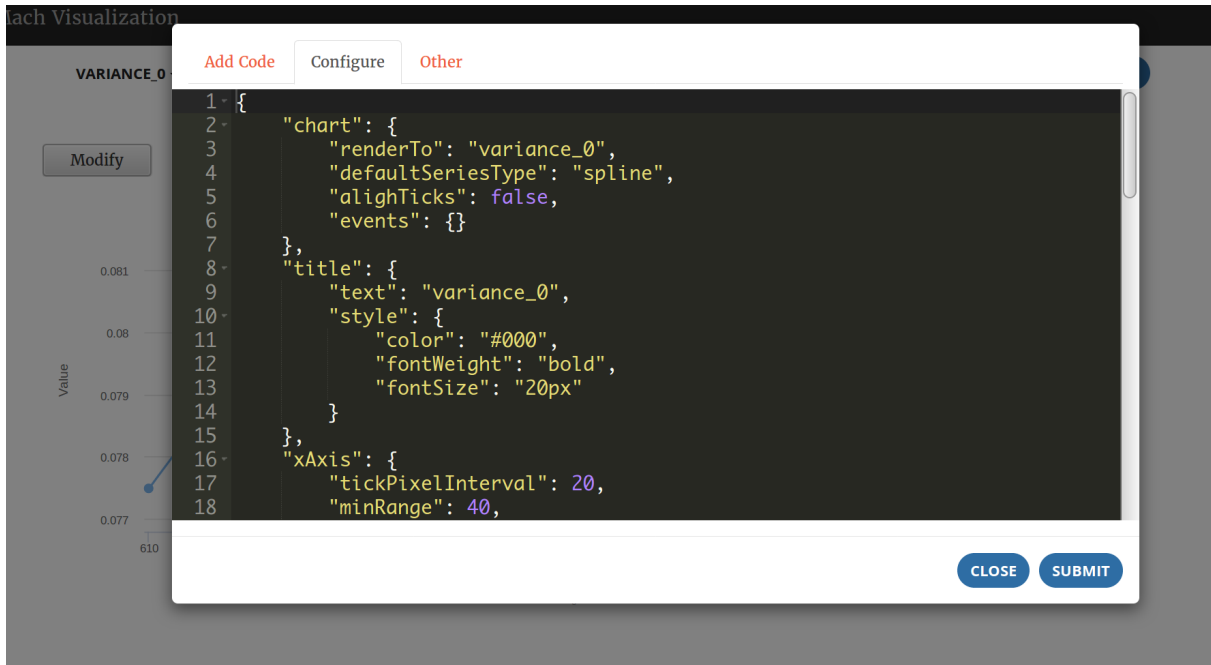


Figure 6. Add code dialogue

5. Add Code and customize charts. One of the most important communication between the user interface and the training model in BIDMach is to dynamically adding new metrics code. We allow users to send in code snippets, under our predefined template, in order to calculate any custom designed metrics during the training process. The code window is demonstrated in Fig. 6. The example here demonstrates a very simple case to calculate the variance of a selected weight matrix. This code snippet will be used to create a function, which will be called on the training model on every iterations, and the evaluation results will

be sent back and displayed in a chart. The custom evaluation function will have access to the current model and the current minibatch data. The output is in the format of a Mat matrix. Essentially, we allow users to use the current model statistics (latest weight vectors, loss, and hyper-parameters), along with the current minibatch data, to evaluate an efficient statistics, such as training accuracy, variance of the weights, or total loss. The results of this call go into a BIDMach dataSink. The dataSink is an abstraction which can be specialized to save to disk, or more commonly to stream blocks of data over http to the real-time plots in BIDViz. The code snippet gives the freedom to define their own metrics evaluation function, which can be widely shared in the community. After users input the code in the window, users need to name their code and specify which kinds of charts that should be plotted out. Currently, our system supports three different kinds of charts: line chart, column chart and scatter plot. Line chart is used to capture the change of a specific metrics throughout the time and show the trends during the past period. It is the most common chart that is used in visualization. In addition, we also support to plot multiple lines in one graph for comparison purposes. Column chart is usually used for histogram to capture the distribution of the input data. And scatter plot can be used to simply show the input data in two dimensional space, which are very useful to illustrate the embeddings.

With the emerging of different research problems as well as various environment. The charts powered by HighCharts sometimes may not satisfy the requirement. Therefore, I provide a uniform interface for user to use while offer choices for different back-end graphic model. Currently, we provide three options to render the graphs, including C3 (c3js.org/reference.html 2017), Vega-Lit (A. Satyanarayan 2017), and HighCharts. In all those three libraries, graphs are defined simply by a JSON file which contains all necessary configurations of the chart. Users have the freedom to choose their favored graphic backend. In Fig. 7, we illustrate how to render

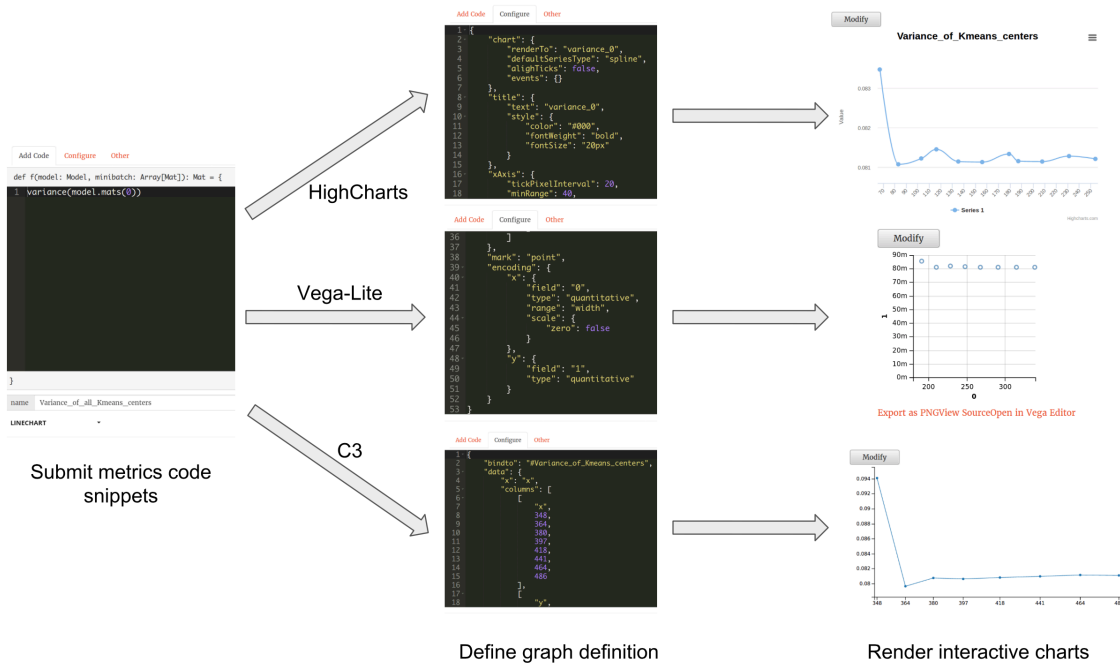


Figure 7. Three different interface, Highcharts, Vega-Lite and C3

three different plots using C3, Vega-Lite and HighCharts based on the same metric code. The panel in the middle shows the graph configuration for each library and the panel on the right shows the results.

Once the metrics are submitted, along with the type and backend of the graph, our system starts rendering the real-time statistics evaluated from the training processes onto an available screen slot. The graph is generated using our default configuration. However, users can click the “Modify” button on top of each chart to change the metrics code or plot configuration. For example, users can change the existing HighCharts configurations to change how the smoothing is done when a new point is added to the chart. Those configurations must be compatible with the JSON definition of each library. Having user-defined graphics grammar, our system provides flexible interface to render any interactive charts that users may be interested in.

1.8 Future Improvement

After finished the above parts of the web system, the main body of our system is finished, then I start to think about some detailed designs of our system.

1. When the user add more than 8 charts in the page, the update of page will become slow and abnormal, sometimes even crash. This is because HighCharts uses SVG to draw image of the page, while when there are too many SVG update happens on the page, the browser cannot handle appropriately. My thinking is to change the update solution from SVG to Canvas, which use another update strategy.
2. User may want to see the previous training data, however, with our current system, the data in previous time will be discarded, so the next function I am going to add is the maintain all the statistics data and allow user to see the previous data.
3. Currently the system did not provide function to record and download the training history. There are some scenario that user may want to pause the training process and shut down the machine. In this case, we would better provide function for user to record the current training process, download current statistics of model and continue when the machine restart. TensorBoard has offer this function and is widely used. This function is particularly helpful for some large scale problems.

Chapter 2: Engineering Leadership

2.1 Context Introduction

As an important branch of machine learning, deep learning provides amazing human like intelligence, such as image captioning or translation. However, this requires a massive amount of data and machine time in order to train a reasonable deep learning model, and a large amount of hyperparameters associated with the training itself. Any mistakes in model design or hyperparameters will result a big loss in both time and energy, as the training needs to be run from scratch after making required modifications to the parameters or the code. Our project is to design and create a tool in which scientists can babysit the training process, inspecting the current result and parameters, and even live-tricking or experimenting with different configurations without stopping and starting the training from scratch. In this way, not only we can detect potential model incorrectness early on and save some time and energy, we can also discover the configuration that yield most efficient model.

2.2 Project Management and Software Engineering

The goal of Project Management is to ensure maximal throughput and efficient usage of team's engineering hours to deliver a project. There are many aspects that could affect the the team's throughput. Some of those are social aspects, such as how aligned the team's goal and personal goals of each team members aligns or does the people feel the team as a friendly environment. Other aspects are technical, such as can several tasks be carried forward without conflict, enabling parallelism among team members. This section will discuss only the technical aspects, to show how following a modular design pattern will enable team members to work more efficiently, and how does this design also fits many software engineering goals.

Software Engineering is a systematic approach to the entire lifecycle of a software systems, such as design, implementation, testing and maintenance (Laplante, 2007). As opposed to the concerns of *computer science*, computer engineering concerns the adaptation of a system to a series of changing and vaguely defined requirements, instead of just creating a solution for a particular well defined problem. A well engineered software should have the following properties:

1. It should be easy to add new features or modify requirements without massively affect other existing features, in particular, this means that we can add features by *adding* code, instead of modifying code.
2. When the system behaves unexpectedly, it should be easy to pinpoint the few places that need to be fixed. Both of them is essential to ensure maintainability of the system.

To achieve both the goals of software engineering and project management, we have followed these principles:

1. Divide the entire system into several independent *modules*.
2. Each modules can communicate to the others only through a well-defined contract, or *interface*. Interface can be expanded, but never modified or removed.
3. Each of modules are free to be modified, without changing the contract.
4. Each module is owned by one team member, though any member can work on any module.
5. A new feature is implemented by defining additional interface each module need to support, and then implemented in parallel.

This design allows each module to be worked independently and in parallel, the owner of a module is responsible to ensure it abides the predefined interface through change, and also serves as the go-to person for questions when other member is working on this module. It also

have the additional benefits of allowing each module to be unit tested independently, allowing less rooms for errors or bugs.

We have divided our project in 4 modules. The first one is *core* : core module is the existing machine learning library (BIDMach) that we are based on, this part is developed and maintained by Prof. Canny. The next one is *channel*: channel observes the event happened inside of core, and send them out. This module follows the “Observer Pattern” specified in the “Design Patterns” book by Gamma et. al. (2004). The channel will observe events, and compute statistics on those events, and finally it will send it out to the last module, *web interface*. The web interface itself is complicated system, so it then divided into smaller modules following the “Model-View-Controller” pattern from the same book by Gamma et. al. (2004). This design allows each components to evolve in their own, also allows prof. Canny’s other projects to continue on the same code base, without affecting each other.

2.3 Industry Analysis

Deep learning is changing the world. AlphaGo beat Sedol Lee, Uber is testing its self-driving car, people ask Siri “what’s the weather today?” before dressing up. However, in the early days, AI research community disregarded the potential of neural networks. For example, Marvin Minsky et al. (1969) in the book “Perceptron” pointed out many drawback and limitation of neural nets. This situation has not improved over years until the popularity of Internet led to a stage of Big Data. People follow topics they are interested in on Twitter, rate the restaurant they have been on Yelp, and share pictures they love on Instagram. All those kind of activities make the internet a giant pool of data. Unlike traditional way of telling the machine what to do by hard coding, machine learning takes the approach to train the machine from data and expect it to make correct prediction on data after training. Therefore, the more data we use to train the model, the

more “experienced” the machine will be, and this highly increases the training accuracy of the model. For example, in the research of generating image caption using deep neural networks, Vinyal’s team used images uploaded to Flickr.com, and this dataset is as big as 30,000 images. (Vinyal et al. 2015,5) In addition, they also used dataset from MS COCO of over 80,000 images and corresponding picture descriptions. (Vinyal et al. 2015,5)

In recent years, many tech giants in Silicon Valley join a so called “Race of AI”. Sundar Pichai, Google’s CEO, claims that we are moving to a AI-first world (D’Onfro, 2016) in Alphabet’s Q1 earnings call. Apple, Microsoft and Amazon are heavily investing in smart personal assistants, such as Siri and Cortana. Intel acquired three AI-focusing startups in 2016 alone. Companies invest tremendous resources in their AI research group, aiming at design better algorithms, build more efficient models to accelerate their product/service quality.

Besides technology firms, deep learning is widely used in other industries, such as financial institutions. Banks build neural nets to provide risk score of a customer based on its multiple data resources such as salary, credit history, etc. Banks and merchants worldwide suffered around \$16.31 billion of fraud loss in 2015 (Allerin, 2016). Deep learning algorithms can be used to predict criminal transaction patterns and distinguish fraudulent activities from normal ones.

The broad application of deep neural networks demonstrates a big need of visualization tools and we will target any industries that uses machine learning algorithms as our potential users. After discussing our potential users, we need to further analyze any potential competitors. We believe TensorBoard will be our major competitor. TensorBoard is a visualization tool that comes with TensorFlow – a widely used machine learning library. TensorFlow generates summary data during training process and TensorBoard operates by reading those data. While both tools have

same operation mechanisms, our tool enjoys some features that are essential to a data scientist. First of all, we allow users to add additional visualization requests during training process, while TensorBoard has to stop the training and add logging data. Second, we enable users to tune hyperparameters while training to dramatically save training time. At last, our platform supports more flexible charts type, which provide flexibility of visualization.

2.4 Marketing Strategy

Our project is about understanding deep neural networks through visualization, and our market will focus on the fields that utilizing neural networks to do data analysis, pattern recognition or image classification work.

The neural networks have plenty of applications in all kinds of fields and have already been integrated into many softwares and devices. One of the most straightforward application is using neural networks to recognize characters. For example, the postman categorizes the letters according to the post code on the envelop, by developing a software that integrated with neural networks, it could distinguish the digit with high efficiency and accuracy, which can save the post office bunch of money and relieve human from this boring work. In order to achieve good performance and accuracy of this application, we need to develop and tune the neural network, in which the product our project can help a lot (B. Hussain 1994:98).

Another application of neural networks that may not be obvious but is much more profitable is in the finance area. According to some companies such as MJ Futures, neural networks have been touted as powerful and easy to use tools to predict the stock market. By integrating the neural network prediction method, the company claims around 200% returns over 2-year period. Meanwhile the software integrated with neural networks are easy to use. As technical editor

John Sweeney said "you can skip developing complex rules and just define the price series and indicators you want to use, and the neural network does the rest." (Artificial Neural Networks 135).

The idea of stock market prediction is not new, of course. Business people always attempt to anticipate the trend of the stock market by their experience in some external parameters, such as economic indicators, public opinion, and current political climate. While with neural networks, software is able to discover the trends that human might not notice and use these trends in the prediction.

Our project is about understanding deep neural networks through visualization, so the outcome of our research is a software tool that could monitor and analyze the training process of neural network. The software can be used to improve the performance of neural networks and help tune the parameters and architecture of deep neural networks. Therefore, our software can play a role in the areas that require well-architecture neural network.

In order to commercialize our product, we have three steps plan. The first step is to present the demos and results in some famous communities, in order to attract the attention of academia field. This can help improve the fame of our product and gain the acknowledgement of experts. The second step is to build a website for our product. We will allow users or companies to freely download our software but with a time-limited trial, which is a common strategy of many softwares. After the period of trial, they have to pay to acquire membership to continue use. The last step is after we have got a certain amount of users and further refined our product, we will try to get contact with some big companies, to promote our product and provide customized

service for them. Through the cooperation with big companies, our product can get advice from industry and further get improved.

2.5 Engineering Leadership Summary

Based on our project management strategies, we efficiently keep track of project progress and make adjustments when necessary, and this ensures on time high quality deliverables. We get good understanding of industry needs and requirements, as well as potential users from the industry analysis. Through marketing strategy analysis, we have a better understanding towards the potential market as well as the steps to promote our product.

Conclusion

Deep learning is currently one of the most flourishing fields in computer science, and our capstone project offers me the chance to get in touch with as well as make some contribution in this field. The BIDViz system provides a interactive way to visualize the training process which gives deep learning practitioner a new approach to understanding their model. In the future, we would like to extend this tool to connect with other popular platform like Keras or Caffe. Moreover, we hope our product could inspire more innovative ideas and help deep learning make some difference in more fields.

Reference:

- [1] B. Jiang and J. Canny. *Interactive machine learning via a gpu- accelerated toolkit*. In *Proceedings of the 22Nd International Conference on Intelligent User Interfaces, IUI '17*, pp. 535–546. ACM, New York, NY, USA, 2017. doi: 10.1145/3025171.3025172
- [2] <https://github.com/BIDData/BIDMach/wiki/Benchmarks>
- [3] <https://github.com/BIDData/BIDMach/wiki/BIDMach's-Architecture>
- [4] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. J. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. G. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. A. Tucker, V. Vanhoucke, V. Vasudevan, F. B. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. *Tensorflow: Large-scale machine learning on heterogeneous distributed systems*. *CoRR*, abs/1603.04467, 2016.
- [5] jQuery API document, <http://api.jquery.com/>
- [6] HighCharts document, <http://www.highcharts.com/docs>
- [7] D3 introduction and demo, <https://github.com/d3/d3/wiki>
- [8] C3 reference, <http://c3js.org/reference.html>
- [9] A. Satyanarayan, D. Moritz, K. Wongsuphasawat, and J. Heer. *Vega-lite: A grammar of interactive graphics*. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*, 2017.
- [10] E. Gamma, R. Johnson, R. Helm, J. Vlissides. (2004) “*Design Patterns: Elements of Reusable Object-Oriented Software*”. Indianapolis, Indiana: Addison-Wesley.
- [11] M. Minsky, S. Papert. (1969) “*Perceptrons*”. Cambridge, MA: The MIT Press.
- [12] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. (2015) *Show and Tell: A Neural Image Caption Generator*. [Arxiv.org/pdf/1411.4555v2.pdf](https://arxiv.org/pdf/1411.4555v2.pdf)

[13] J. D'Onfro. (2016) *Google's CEO is looking to the next big thing beyond smartphones*

<http://www.businessinsider.com/sundar-pichai-ai-first-world-2016-4>

[14] Allerin - *How is deep learning being used in the banking industry?* [https://www.allerin.com/](https://www.allerin.com/blog/how-is-deep-learning-being-used-in-the-banking-industry)

[blog/how-is-deep-learning-being-used-in-the-banking-industry](https://www.allerin.com/blog/how-is-deep-learning-being-used-in-the-banking-industry)

[15] B. Hussain and M. R. Kabuka, "A novel feature recognition neural network and its application to character recognition," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 1, pp. 98-106, Jan 1994.

[16] *Artificial Neural Networks: Applications in Financial Forecasting* pp.135