Understanding Deep Learning Through Visualization

Jingqiu Liu

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2017-85
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2017/EECS-2017-85.html

May 12, 2017

University of California, Berkeley College of Engineering

MASTER OF ENGINEERING - SPRING 2017

EECS

Data Science

Understanding Deep Learning Through Visualization

Jingqiu Liu

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

Jingqiu Liu
jingqiu_liu@berkeley.edu

Department of Electrical Engineering
and Computer Science

Co-authors: Xuan Zou, zouxuan@berkeley.edu
Han Qi, qihan@berkeley.edu
Chen Tang, chen.tang@berkeley.edu

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

Executive Summary

Our project aims at building a deep learning visualization tool which can supports real-time visualization and interactive user-tool communication. We have completed building the tool and conducted some example studies. The chapter 1 of this paper will focus on introducing the techniques we used to realize code evaluation. And the chapter 2 will discuss three aspects related to this project: Project management and software engineering, Industry analysis, and Marketing strategies.

Chapter 1. Technical Contribution

1. Introduction

1.1 Project background

As an important branch of machine learning, deep learning (a.k.a. deep neural networks) provides human like intelligence which allows machine to appropriately describe a given image or correctly translate Chinese sentence to English. We describe the training of a deep neural networks as a process of inputting preprocessed data into a selected model, waiting for the machine to compute for days and weeks and getting the final model for specific use. If the model does not work well, we may reprocess data in different way, tune model hyperparameters, change update rule or even the model itself, and retrain the model. For example, we use millions of images and their corresponding category as input, and we train the Convolutional Neural Network based on those data and get a final model which has the ability to tell you (we call it “prediction”) whether a given image is a dog or a ship.

The training process is critical to a successful model. On the one hand, each model has its strengths and drawbacks, and is usually used to solve a specific kind of problems. The selection and design of initial model directly affects the prediction accuracy of the final model. That is why we consider correct hyperparameter tuning as an important aspect of a training process. On the other hand, the time scale of a training process is usually in days, weeks or even months. It is a pain point for most data scientists when they find out their final model does not work as expected after training for a long time. Our project is aiming at building a visualization tool to relief this pain.

1.2 Project Goals and Tasks

One efficient way to understand, monitor and debug the deep neural network training process is to see how the important statistics change during the process. Indeed, people in the field of study already recognize the importance of training visualizations, and a lot of excellent visualization tool demonstrates this fact. For example, Google embedded TensorBoard into TensorFlow to support visualization (TensorBoard, 2017). Compared to other visualization tool, one unique feature of our product is the interactive communication between users and the tool. Our visualization tool will draw graphs to show the changing pattern of desired statistics, and allow users to access and download those graphs while realizing real-time visualization. Also, we would like to enable users to submit request for adding/deleting specific statistics visualization tasks without interrupting the training process. In addition, we hope users can dynamically tune model hyperparameters based on visualization result while training is still going on.

Our tool is based on BIDMach which is a fast machine learning library written by Scala. BIDMach will handle users' tasks of training different neural networks. Based on the timeline, our main tasks are:

Sep – Oct 2016	Study BIDMach, try to run some training samples to get familiar with the library
Oct – Mid Nov 2016	Build a publisher that will initialize a web browser, publish all required statistics from BIDMach, generate graphs of those statistics
Mid Nov – Jan 2017	Build a bidirectional messenger with which user can push their code as string (request for additional visualization tasks) into BIDMach, and BIDMach can send specific error message to the user if the user input errors out
Mid Nov – Jan	Find a way to enable BIDMach to interpret pushed string as real code that

2017	Scala can run and extract corresponding error message when error occurs
Jan – Feb 2017	Use JavaScript to design a user friendly webpage and draw graphs using published statistics.
Feb – Apr 2017	Improve webpage to contain all the required visualization graphs, a message box to send further visualization request, as well as a table to display current hyperparameter values and support parameter tuning. Conduct user study and improve product based on feedback.

Table 1. Project timelines. This table briefly introduce the project task splits and time of delivery.

Han's paper will mainly discuss potential user requirements of the tool from a broad view, and the design of the publisher and messenger based on those requirements, involving communication protocol comparison and selection, as well as webserver implementation details. Xuan will focus on introducing our current visualization webpage. Topics include basic web system architecture, visualization technique, as well as some visualization charts. And my paper will focus on reasoning why we need the messenger, how do we complete the connection between messenger and web browser from the stand point of successful user's code interpretation and error extraction by Scala.

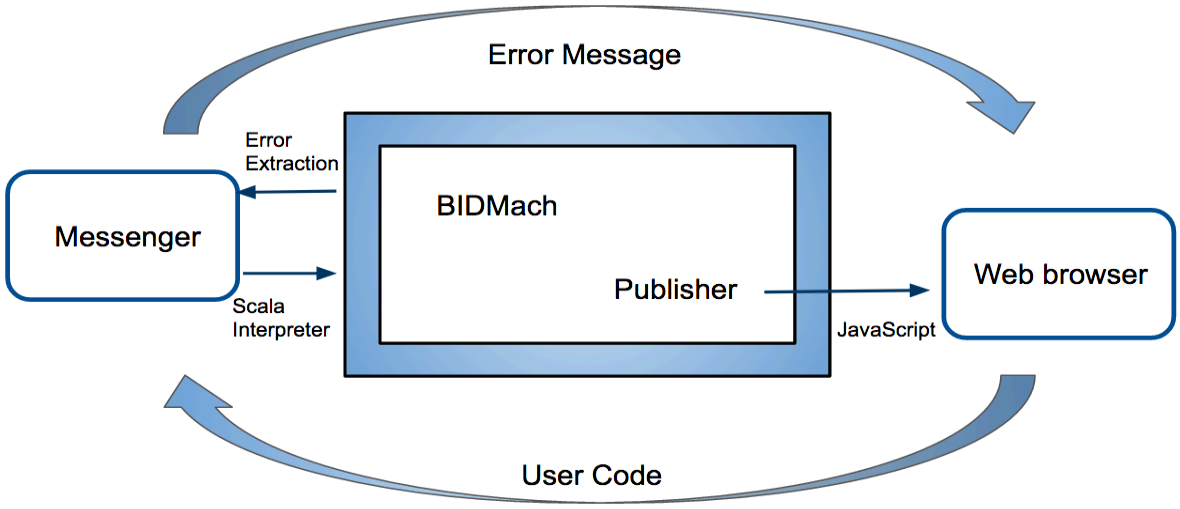

Figure 1. Visualization tool structures and work flow.

Figure 1 briefly shows the whole working flow of our tool. BIDMach generates statistics for graphs while it is training. Publisher grasps the generated statistics and send them to web browser to draw graphs using JavaScript. Users view the graphs in the web browser and send additional requests via the webpage. Messenger receives user code and interpret it as real Scala code so that BIDMach can run. When the code errors out, messenger extract the error message and send it back to webpage.

It is clear from the flow that the messenger is essential as an intermediate between BIDMach and the webpage. Though we can ask webpage to directly send user input to BIDMach, creating a messenger efficiently split work among each part and allows more flexibility in architecture design.

The rest of the paper will discuss different paths we took to realize user code interpretation and error extraction, as well as some future work.

2. Tools

Before we get into more technical details, let us first introduce any tool we use. As mentioned, BIDMach is written by Scala. Scala is an object oriented programming language that is inspired by Java. Compared to Java, Scala has more complex structure, resulting in more concise code. As Hicks mentioned in his article, “In contrast, Scala was created specifically with the goal of being a better language, shedding those aspects of Java which it considered restrictive, overly tedious, or frustrating for developer” (Hicks , 2014).

Since we pass user input as a string, in order to make BIDMach successfully understand user's code and process new requests, we need to find a way to enable Scala "read" and "understand" user input, and evaluate it as real Scala code and call it during training to generate new required statistics. To make the interpreter work, we tried to utilized the Scala IMain package and ToolBox package.

3. Problems and Progress

3.1 Code Interpreter

When BIDViz starts, a training thread will be created and is responsible for running the training loop. A serving thread will also be created and is responsible for communicating with the web browser to handle user request and send out relevant information. Inside the serving thread, we create an interpreter such that each time when new request comes in, the user input will be sent to the interpreter and get evaluated as real Scala code. And the evaluated function will be added to a Map so that it will be called when BIDViz is notified.

When user send new visualization request, we expect the code to construct a class object with methods needed to complete user's task. And this code will be sent into BIDMach as a string. Our engine should interpret this string and evaluate it as a class in Scala so that we can further call the functions defined in the class.

Our first approach is to use Imain package. Imain package has methods called "interpret" and "valueOfTerm". Interpret method will take a string of Scala code as an input, and it will execute this code at the request of the user. ValueOfTerm method will take an object variable as input and return the value of that object. The returned value can be as simple as an integer or as complex as a class with multiple methods and values.

Ideally combining these two methods will give us expected result: we pass a string into the "interpret" function and the string is to assign a function as value to a variable. We then call "valueOfTerm" on that variable and it will return the function for us. We tested it by simply passing a string to assign a variable 'example' with value 1 into interpreter. And we called the "valueOfTerm" method and it actually gave us result of 1, which is the value of 'example'. We then tested it by writing a piece of more complicated code, which is defining a class 'C' and a function 'f' inside the class, and passed it to the interpreter as a string. However, when we try to evaluate the class and run the function 'f', we received an error message indicates that Scala cannot find class 'C'. This is because after interpreting, the class 'C' will be returned as an object with type "Any". And we actually lose the class "C" as type "C".

We tried to fix it by forcing the class type to change from "Any" to "Class C". However, this did not work since the "Class C" type is a user-defined type and it only exists inside the interpreter. If we tried to call it outside the interpreter, Scala will generate an error since it cannot find the class.

Our second approach is based on Scala's nature of how it works. Similar to other compiled language, after a Scala file is created with Scala codes, we need to first compile the file into a bytecode file, and then use Scala to run the bytecode. Instead of using interpreter, we decided to take the approach that consider the user input as a real Scala file, compile user's code in run time and directly call the compiled code.

We first created a classloader for class "C", then used the "compile" method in ToolBox package to compile the string, then instantiated class "C" from the bytecode. In addition, we played a little trick on how we define class "C" in the string. We made class "C" inherited from class "Function[input type, output type]". In this way, "C" not only has type "class C" but also type

"Function[input type, output type]". And the latter type is always valid in Scala. So we instead cast the class to type "Function[input type, output type]" when we instantiate the class and call function 'f'. In this way, we do not need to bother even if "Class C" if not in the environment outside the interpreter. This approach works successfully.

3.2 Messenger Receiving Messages

As described in early page, user types their new request into the message box in the webpage and the web browser sends the commands as a string to the messenger. When BIDViz receives a fragment of Scala code, it will instantiate a function object by placing the fragment into the following template:

```
class ClassName extends Function2[Model, Array[Mat], Mat] {
  override def apply(model: Model, minibatch: Array[Mat]): Mat = {
   %s
 }
}
scala.reflect.classTag[ClassName].runtimeClass
```

Based on the interpreter we introduced above, this function template is then compiled into Java bytecode through Scala's runtime mirror toolbox in the standard library. This toolbox then loads the bytecode directly using the classloader instance of the current running environment. After the new class is compiled and loaded, it is not different from any native Scala function object written beforehand. We will then obtain a *Class*<> object for the newly created class and instantiate it by invoking its constructor reflectively.

The instantiated function object is then placed into a thread-safe HashMap that maintains the mapping between the name (or id) of the metrics to the function object. The compilation and creation of the function object is done in the serving thread, but the invocation of the said function object will be on the training thread, on every pass of the training loop.

Besides using this code evaluation mechanism for creating function objects to be used to compute metrics at each training iteration, the same mechanism is also used to evaluate commands typed into the terminal. When the user types and executes a command, that string is placed inside of a function definition template and compiled then invoked once. The return value of that function is converted to a string using *toString* and send back to the terminal. Since the return value of a function in Scala is just the last expression when no *Return* keyword is present, this works as expected.

For now, the terminal feature is intended to be a tool to inspect the current state of the model, so the functions are evaluated in the serving thread, as we don't expect it to modify the model. We can easily execute those generated function objects in training thread if we intend to use the terminal for modifying the model by creating a thread-safe queue and push the to-be-executed function into it, which can be called and removed when *notify* is called. We do not maintain a separate Scala interpreter as it is not as easy to share variables between 2 isolated JVM environments.

3.3 Error Handling

When user submits a code snippet, or types in a command in the terminal, that line of code could cause compile time or run time errors. For command in the terminal both of those errors are caught as *Exception*s when compiled and executed, and the result will be converted to string through *getMessage* method and displayed in the same terminal.

However, for metric codes, the compile error will be discovered immediately so it is send back through the callback, but the runtime errors, such as *NullPointerException*, will only be discovered when the newly created metric function object is evaluated in the training thread.

When this type of error occurs, we use a new message type *error_message* to send it along with data points. That error message will also be routed to the terminal.

To capture the error message, since we use ToolBox package to compile and instantiate the new function, if the function errors out when running, a ToolBoxException will be generated, and the corresponding error message will be attached to the Exception. Our approach is to simply extract the error message attached using getMessage() function. As a result, if some error occurs and an exception is raised, our program will catch it, extract the error message, and send it to web browser to inform the user.

3.4 Future Works

One important heuristic we base on here is that when a lot of error messages about one error are provided, the top one will be most relevant to the error and it is a good point to start to debug. But we also admit sometimes the latter error messages may also help debugging. Therefore, we will try to use other ways to extract the full error message and display them to the users via web page.

On the other side, we believe that it is better if our tool can have some default visualizations that are commonly used by most data scientist to understand and diagnose their training models. Therefore, we need to do some research on what kind of statistics and what type of graphs are useful.

We currently finished the prototype of the first version of our tool based on our initial design, which should support statistics visualization as well as interactive communication with users through web browser. Our next step will involve user experiments by running different

deep neural network models by ourselves and ask some other students to try our tool. We will further improve our tool based on any feedbacks.

Since there are a lot of outstanding machine learning libraries/frameworks such as TensorFlow and Caffe, we think it is better if we can find a way to make our tool able to receive, understand, and visualize results generated from other libraries. This will be a huge step in connecting our tool to other libraries and make it useable to data scientist who do not use BIDMach to train the model, thus gives the tool higher flexibility.

4. Conclusion

Based on our current progress, we should be able to follow our timeline and produce sound deliverables. Our product will work as an efficient tool that can help data scientists better understand and monitor the deep neural network training process, and save the training time by allowing tuning parameters while training.

Chapter 2. Engineering Leadership

1. 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 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. And 2. When the system behaves unexpectedly, it should be easy to pinpoint the 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 module 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 has 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 is 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. Industry Analysis

Deep learning is changing the world. 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. Online 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. (CB Insights-blog, 2017) 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 is embedded in 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.

3. Marketing Strategies

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 software 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 all-powerful tools in stock-market prediction. It claims 199.2% returns over a 2-year period using their neural network prediction methods. Meanwhile the software integrated with neural networks are easy to use. As technical editor John Sweeney said "you can skip developing complex rules (and redeveloping them as their effectiveness fades) 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 this 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 software. 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.

Reference:

TensorBoard.(2017). TensorBoard: Visualizing Learning. Retrieved from https://www.tensorflow.org/get_started/summaries_and_tensorboard

Hicks, M. (2014). Why Should I Learn Scala? Retrieved from https://www.toptal.com/scala/why-should-i-learn-scala

P. Laplante (2007). *What Every Engineer Should Know about Software Engineering*. Boca Raton: CRC. ISBN 978-0-8493-7228-5. Retrieved 2011-01-21.

E. Gamma, R. Johnson, R. Helm, J. Vlissides. (2004) "Design Patterns: Elements of Reusable Object-Oriented Software". Indianapolis, Indiana: Addison-Wesley.

M. Minsky, S. Papert. (1969) "Perceptrons". Cambridge, MA: The MIT Press.

O. Vinyals, A.Toshev, S. Bengio, and D. Erhan. (2015) Show and Tell: A Neural Image Caption Generator. Arxiv.org/pdf/1411.4555v2.pdf

J. D'Onfro. (2016) Google's CEO is looking to the next big thing beyond smartphones. Retrieved from http://www.businessinsider.com/sundar-pichai-ai-first-world-2016-4

CB Insights – blog. (2017) The Race for AI: Google, Twitter, Intel, Apple In A Rush To Grab Artificial Intelligence Startups. Retrieved from https://www.cbinsights.com/blog/top-acquirers-ai-startups-ma-timeline/

Allerin. (2016) How is deep learning being used in the banking industry? Retrieved from https://www.allerin.com/blog/how-is-deep-learning-being-used-in-the-banking-industry

B. Hussain and M. R. Kabuka. (1994) "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.