BIDViz: Real-time Monitoring and Debugging of Machine Learning Training Processes

Han Qi
Jingqiu Liu
Xuan Zou
Allen Tang
John F. Canny, Ed.

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2017-99
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2017/EECS-2017-99.html

May 12, 2017

University of California, Berkeley College of Engineering

MASTER OF ENGINEERING - SPRING 2017

Department of Electrical Engineering and Computer Science

Computer Science: Data Science

BIDViz: Real-time Monitoring and Debugging of Machine Learning Training Processes

HAN QI

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: Prof. John Canny/EECS

2. Faculty Committee Member #2:

Signature: _____ Date 05/08/17

Print Name/Department: Prof. Joseph Gonzalez/EECS

BIDViz: Real-time Monitoring and Debugging of Machine Learning Training Processes

Han Qi

Project in collaboration with Allen Tang, Jingqiu Qiu, Xuan Zou,

under supervision of prof. John Canny

Executive Summary

Artificial Intelligence is a thriving field with many applications, whether in automating routinary human labors or support basic research such as diagnosing diseases (Goodfellow et. al., 2016). Deep learning, or machine learning with deep neural networks, is particularly successful in providing amazing human like intelligence, such as image captioning or translation, because it has a more general model of the world and makes relatively few assumptions on the world it trying to model (Goodfellow et. al., 2016). However, comparing to the tools for programming and debugging, the tooling for building and learning deep learning models is not as good.

In this report, we present BIDViz, a visualization platform that allows data scientists to visualize and debug his/her model interatively while the model is training. BIDViz is applicable to general machine learning but is oriented toward deep neural models, which are often challenging to fully understand. BIDViz emphasizes dynamic exploration of models by allowing the execution of arbitrary metrics or commands on the model while it is training.

BIDViz is developed jointly by Jingqiu Liu, Xuan Zou, Allen Tang and myself, under supervision of professor John Canny and PhD candidate Biye Jiang. The implementation of BIDViz consists of 2 parts: the design and implementation of the User Interface, which is covered mainly in Xuan's report; and the design of the computing and serving backend, which is covered jointly in Jingqiu's report and this one.

Click here to enable desktop notifications for UC Berkeley Mail. Learn more Hide Idle Xuan Zou Offline Jingqiu Liu Allen Tang BIYE JIANG Felicity Zhao Isla Yang John Canny Joseph E. Gonzalez SHUAI LIU Invited Sijia Teng More 3 of 3,365

Chapter 1

Technical Contributions

1.1 Introduction

Machine learning are powerful and versatile tools for understanding and exploiting data. But it is also the case that users often under-exploit the power of ML and DL because of limited understanding of the loss functions and optimization method [17]. For instance, clustering models like k-Means optimize only internal cluster coherence, not inter-cluster distance or cluster size uniformity. The latter goals are often more important or assumed by the user. Previous interactive machine learning tools like [17] allow users to tailor the loss function as a blend of component losses while visualizing the effects on the component losses. These losses and visualizations must be defined ahead of time, although they could be added or removed to a particular training session.

For deep neural models, training often lasts hours or days. It is extremely expensive to stop and re-start training in order to add diagnostics or controls. Therefore, we propose a system that allows dynamic creation and customization of diagnostics or controls to deep learning models while they are training. To illustrate the main purposes of our system, we demonstrate three scenarios below that are critical to machine learning researcher and data scientist:

1. Introduce code to compute and stream new statistics. During the training process, instead of waiting for the process to complete, practitioners can print or plot useful statistics to understand model training progress. For example, printing out the training loss or validation accuracy is the most common strategy that people use to see if model training is on the right track. Training may stall for a variety of reasons - vanishing or even exploding

gradients, bad operating points (relative to the "bends" in non-linear layers), and "flat" regions on the loss surface. Its difficult to anticipate all the possible pathologies, and ad-hoc querying is often needed to diagnose which is in play. Once a cause is found, further exploration may be needed to localize and solve the problem. These queries may be quite complex, and need to be defined on-the-fly.

2. View these statistics as streaming or dynamic plots. Although it is useful to see the current training accuracy, it is more meaningful to see how much the accuracy has changed in the past few iterations. By looking at historical data, users can conclude whether the training processes have incorrect implementation, suboptimal hyper-parameters or even have converged. Those data are in effect time series and can be conveniently viewed as streaming plots. It is also useful to make other forms of plot, such as histograms, dynamic so that they change over windows of data.

3. Change model parameters or variables by intervening in the training process. The deep learning training process is very time-consuming, and it is very inefficient to diagnose the model by trial and error. If a hyper-parameter such as learning rate is sub-optimal, shutting down the current process and restarting wastes the work invested in training to that point. Once problems with the model are identified through streaming/dynamic plots, users will often want to directly make a fix in the model and continue the current training process. BIDViz supports direct setting of hyper-params, creation of dependencies between parameters, and creation of new time-dependent policies to adjust parameters.

Currently, monitoring the machine learning processes is done in the traditional way. It requires users to:

- Modify the production/experiment script to include code that computes desired metrics before training starts. Once the training processes start, there is no way to add new metrics or delete old metrics.
- Log those metrics into a file or print them to screen. Users are not able to freely browse through the history and select specific metrics to look at.
- Write custom code to parse and plot the logged metrics to gain insights. For a different experiment, users may need to write different code to parse those files and plot debugging statistics.

It is clear to see that the current approach is inefficient and slow, especially when training cycles are long. In particular, it has three main disadvantages:

1. Coding both the logging part and the plotting part is tedious. The code that users write for one training process is usually not reusable to other experiments. In addition, plotting program must directly follow the format of logging program, which makes the code not extensible.

2. Metrics computing and logging code cannot be changed after the model is running. It is often the case that user specified metrics cannot capture the whole training process, which needs to add other metrics after training starts. Therefore, if users forget one of the metrics to log, or want to explore more of the model after seeing something rare, they must stop the current process, modify the script to include the new logging, and rerun the model.

3. If the model is showing unexpected behavior, the only way to modify it is to stop and restart. Having a model trained for an hour, or even longer, and then detected abnormal metrics, users need to start over the whole process. The current approach is very time-consuming and inefficient.

Therefore, we present our machine learning visualization and monitoring system, BIDViz, an extensible toolkit to help machine learning practitioners to interact with the model training process. BIDViz is based on the GPU-accelerated machine learning library BIDMach [6]. To overcome the drawbacks that come from the traditional approach mentioned above, the system support adding metrics, rendering plots and modifying model parameters in the same live training session with an user-friendly interface. We in fact follow those procedures to demonstrate our effort to diagnose deep neural networks run on very sparse input data. This approach leverages support for real time compilation, serialization and multi-threading in BIDMach and Scala. We also utilize the Scala Play framework to support high-level HTTP client/server communication with the web visualization client. We will discuss the features and system design in later sections.

1.2 Related Work

1.2.1 Machine Learning Visualization Platform

There exists several visualization platforms for machine learning training. The most popular one is TensorBoard [2], supported by the Google TensorFlow project. To use TensorBoard, a user needs to instruct the model to log the

information into a specific file directory, then TensorBoard will be able to read that directory and visualize the logged data. TensorBoard is extremely useful to visualize TensorFlow computation graph, which helps debug if the model is built as expected. In addition, users use TensorBoard to visualize some statistics to monitor machine learning training process. We want to argue some key features that our system outperforms TensorBoard:

- TensorBoard is a separate process from the running TensorFlow training process. Those two processes do not share the memory space, which makes interaction very 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 modify training system.

- Tensorflow has the computational graph structure, which is very 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.

- TensorBoard operates by reading TensorFlow events files. Thus, visualization server needs to check the events files every period of time which may create busy waiting. In addition, events files are written and read on disks which might be more expensive compared to memory. Our system mitigate such issue by directly allowing training thread to communicate with the visualization server through a socket. Such socket does not create busy waiting in the system and primarily only uses memory.

In addition to TensorFlow, other machine learning visualization toolkits include Keras Callback [9] and TensorDebugger [16]. Keras Callback has very similar features to TensorBoard which opens a separate process to evaluate all pre-specified logging code. Therefore it has all the drawbacks mentioned above. Another Tensorflow tool, TensorDebugger, uses Jupyter notebook [26] as the backend to support running both training and plotting in the same process. It can set breakpoints to allow users to pause the training process and debug the current data through the computation graph. However, the key issue here is still the limitation of computation graph. No matter whether one uses TensorDebugger or TensorBoard, once the computation graph is constructed, no new code or diagnostics can be added. In addition, although using Jupyter notebook is very convenient for exploring with a paused training session, it is very

difficult to reuse the logging and plotting code in other experiments, or share with other people. Our implementation has a simple interface that makes sharing and open source of diagnostic code possible.

1.2.2 System Monitoring Platform

Part of the functionality of BIDViz is monitoring the machine learning system. System monitoring has developed several very mature platforms and toolkits. One of the most popular platform in the industry is Graphite [10]. Graphite allows users to submit metrics files to its own database system and store numeric time-series data of those metrics in the database once a certain period of time. Then it supports fetching those metrics data from the database and rendering graphs on demand. Along with Graphite, Grafana [1] is a powerful dashboard on top of Graphite to support time-series anlaytics. This whole ecosystem is very similar to our system but with the following key differences:

- Since our goal is to monitor machine learning training proceses, if we use Graphite and Gradana ecosystem, the result is very similar to what TensorBoard has done. In particular, since Graphite askes users to submit metrics scripts to the database, it is the same thing as having the metrics code in the training script and submitting the training script to Graphite database. It is two different processes in nature, which creates the separation between server and training processes. Therefore, we are not able to modify logging code and changing model parameters once the job starts.

- The monitoring system in Graphite can be very expensive once users request more graphs. Because the key underlying structure is to have the web interface fetch data directly from the database, cache or even on disk, once users open a new tab and request to display new graphs, Graphite needs to query more data from the database, and refresh all the graphs currently displayed in the browser.

Besides those two key points mentioned above, our system actually shares some similarity to Graphite ecosystem. For example, Grafana has online dashboard community to share their self-defined visualization. We also have clearly defined API for users to share their plots with others.

There are a lot of real-time monitoring platform developed in the academic world such as [8, 25, 29]. However, those platforms have specific targets to monitor which do not allow dynamic changes to the monitored metrics. In addition, similar to Graphite, those systems are not suitable for machine learning processes as they do not directly

share memory resources with the training thread, which cannot serve as the real-time debugger.

1.2.3 Interactive Environment Platform

Interactive machine learning has become a very popular field of research in human-computer interaction (HCI). However, interaction was approached very differently across different research papers. For example,

- In "Interactive learning with convolutional neural networkds for image labeling", [21], Langkvist et. al. propose a human-in-the-loop system to fix falsely labeled data when training a convolutional neural networks. A similar work is [18] where the Kappor et. al. ask users to express their preferences on decision boundaries for multiclass classification problem. In [20] Kulesca et. al. introduces an explanatory debugging system that allows users to commnuicate with the system about how the learning is done in the system and to correct false predictions. All the papers mentioned here, along with other papers such as [19], mostly focus on using human knowledge to guide the system label or train dataset.

- In [24], Patel et. al. propose to let users use the results from multiple models to diagnose bad features, noisy observations and suboptimal algorithms. A similar work is Talbot et. al.'s EnsembleMatrix, a tool to present confusion matrices to help users understand performance of various models [30]. There are a lot of other research work, such as [28], all try to provide a platform that users can directly analyze the outcomes after training models.

- In [17], the Jiang et. al. propose a machine learning architecture what can directly change model parameters during training. The interaction here is directly changing training models to tune hyperparameters. New code and diagnostics cannot be added to running models.

It can be seen from the list above that interactive machine learning is a very broad area. From using human knowledge to guide machine learning training processes, to visually analyzing the training results, interactive machine learning covers every aspect of machine learning training processes. As the authors argue in [4, 5, 7, 13] about the expectation and challenges of interactive machine learning systems, that ideally one should involve users from exploring model building using human interaction patterns to refining models through the interface. However, none of the papers mentioned above covered the whole process of interactive machine learning. Our system exactly defines interaction as

a means to allow users to be directly involved in the entire training process, starting from explorary analysis, to model diagnositcs, and production inference pipeline debugging. Such interaction has never been fully accomplished.

1.3 System Design

1.3.1 Overview

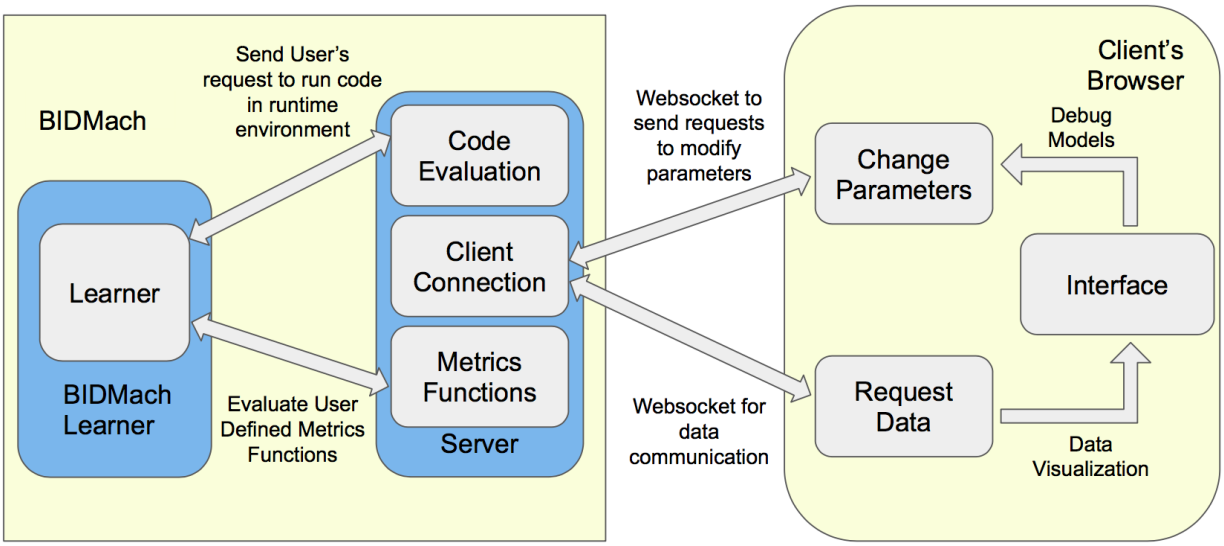

Figure 1.1: An illustration of our system design connecting BIDMach server with user interface. Each blue square represents a thread.

BIDViz is designed with extensibility in mind. We want a flexible tool for scientists and developers that can to customized to their needs. To accomplish this, we designed this system to be modular with a clear defined interface between several pieces. This did not only accomplish extensibility, but also made our own development easier.

A BIDViz application consists of 3 modules, illustrated in 1.1. The first is a machine learning library that handles the definition and training of the model. The second is the BIDViz server, it consists of a channel that observes the current training state, and a web server that interacts with the user. Finally a third module is the web application served by BIDViz that users see and interact with.

We have chosen BIDMach [6], a fast GPU-accelerated machine learning library to base BIDViz on. A script written for BIDMach usually instantiates a ”Learner” object, and trains that Learner. A Learner contains the main

training loop that iterates through each mini-batch of data and performs optimization algorithms, such as stochastic gradient descent or ADAM, and regularizers defined by mixins.

BIDViz server will handle the creation of a new function when users submit a code snippet that defines a metric through code evaluation, and handle the request from users as a websocket RPC server.

When BIDViz starts, 2 separate threads are created: the first one is the *training thread*, which is a thread that runs the current training loop; and the second is the *serving thread*, which is the thread that responds to the user requests and sends out relevant information. BIDViz has the option to execute code in either thread.

1.3.2 Webserver as listener of the main training loop

To make the webserver accessible to the data stored in `BIDMach`, we make BIDViz server an observer of BIDMach's Learner object by implementing the trait below.

```
trait Observer {
   def notify(model:Model)
}
```

`Learner` class is the class that owns the training loop, it is responsible for loading the minibatch of data, and execute model update steps. In case of gradient based optimization problems, such as training a neural network using SGD, this means computing the gradient and calling the corresponding updater; in case of EM style cluttering algorithms such as kMeans, it means one step of mean update.

An observer will get notified on each iteration of the training loop, and it will get a reference to the `model` object. The model contains everything we need to define the current model. In the case of a neural network, this means a list of matrices representing the weights of every layer. It also contains a list of matrices of the current gradient and the current minibatch of data. With access to these information, we can compute interest metrics such as the training loss and accuracy, or distribution of model weights and gradients.

When BIDViz is notified, it will iterate through a Map containing functions that takes a model and return a matrix. How those functions are created is described in the next section. The matrix returned is then serialized and sent out to the front end through websockets. This evaluation is done in the *training thread*, because in this way we are sure that the model weights do not change while we are doing this computation. If we want to do the computation in the *serving*

thread, then we would need to snapshot the current model weights to avoid the race condition of potentially computing on partially updated weights, and that means copying out the weights out of GPU, which could be expensive giving the large size of model weights in a deep neural network. We assume that the result of the metric computation, such as model loss and accuracy, is small in comparison. Therefore, we only sends out small amount of data out of the training thread and over the wire (websocket). The writes to the websocket are asynchronous, and are handled by Scala's Play framework, using the Actor-Reactor library (Akka) that comes with Play. Doing metric computation does add some overhead to the *training thead* that potentially can slow down the computation. However, the metric computation should be small compared to the gradient computation for one gradient step, so the overhead should be minimal. However, it is possible that a user submits a slow code in metric computation, such as network calls. In the future we would try to detect and add warning for this case.

1.4 Communication Protocol with the Front End

All the communication are handled via JSON-based messages through a websocket, which is established when the webapp loads. BIDViz defines a `VizManager` (javascript) singleton object that handles the communication with the server.

There are mainly 2 types of communication between the server and client. One is `data_point` notification, this occures when the BIDViz computes a metric, such as loss, and send it to the front end to be displayed; another is client requests, such as getting a list of current parameters or send a code snippet to be evaluated. The first type is initiated from the server, so it is natural to use websocket as this is the only way the web browser can get unsolicited data from the server. The second type starts with a request from the client, and then the server performs the request actions and reply with the requested data. This process is normally implemented as an AJAX request. However, in BIDViz both of them are implemented using the same websockets.

1.4.1 Handling data point event

A data point message looks like the follows:

```
{
```

```
  "msgType": "data_point",
  "content": {
    "name": "",
    "ipass": int,
    "shape": [int, ...,],
    "data": [float, ...,],
  }
}
```

The msgType field indicate the type of message, so VizManager knows to route it to a chart. The `name` field in `content` is used to find out which chart this data point belongs to. The `ipass` field defines the current pass, which is an increasing integer recording roughly how many iteration the training loop has run. And finally, the `shape` and `data` are the serialized matrix. `data` is a list of float representing the straighten up matrix, and `shape` the sizes the original sizes. For example, a 5×5 matrix will be represented with `shape = [5,5]` and `data` a list of 25 numbers.

Once this message is understood, `VizManager` will then route that data to the corresponding graph by calling the `addPoint` method on that graph. Every object that has that method can be treated as a graph as long as it implement this method. It is very easy to create different charts using any graphing backends, such as D3, C3, HiCharts or Vega. Xuan's report will go on depth on different graphs that we suppport.

1.4.2 Client initialized request event

When the user performs an action, such as add a new chart or edit a hyperparameter, a request will be sent to the server. Traditionally this is done using AJAX requests. Using AJAX has the benefit of the browser handling the routing of each response to the corresponding callback, but has the disadvantage of being a full HTTP request, which incurs significant overhead in sending the full HTTP header. We have chosen to minic AJAX using websockets. VizManager assigns an id to each outgoing websocket requests, and keeps a map of those ids and the corresponding callback function. On receiving a response, it will use the id to route the result to the correct callback. A callback message looks like below:

```
{
```

```
  "msgType": "callback",
  "content": {
    "id": "",
    "success": true,
    "data": {
    }
  }
}
```

When the client initiates a request, VizManager assigns a caller id to that request and remembers the callback function of that request in a dictionary. When the corresponding reply arrives from the websocket, VizManager will route `data` field to its callback.

1.4.3 Usage

BIDViz currently targets machine learning practioners that use BIDMach. It is very simple to add usage of BIDMach in any existing script.

Usually, a BIDMach script looks like the follows:

```
// ... defines options
val learner :Learner = Net.createSomeSubclassOfLeaner(...)
// ... more code
learner.train
```

The user defines options allowed in Leaner.Options class, instantiates a object of the Learner class (or its subclass), then call the `train` method on it.

To add BIDViz, simply instantiates a `WebServerChannel` as observer of learner before calling `train`, as follows:

```
      learner.opts.observer = new WebServerChannel(learner)
```

When when this line is executed, a web server will be created and you can direct your browser to `localhost:10001`

to start using BIDViz. nn.opts.observer = new WebServerChannel(nn)

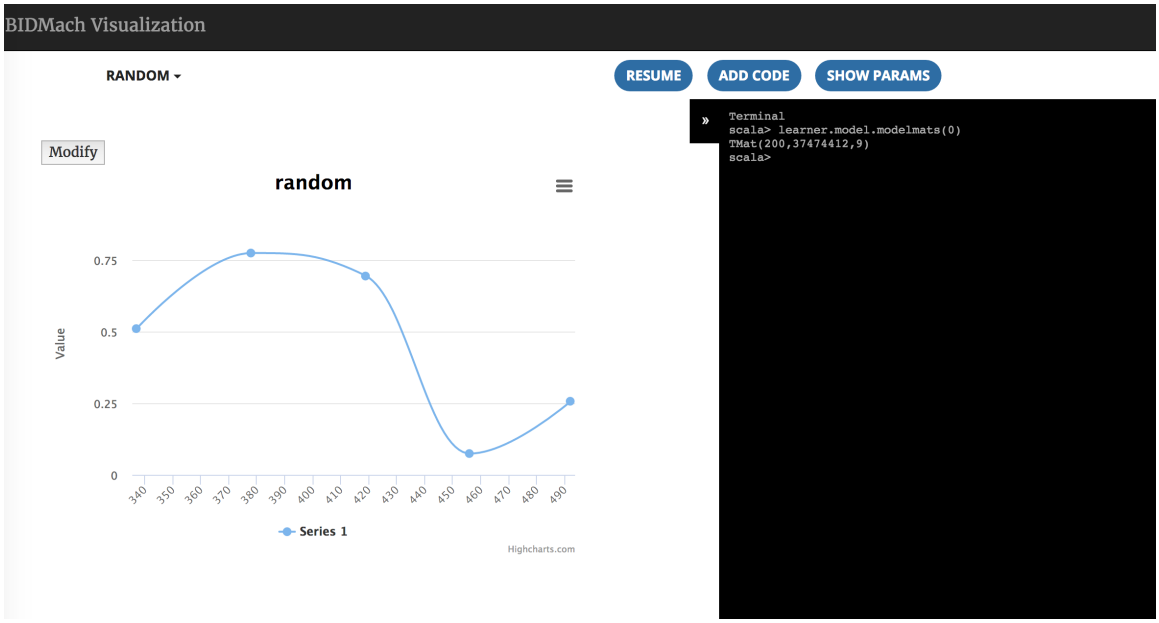

Figure 1.2: User interface that communicates between end-users and training processes. Most of the space on the webpage are for rendering time series of user-defined metrics. A sample line chart is shown in the figure. There are three blue buttons on the top of the page which handle resuming/restarting training process, adding metrics code, and changing model parameters. On the right side of the page, users can choose to expand a floating terminal, which is already expended in the demonstration above. Users can use such terminal to evaluate specific commands in real-time training environment.

As shown in figure 1.2, BIDViz UI provides feature such as displaying real time charts, running custom commands on a Scala interpreter attached to your running model, and defining custome charts through code snippets. Details on UI features is described more in depth in Xuan's paper.

1.5 Conclusion

BIDViz provides a way to interactively visualize the *training process* of a learning model. This dynamic interaction provides an alternative workflow for machine learning practitioners and data scientists. In the future, we would like to extend this tool to also work with other popular machine learning frameworks, such as Keras or Tensorflow.

Also, bundling the code that computes a statistic with a declarative graph definition could be used to build a

sharable format, in which one could create useful metrics and share them to the community.

Chapter 2

Engineering Leadership

2.1 Introduction

As an important branch of machine learning, deep learning provides 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 will 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. 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 teams engineering hours to deliver a project. There are many aspects that could affect the the teams throughput. Some of those are social aspects, such as how aligned the teams 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 [22]). 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 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: Divide the entire system into several independent modules. Each modules can communicate to the others only through a well-defined contract, or interface. Interface can be expanded, but never modified or removed. Each of modules are free to be modified, without changing the contract. Each module is owned by one team member, though any member can work on any module. 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, [12]). 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 [12]). This design allows each components to evolve in their own, also

allows prof. Cannys other projects to continue on the same code base, without affecting each other.

2.3 Industry Analysis

Deep learning is changing the world. However, in the Backendearly days, AI research community disregarded the potential of neural networks. For example, Marvin Minsky et al. (1969, [23]) 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, Vinyals 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 [31])

In recent years, many tech giants in Silicon Valley join a so called Race of AI. Sundar Pichai, Googles CEO, claims that we are moving to a AI-first world (DOnfro, 2016 [11]) in Alphabets 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 [15]). 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 [3]). 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.

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 [14])

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 [27])

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 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 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 Conclusion

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. We analyze current market, and identify steps to efficiently promote our final product to users.

Bibliography

[1] Grafana - the open platform for analytics and monitoring. https://grafana.com/.

[2] 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. Józefowicz, 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.

[3] Allerin. 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, 2016.

[4] S. Amershi, M. Cakmak, W. B. Knox, and T. Kulesza. Power to the people: The role of humans in interactive machine learning. *AI Magazine*, 35(4):105–120, 2014.

[5] F. Bernardo, M. Zbyszynski, R. Fiebrink, and M. Grierson. Interactive machine learning for end-user innovation. 2017.

[6] J. Canny and H. Zhao. Big data analytics with small footprint: Squaring the cloud. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 95–103, New York, NY, USA, 2013. ACM.

[7] D. Chen, R. K. E. Bellamy, P. K. Malkin, and T. Erickson. Diagnostic visualization for non-expert machine learning practitioners: A design study. In *2016 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 87–95, Sept 2016.

[8] S. E. Chodrow, F. Jahanian, and M. Donner. Run-time monitoring of real-time systems. In *Real-Time Systems Symposium, 1991. Proceedings., Twelfth*, pages 74–83. IEEE, 1991.

[9] F. Chollet. Callbacks - keras documentation. `https://keras.io/callbacks/`, 2016.

[10] C. Davis. Graphite. `http://www.aosabook.org/en/graphite.html`, 2008.

[11] J. D'Onfro. Googles ceo is looking to the next big thing beyond smartphones. `http://www.businessinsider.com/sundar-pichai-ai-first-world-2016-4`, 2016.

[12] H. R. V. J. Gamma E., Johnson R. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addision-Wesley, Indianapolis, Indiana, 2004.

[13] M. Gillies, R. Fiebrink, A. Tanaka, J. Garcia, F. Bevilacqua, A. Heloir, F. Nunnari, W. Mackay, S. Amershi, B. Lee, N. d'Alessandro, J. Tilmanne, T. Kulesza, and B. Caramiaux. Human-centred machine learning. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '16, pages 3558–3565, New York, NY, USA, 2016. ACM.

[14] B. Hussain and K. M.R. A novel feature recognition neural network and its application to character recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2):98–106, 1 1994.

[15] C. Insights. The race for ai: Google, twitter, intel, apple in a rush to grab artificial intelligence startupss. `https://www.cbinsights.com/blog/top-acquirers-ai-startups-ma-timeline/`, 2017. retrieved: 2017-03-10.

[16] E. Jang. Tensordebugger. `https://github.com/ericjang/tdb`, Jan 2017.

[17] 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, pages 535–546, New York, NY, USA, 2017. ACM.

[18] A. Kapoor, B. Lee, D. Tan, and E. Horvitz. Performance and preferences: Interactive refinement of machine learning procedures. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, AAAI'12, pages 1578–1584. AAAI Press, 2012.

[19] P. W. Koh and P. Liang. Understanding black-box predictions via influence functions. *arXiv preprint arXiv:1703.04730*, 2017.

[20] T. Kulesza, M. Burnett, W.-K. Wong, and S. Stumpf. Principles of explanatory debugging to personalize interactive machine learning. In *Proceedings of the 20th International Conference on Intelligent User Interfaces*, IUI '15, pages 126–137, New York, NY, USA, 2015. ACM.

[21] M. Längkvist, M. Alirezaie, A. Kiselev, and A. Loutfi. Interactive learning with convolutional neural networks for image labeling. In *International Joint Conference on Artificial Intelligence (IJCAI), New York, USA, 9-15th July, 2016*, 2016.

[22] P. Laplante. *What Every Engineer Should Know about Software Engineering*. Boca Raton, 2007.

[23] S. Minsky, M. Papert. *Perceptron*. The MIT Press, Cambridge, Masachussets, 1969.

[24] K. Patel, S. M. Drucker, J. Fogarty, A. Kapoor, and D. S. Tan. Using multiple models to understand data. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two*, IJCAI'11, pages 1723–1728. AAAI Press, 2011.

[25] V. Paxson. Bro: a system for detecting network intruders in real-time. *Computer networks*, 31(23):2435–2463, 1999.

[26] F. Pérez and B. E. Granger. IPython: a system for interactive scientific computing. *Computing in Science and Engineering*, 9(3):21–29, May 2007.

[27] A. Roghani. *Artificial Neural Networks*. CreateSpace Independent Publishing Platform, London, 2 edition, 2016.

[28] D. Sacha, M. Sedlmair, L. Zhang, J. A. Lee, D. Weiskopf, S. North, and D. Keim. Human-centered machine learning through interactive visualization. ESANN, 2016.

[29] C. Stauffer and W. E. L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):747–757, 2000.

[30] J. Talbot, B. Lee, A. Kapoor, and D. S. Tan. Ensemblematrix: Interactive visualization to support machine learning with multiple classifiers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 1283–1292, New York, NY, USA, 2009. ACM.

[31] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. `https://arxiv.org/pdf/1411.4555.pdf`, 2014.