Scaling Up Deep Learning on Clusters

Jiaqi Xie

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2017-59
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2017/EECS-2017-59.html

May 11, 2017

Acknowledgement

I really appreciate the instructions and help from my advisor Professor John Canny. I would like to thank my teammates Aleks Kamko and Quanlai Li for tackling all the obstacles and finishing this capstone project together. I have learned a lot from this project.

University of California, Berkeley College of Engineering

MASTER OF ENGINEERING - SPRING 2017

Department of EECS

Data Science & Systems

Scaling Up Deep Learning on Clusters

Jiaqi Xie

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

Approved by:

1. Capstone Project Advisor:

Signature: [illegible] Date 5/11/17

Print Name/Department: John Canny / EECS

2. Faculty Committee Member #2:

Signature: [illegible] Date 5/11/17

Print Name/Department: Joseph Gonzalez / EECS

Scaling up Deep Learning on Clusters

Capstone Project Paper

Jiaqi Xie

Electrical Engineering and Computer Sciences
Data Science & Systems
M.Eng., Class of 2017
May 11, 2017

Executive Summary

This capstone project focuses on developing a cost-effective, energy-effective and computationally-powerful distributed machine learning library (BIDMach[1]), to catch and lead the current trend of big data. We are collaborating with our industry partner OpenChai, a start-up aiming at putting distributed BIDMach on their mobile-GPU[2]-based hardware and leveraging all its advantages. Ultimately, our goal is to bring machine learning to small companies and customers, to further benefit the general public.

Our capstone team work on making BIDMach running on cluster of machines to increase the overall computing power. As a setup, we made great improvement on BIDMach's communication framework, which is discussed in my paper. The capstone report of my teammate Aleks Kamko is all about our core technical accomplishments - parallel version of machine learning algorithms. Quanlai Li will talk about other miscellaneous achievements in his part, such as the better updating rule - EASGD[3] and computation of network communication bandwidth.

Chapter 1 of this paper (Technical Contribution) covers the motivation, design, implementation, result and discussion of the communication framework, which is the underlying core of parallel models. On the business side, OpenChai's integrated product aims to solve three main problems in current mainstream machine learning solution - low computation power, waste of energy and data privacy issue. Details will be further discussed in Chapter 2 - Engineering Leadership part of this paper.

[1] The prefix BID stands for Berkeley Institute of Design. The suffix "DATA" for data, "Mach" for Machine and "Mat" for Matrix.
[2] Graphics Processing Units
[3] Elastic Averaging Stochastic Gradient Descent (Zhang, 2015)

Chapter 1: Technical Contribution

Jiaqi Xie

Table of Contents

1. Introduction

Project Description

BID Data is a toolkit for big data developed by our advisor Professor John Canny. For now, it is the fastest big data tool when running on single machine (BIDData, 2015). Our capstone team work on developing a distributed version BIDMach (the core machine learning component of BID Data suite). The final delivery will be a highly cost-effective, energy-effective and computationally-powerful machine learning library. Ultimately, our goal is to bring machine learning to small companies and customers, and further benefit the general public.

Tasks & Work Distribution

To achieve the ultimate goal, tasks of this capstone project are divided into technical and business sides. Technical tasks include technically writing codes, scaling up (writing distributed version) machine learning algorithms and improving BIDMach's performance on clusters. Business tasks are more related to the direct cooperation with OpenChai, though they still come with some technical details. Below is a brief task list generally ordered by time:

Technical:

1. Get familiar with systems and tools, learn the never-met programming language Scala.
2. Design, implement and improve BIDMach distributed communication framework.
3. Implement data-parallel algorithms on clusters, including kMeans and Random Forest.
4. Scale up model-parallel algorithms (such as GLM[4]) with the implemented distributed communication framework.
5. Implement model-parallel algorithms on clusters, with Spark.
6. Work on scaling up distributed version deep learning models like sequence to sequence model.
7. Other miscellaneous and ad-hoc works related to the communication part or parallel code.

[4] General Linear Model

Business:

1. Get distributed version BIDMach work on OpenChai hardware.
2. Run benchmark of our algorithms on OpenChai machine, against other machine learning toolkits like MXNet.
3. Optimize algorithms on new OpenChai architecture, with technology like DMA[5].

Within each one of the tasks listed above, there are several sub-tasks and many smaller ad-hoc issues. As from the nature of Computer Science (especially software engineering) group project - tasks are in flat structure without clear borders between different parts, we don't have a high-level work breakdown.

Our typical working style is discussing and assigning work when new small tasks come up in meetings. We have weekly meetings with our advisor to talk about the work finished in previous week and upcoming tasks. The industry partner OpenChai visits UC Berkeley every 2-3 week to ensure that everyone is in the same track. Our capstone team sets up two working sessions (5-6 hours in total) per week to work collaboratively. We peer review each other's code, bring up obstacles we met and overcome them together. With that being said, for each specific task, we indeed clearly assign one person to be the person-in-charge.

My Contribution

I contributed mostly to the second task in Technical side - to improve BIDMach distributed communication framework. This is essentially a core base of the whole technical part of this project. Only with it can different machines communicate well with each other to update the models and do machine learning efficiently.

Data-parallel algorithms mean that different machines only train on their own chunk of data, without the mixing of models. Those algorithms don't utilize much of this communication

[5] Direct Memory Access

framework, and the collection of models at the final step can be handled easily. In contrast, model-parallel algorithms require massive interactions between machines to update models at training time. Thus, a well-designed and powerful communication framework is necessary.

This paper is mainly about the implementation details of such framework, along with challenges I met and solutions to those problems. Result is shown later to substantiate the improvement from my work, against the current working code. Directions of future exploration on this part will be discussed at the end.

My teammate Aleks Kamko is the primary contributor to distributed machine learning algorithm, with my work as the base. He will write about the implementation details and extend to more complicated Deep Learning models in his paper. Quanlai Li's paper covers his work on different small tasks. One is about the better updating rule - EASGD, the other is the functionality of computing bytes transferred among different Workers in the cluster.

2. Previous Work

There are various design choices of the communication framework between machines in clusters. Apache Spark, a fast and general-purpose open source cluster computing system, with the main advantage of easily handling data over clusters, is one of the state-of-the-art mature product in market (Spark, 2017). Its own machine learning library (MLlib) builds on the communication framework to do large-scale big data processing and computation.

Previous year's MEng capstone project team contributed on integrating BID Data with Spark. As James Jia states in his capstone project paper, he used Spark's implementation of treeReduce as a start for the distributed version machine learning algorithms (Jia, 2016). He created an API[6] to abstract the details and ran BIDMach on Spark. The team successfully scaled up KMeans, a data-parallel algorithm, to validate the integration with Spark (Jia, 2016). Some of the work in our capstone project, such as scaling up Random Forest and GLM, are also based on Spark.

[6] Application Programming Interface

Kylix is a butterfly all-reduce communication network developed by our advisor Prof. John Canny and his student Huasha Zhao at 2014 (Zhao & Canny, 2014). It is a brand new and efficient network which handles the model mixing between machines and improve network performance within clusters. The previous implementation of distributed algorithms with Spark framework has an inherent problem when working on Kylix. Weights of models are only allowed to be moved/transferred at the time interval between two supersteps. In other words, machines must be synchronized to get updates from other machines' models. Kylix loses its efficiency of communication, since faster machines are idly waiting for slower machines. This is where my work targets on - to make model updating asynchronized, making fully use of the advantage of Kylix, and improve the overall computation power of distributed version BIDMach.

3. Methods

Architecture of the communication

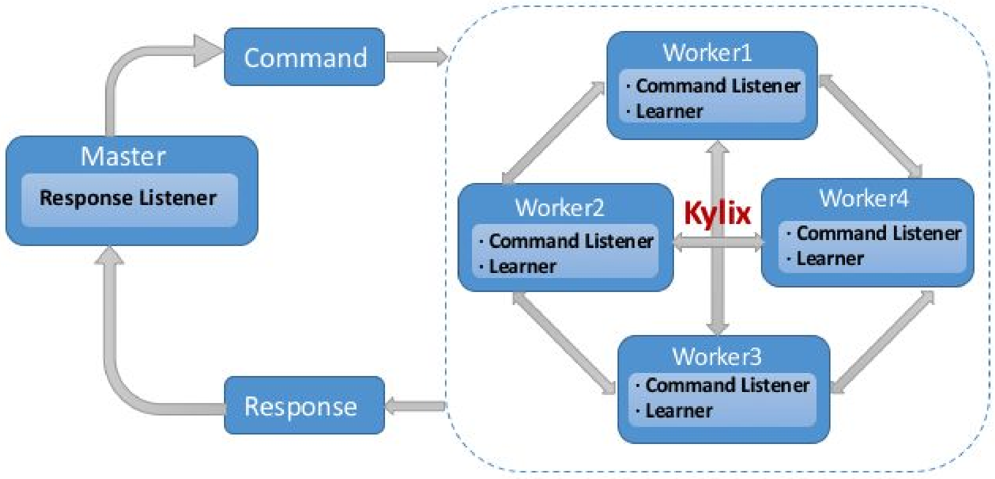

Figure 1. Architecture of BIDMach communication

Before going deep into the implementation details, it's necessary to understand the architecture of the communication framework. As shown in the graph above, we adopts a similar Master/Worker architecture as Spark. The *Master* machine runs the main program and coordinates the whole training process, as well as allocating resources. *Worker* machines are where the real training resides (i.e. the *Learner*s associated with *Worker*s running assigned training tasks). The *Master* communicates with each one of the *Worker* through the system of *Command* and *Response*, and they also have corresponding *Response Listener* and *Command Listener*. *Worker*s are closely connected together, with the butterfly allreduce communication network Kylix locating within their network, at the right side of the graph - it essentially connects all the *Worker*s in a complicated way and makes the allreduce operation efficient.

The Master should be able to directly instruct Workers to do whatever possible and needed by the Master. Tasks can be as small as computing 1+1, or as complex as a multi-layer deep learning model with frequent mixing between models of different Workers. Three important functions, *parEval*, *parAssign*, *parCall*, are designed and implemented in the class *Master* to provide the Master such a method, and of course, to make sure Workers respond accordingly.

"Parallel" Functions

parEval stands for "parallel Evaluation a statement". It takes some codes (in string format) as the argument and essentially enters the given codes at the terminal of all Workers, then Workers send the standard output back to Master's terminal. As an example, *parEval("1+1")* gives a 2 as the result, and other relatively simple tasks can be done in this way.

parAssign is "parallel Assign value to object", taking a string as the variable name and an object. It assigns the object with the given name at all Workers' workspace. With this function, Master can get access to variables in all Workers, setting a base for operation on those variables. A simple example is that *parAssign(1, "a")* assigns the integer 1 to the variable *a* at all Workers, then this can be used together with *parEval* to do *parEval("a+1")* - sending back 2 as the result.

The last method is also the most powerful one, *parCall* - "parallel Call a function". It takes a function as the argument and effectively call the function at Worker machines. Return values are sent back from Workers after that function exits. I designed this method to be generic, in sense that the Master can instruct Workers to do almost all the tasks needed for communication. The function to be sent takes one argument - a *Worker* object, which is the core at each Worker machine. Through the *Worker* object, the function can get access to all data needed, such as the *Learner* object, training options, model weights and parameters. Besides, functions of the class *Worker* or *Learner* can be easily called in the function sent. The return type of the function to be sent is set to be *AnyRef*, which plays as a placeholder, meaning objects of all classes can be sent back to the Master, providing much freedom in various applications.

Challenges were met in my process of working. I was not familiar with the programming language Scala, such as the syntax, data types and the way function operates. It brought me some difficulties in designing the code structure and really writing codes. This is especially the case in the relatively more complicated *parCall* function. One specific and serious challenge originates from a special property of functions in scala. For each function defined, it initiates an anonymous class associated with the function. Even if the Master transmits content of the function to Workers, they have no idea of the existence of that anonymous class and could not transfer the bytes data back into the original function. It took me much efforts to search and hack a solution. With the help from my teammate Aleks Kamko, we digged into source code of Spark to learn how it handles the problem. Finally we added a new file *ClosureCleaner.scala* (used under the contributor license requirement of Apache Spark) to solve the issue. The *Cleaner* essentially renders any *Closure* to be serializable (transmittable through the network) if they can be done so safely, thus, it helps sending the anonymous class together with the function (within a *Closure*) to the Workers[7].

Master actively initiating model updating

As briefly introduced in the "Previous Work" section, the main advantage of our newly-implemented communication framework is its compatibility with the efficient Kylix

[7] The Apache Spark source code can be found at
https://github.com/apache/spark/blob/master/core/src/main/scala/org/apache/spark/util/ClosureCleaner.scala

algorithm. Its core is the asynchronized model updates between Workers, so that the programs on Worker machines don’t need to pause and wait for all others to be on the same step.

In our design, Workers keep running the training process of machine learning by themselves, as fast as possible, just like it’s operating as a single machine. They would only send and receive data and do the update when Master gives an explicit instruction (sending a signal). Essentially the Master works as a coordinator for the training and updating of the whole cluster of machines.

A naive implementation to improve the efficiency is to make Master keep sending the signal again and again, also as fast as possible. Obviously this is bad idea, as it’s busily using the Master’s CPU time and occupying the network bandwidth. Too frequent updates may also slower the training at each Worker, since they have to spare CPU time to do the updates.

We made a design that the action of Master sending signals is actually responsive. It would only initiate the update again if it receives a certain number of responses of the previous signal from the Workers. This is a relatively “intelligent” updating method as the Master at least get some feedback of whether Workers have accomplished the previous update. Furthermore, the criteria for Master to decide whether to initiate update is not the brute-force thought of receiving all responses. Instead, we designed three levels of the number of responses for the Master to react accordingly.

First, when the waiting time is relatively short, say 500ms, the Master will wait for all responses and go to the next iteration. This is the typical situation as the updates should be as frequent as possible, while making sure that each Worker really gets the update. Second, if the Master waits for longer, say less than 1000ms, it would relax the criterion and decide to send next signal once some of the responses have been received. The portion of responses received is set manually - we set it to 0.75 in our testing on the 4-machine cluster, which means that 4*0.75 = 3 responses are enough to initiate the new update. Finally, if waiting for too long, the Master would just ignore responses and send the updating signal anyway, since probably a problem happens in the previous update.

The extreme case of the long waiting time would be losing some of the *Worker*s. We haven't implemented such a functionality to handle this in this capstone project's work yet. Actually, with the nature of robustness of the training result retrieved from distributed Machine Learning algorithm, ignoring the loss of *Worker*s and just proceeding the learning process won't affect the final result much. Effectively, we are just losing part of the training data in getting the model, though in this case there would be idle waiting for the failed machine. A better but still naive way would be stopping the whole system instantly if several machine failures are detected, for example, "waiting for too long" for more than 5 consecutive iterations. In this way, the human operator can immediately get the alert and respond correspondingly. Ideally, in the future, such failure should be solved automatically by the machine, preferably with a logging system to recover the training process.

4. Result

Our design and implementation of the communication framework was tested for many times and it performed well. We worked on an Amazon EC2 4-machine cluster and ran several distributed machine learning algorithms (we have the plan of expanding the test to). KMeans was implemented by previous year capstone team. The details and result of distributed Random Forest and GLM will be discussed in Aleks Kamko's paper.

Our distributed algorithms showed a comparable accuracy with the single-machine version, proving the correctness of the "scaled up" algorithm itself, as well as the newly-implemented communication framework. Then the algorithm efficiency can be shown by the computation time or speedup. Ideally, a distributed algorithm running on a cluster of m machines would take only $1/m$ of the time of the single-machine version. In reality, there's cost associated with the communication, so the speedup could not reach the theoretical value of m.

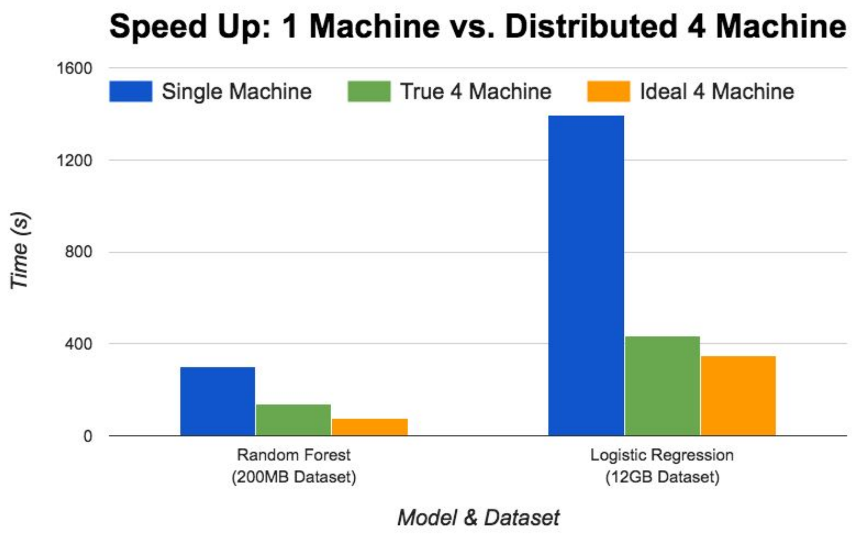

Figure 2. Training Time (Speedup) for Random Forest and Logistic Regression

Shown in the graph above is the time of algorithm used on single machine vs. distributed 4-machine cluster, for Random Forest and Logistic Regression (one kind of GLM). The orange bar represents the ideal running time for distributed algorithms. It's obvious to see that our result is pretty close to the ideal, implying the high efficiency of our codes. And it's also easy to notice the gap from theoretical optimal time, which is our future exploration direction.

As for the accuracy, our distributed version algorithms reach almost same level as the single machine version. For Random Forest on the UCI Year Prediction Dataset, the MSE (Mean Squared Error) is 90 for both versions. For Logistic Regression on Criteo Click-Through Rate dataset, original algorithm gets an AUC (Area Under the Curve) accuracy of 0.7825, and the distributed version reaches 0.7745. We feel such a small loss on accuracy is acceptable, given the nature of the randomness in distributed algorithms.

The close-to-optimal speedup and no loss in accuracy proves that distributed BIDMach could safely be applicable to most practical Machine Learning applications, and it greatly solves the problem of training time in the current era of Big Data.

5. Discussion & Future Work

Although our design and code of the communication framework is proved to be working well and boosting the performance of distributed machine learning algorithms, there's still much room for future researchers to improve.

First, the current low-level communication channel of BIDMach is still the traditional ethernet, with connection through sockets. With specific hardwares (such as OpenChai's product that the clusters locate in single physical machine), it could be replaced by DMA (Direct Memory Access). This technology allows the Workers to directly access data on other machines' memory, bypassing the CPU, thus shortening the communication time significantly. Another system design may be more suitable and efficient for such a data transmission channel. Some explorations will need to be done.

Second, the three newly-implemented functions *parEval*, *parAssign* and *parCall* are applicable on the models and tests we are using now. They should enable the Master to perform all possible and needed tasks. However, it's possible that in the future, they will not be compatible with a new machine learning algorithm requiring some more advanced instructions from Master. Utilizing current functions to achieve such a task could be messy, then at that time, the framework should be redesigned or the functions should be rewritten.

Third, our current implementation for the functionality of Master actively sending updates signal, as stated above, still takes manually-set parameters. There's a portion-of-workers threshold which determines number of responses the Master should collect before sending signals. Several waiting-time threshold indicates the different levels for the Master to wait. These should definitely be different for different algorithms and hardwares. A smarter way to determine these parameters is making use of the statistics of the response time, such as average, variance and maximum. Auto-tuning the parameters or using machine learning are even fancier methods.

6. Conclusion

This capstone project - Scaling Up Deep Learning on Clusters, aims at bringing the state-of-the-art machine learning toolkit BIDMach onto distributed machines. The distributed algorithms improve computation efficiency significantly, while maintaining the advantage of low energy cost. Our cooperation with OpenChai focuses on integrating BIDMach with mobile GPU and making a box-size machine learning cluster, to challenge the mainstream solution in the industry, to make machine learning more easily accessible for the public.

My main contribution to this project is implementing the BIDMach communication framework aside from Spark's framework, to make use of the efficiency gained from the mixing algorithm Kylix. My work essentially sets up the base for later development of fancier models like deep learning models. The three "parallel" functions and the functionality of Master actively initiating updates are all important components of it. Implementation details were discussed as well. Future work of this part would be mainly about increasing the "intelligence" of the algorithm and possibly refactoring along with the whole project.

This is an interesting and important project, of great significance to academia, industry, as well as the general public. It's my great honor to make a contribution and learn a lot from it. I have explored my ability to work on real-world projects, improving skills like teamwork, communication and time management.

Chapter 2: Engineering Leadership

Aleksandr Kamko
Max Jiaqi Xie
Qualia Quanlai Li

Table of Contents

I. Introduction

Big Data is a growing trend in the technology and business markets. More companies are collecting, storing, and analyzing enormous amounts of data to gain meaningful insight into their business practices. One study claims that the average industry dataset size is ≥ 1TB and growing (Canny 2013). As a result, these companies are in eager need of large-scale data storage and Machine Learning systems. The recent explosion of the Data Center and Cloud Service industry is a testament to these computational demands.

However, the growth of Data Centers and Cloud Services comes at a price. Three important challenges emerge for today's Machine Learning systems:

1. Maximizing computational throughput
2. Balancing energy consumption
3. Preserving data privacy

This chapter introduces the context behind these surfacing issues, namely how they are a consequence of the industry's recent and zealous interest in using machine learning methods to analyze big data.

Our capstone project aims to tackle these three issues using BIDData, a novel Machine Learning developed by our advisor Prof. John Canny. On a single machine, BIDData is currently the fastest toolkit for multiple machine learning models (BIDData 2015). Our team's work lays the foundation to scale these models to effectively utilize the power of cloud clusters, enabling analysis of massive datasets with unprecedented efficiency. This addresses the first two challenges.

II. Machine Learning Trends

II.1. Booming Machine Learning Industry

After six decades of research since its conception, artificial intelligence (i.e. machine learning) is receiving unprecedented attention. Leading technology giants, like Google, Microsoft, and Uber (Mercer 2016) are in intense competition with each other to build the most intelligent systems. Search engines, autonomous vehicles, language translation services, and more are becoming more advanced every day (Merrett 2015).

Other industries besides software are also catching up by integrating machine learning algorithms into their products and services. These newcomers, ranging from the financial industry to the manufacturing industry, are increasingly investing in AI (Naimat 2016).

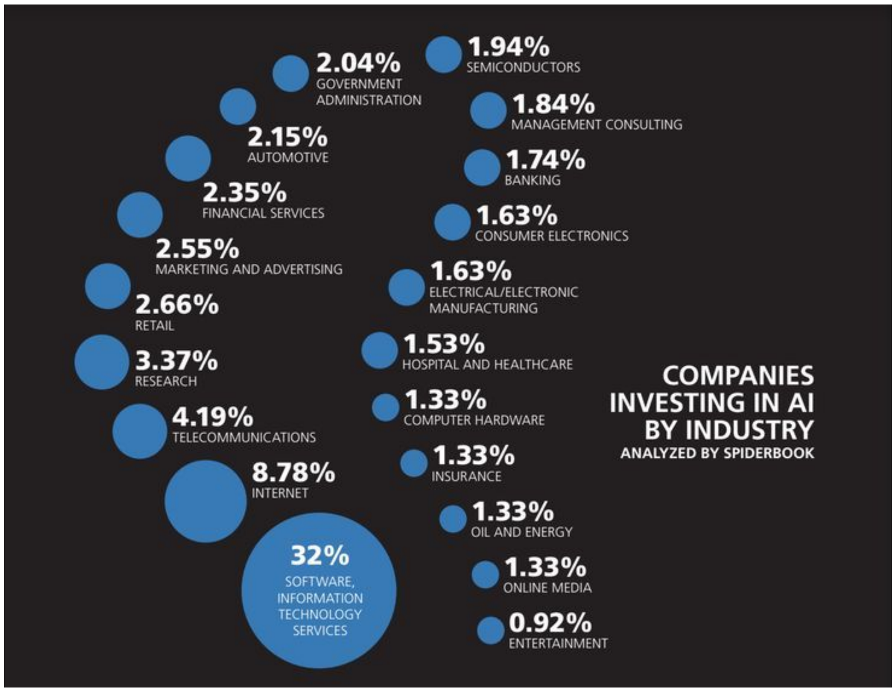

Figure 1: Companies Investing In AI By Industry (Naimat, 2016)

These investments seem to be worthwhile, driven by lucrative projected revenues. A market forecast by Tractica shows the momentum of artificial intelligence revenue in the following decade (Tractica 2015).

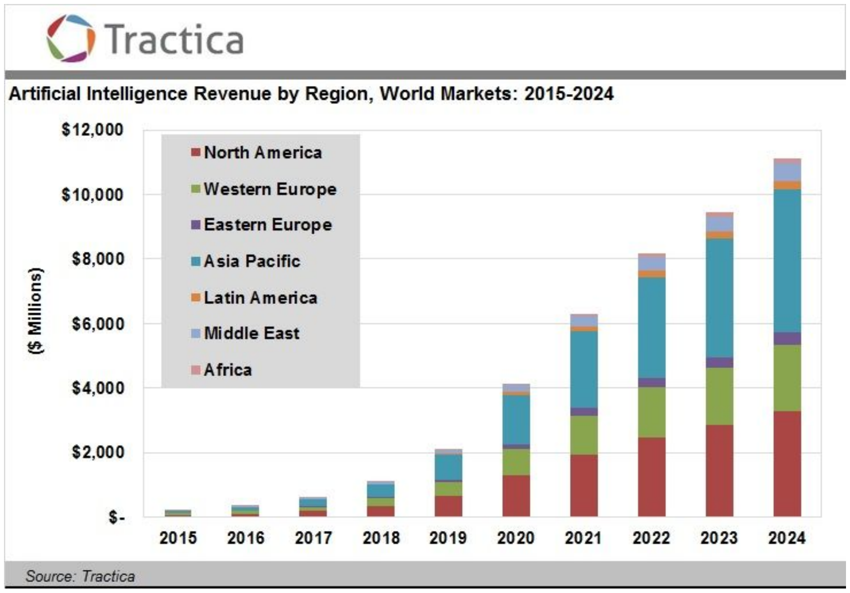

Figure 2: Artificial Intelligence Revenue by Region, World Markets (Tractica, 2015)

Naturally, the booming of machine learning and artificial intelligence necessitates more research into better methods, models, and algorithms.

II.2. New Machine Learning Research Topics

To meet industry needs, machine learning models are becoming more sophisticated. This fact is especially apparent with the recent popularity in neural networks. A popular computer vision competition, the ImageNet challenge, shows an increase in the depth (i.e. complexity) of neural networks correlating with a significant decrease in classification error (Vieira 2016).

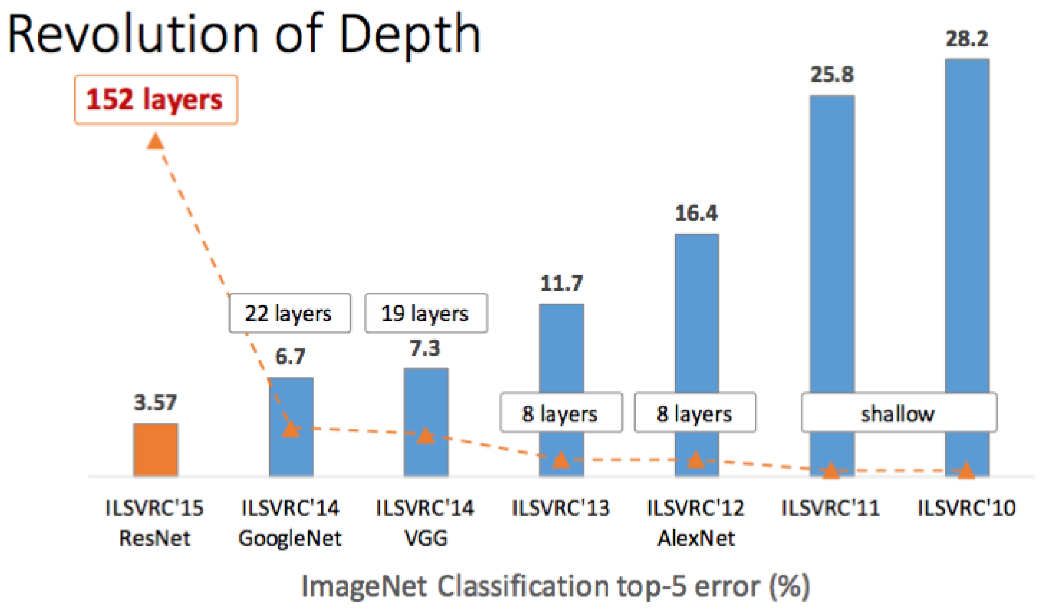

Figure 3: Revolution of Depth (He, 2015)

This increasing complexity calls for better utilizations and management of computational resources.

Until recently, machine learning algorithms have been made faster via hand-tuning of algorithms and improvements in hardware capabilities (e.g. leveraging a new GPU's[8] computation throughput). However, the need for faster algorithms is beginning to outpace these methods. Methods for scaling algorithms to be efficiently computable in parallel using multiple machines are gaining traction.

An algorithm's scalability is a measure of how well it is able to run on a distributed system, like cluster of machines. A machine learning model with perfect scalability can run at a speed proportional to the size of cluster. Scalable machine learning would allow us to learn from massive datasets at high speeds, enabling us to solve previously unprecedented problems (Braun 2014).

[8]Graphics Processing Unit

III. Big Data Trends

III.1. Big Data

The Big Data industry lies at an intersection of Business Analytics and Technology. Analytics teams of large companies have been using data mining and other predictive analytic techniques for a long time (Blau 2016). With the rapid development of the Internet of Things industry in the 21st century, massive volumes of data—50,000GB per second—are being created every day (VCloudNews 2015). Companies leverage this data by using powerful Machine Learning algorithms to extract meaningful information, creating real business value. As predicted by McKinsey Global Institute 5 years ago, "Big data [is becoming] a key basis of competition, underpinning new waves of productivity growth, innovation, and consumer surplus" (Manyika 2011).

We anticipate a significant growth potential in the data analytics market. U.S. Industry Report predicts that in 5 years, increasingly powerful computing technology will drive revenues for the data analytics industry to $53.9 billion, with an annual increasing rate of 5.5% (Blau 2016). Consequently, today is an opportune time to make an impact in the industry.

III.2. Data Center & Cloud Service

The Data Center industry is long established, helping companies store and process data since the 1950s. However, in the current era of Big Data, data centers have been evolving to fit a booming demand, resulting in modern day Cloud Service Providers. These providers modularly rent out their networked data center machines for expensive computing tasks.

The Cloud Service Provider industry is in rapid development, and its customers have a variety of interesting requirements (Blau 2016). Mainstream cloud solutions are far from perfect. Our capstone team aims to improve these services.

III.3. Problems

The first challenge our project attempts to address is computational throughput maximization. One advantage of a cloud compute cluster is its higher computational capability compared to a single machine. In theory, a cluster of 1,000 computers could achieve a peak performance equivalent to 1,000 times that of a single machine. However, in practice, this is not the case. Communication and synchronization bottlenecks between the machines in a cluster cause latency, reducing aggregate computing speed. This problem is exacerbated as data sizes grow and the system is scaled to more machines. By maximizing network throughput and lowering communication overhead, our capstone is able to improve upon the status-quo.

Power consumption and energy waste in data centers is a second challenge. An environmental action organization—Natural Resources Defense Council (NRDC)—pointed out the problem in a recent report stating that, in 2013 alone, U.S. data centers used an amount of energy equivalent to the annual output of 34 large (500-megawatt) power plants. This amount of energy could be used to provide two years' worth of power for all of New York City's households (Delforge 2014). Pierre Delforge, an expert on energy efficiency from NRDC, claims that the Data Center industry is "one of the few large industrial electricity uses which are growing" (Thibodeau 2014). This is growth in energy consumption is tied to the growing sizes of datasets, so the problem will continue to compound unless preventative measures are taken. Our capstone project aims to alleviate energy waste by utilizing data centers more efficiently, requiring fewer machines to do the same data analytics and therefore using less energy.

Finally, data security and privacy is also becoming an important issue in this emerging industry. As stated previously, companies collect massive amounts of data in order to extract useful insights for their business. The drawback is that a malicious organization could extract private information if it were to get access to the such data. Since cloud services require network connectivity, many company's data is not protected by physical boundaries — there may always be a possibility of private information being exposed via a leak or a hack. Furthermore, as the market expands with more data-driven organizations, the attack surface will only broaden. For industries like healthcare and banking, where data contains highly confidential client information, this issue becomes a top priority. Our capstone also targets these industries, and this is where our industry partner enters the picture.

IV. Tackling the Data Privacy Issue with OpenChai

Our capstone team is partnering with OpenChai to tackle the security concerns which surface from sending data into the cloud. Together, we aim to avoid this issue by enabling enterprise customers to run their machine learning models entirely offline and in-house. OpenChai is using mobile GPUs to craft a energy-efficient yet computationally powerful desktop product that is optimized for machine learning; essentially, OpenChai is building a cloud-in-a-box (OpenChai 2016). This means that OpenChai customers get total visibility and control of their information assets. Our team is working to adapt the BIDData suite to run efficiently on OpenChai hardware.

IV.1. Smartphone Market Analysis

OpenChai's product is only feasible because to their novel use of mobile (e.g. smartphone) processors. Consequently, OpenChai's market strategy rides on the crest of the global proliferation of smartphones. As shown in Figure 4 below, in China alone, the number of smartphones has increased from 189M in 2012 to over 600M in 2015, and is projected to grow to 1.6B by 2021.

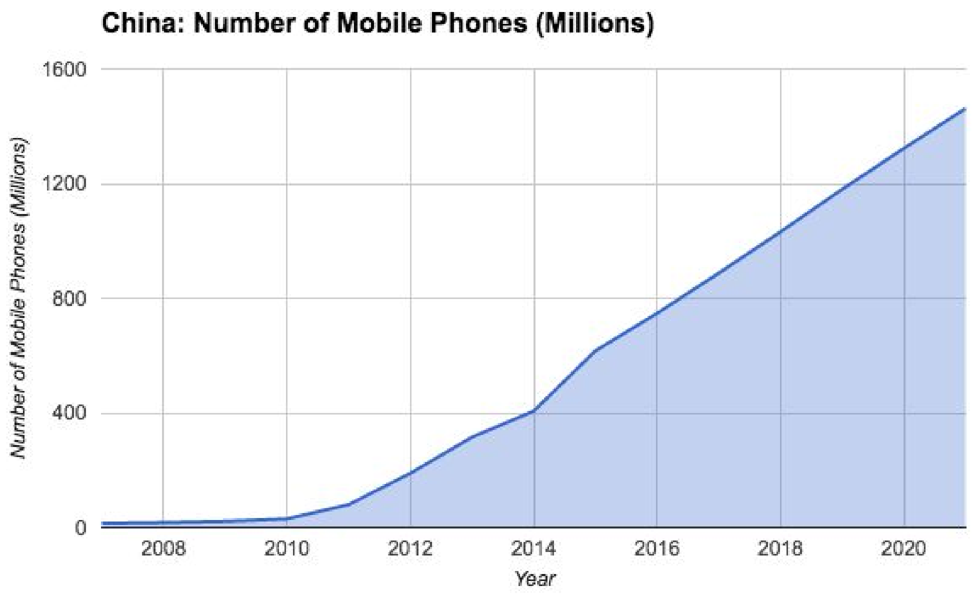

Figure 4: The number of mobile phones in China is growing quickly. (IBISWorld 2016, p.4)

India, too, is likely to follow a similar trajectory according to Morgan Stanley Research (Truong 2016). To sustain a competitive advantage under this rising demand, manufacturers are pushed to innovate (IBISWorld 2016, p. 19). One crucial avenue for innovation lies in developing more powerful mobile processors. ARM and Nvidia are two of the most prolific producers of mobile CPUs[9] and GPUs, respectively; they are also the main suppliers of the mobile processors OpenChai is putting into their product. We extrapolate that the rapid growth of the global smartphone market trickles down to pave the way for OpenChai. As the smartphone market expands, ARM, Nvidia, and by extension OpenChai, will continue to innovate with better, faster products.

IV.2. Nvidia and the TX1

Nvidia in particular, a company specializing in GPU design, is a key enabler for OpenChai's strategy. Nvidia hit the machine learning world in a blaze in 2012 when Krizhevsky et al. won the yearly ImageNet Image Classification Challenge (ILSVRC) with a neural model using Nvidia GPUs (ILSVRC Results 2012). The research group used these GPUs to engineer a novel

[9]Central Processing Unit

computer vision method, producing the most outstanding result in ILSVRC to date (Russakovsky et al., Table 8). Following this event, the use of Nvidia GPUs in machine learning exploded, correlating with continuing improvements in machine vision accuracy (as shown in Figure 5 below).

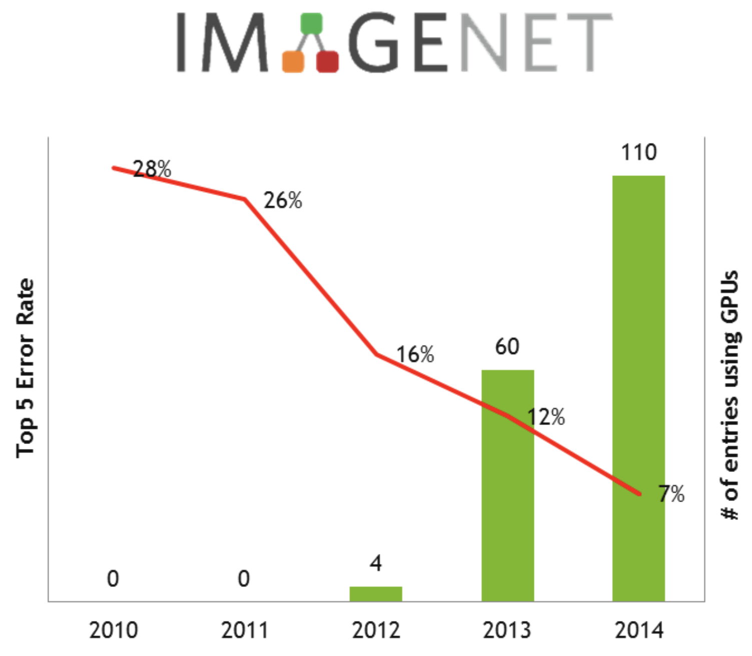

Figure 5: Following the “Krizhevsky result” of 2012, the use of GPUs in computer vision has exploded. (Gray 2015, Figure 2)

Fast forward to 2015: Nvidia unveils the TX1, one of the first processors that brings the same machine learning capabilities of high-end desktop GPUs to a mobile chip (CES 2015). The TX1 boasts impressive performance while staying up to 4x more efficient than its desktop counterpart on heavy machine learning workloads (Nvidia 2015).

The TX1 forms the backbone of OpenChai’s product. Using multiple networked TX1 chips, OpenChai can perform swift machine learning computations on large datasets offline and at a fraction of the power and cost of the GPUs provided by cloud computing platforms.

IV.3. Conclusion

Through our analysis of the expanding smartphone landscape and the machine learning space, we believe that OpenChai is poised for growth and success. Nvidia, the main GPU hardware supplier for OpenChai, is fueled by the these two markets. Any innovation in mobile GPU technology for these factors will be realized in better performance and efficiency of machine learning algorithms on mobile GPUs, transparently improving OpenChai’s product.

Reference

Apache Spark (2017). Spark Overview. Retrieved March 7, 2017, from http://spark.apache.org/docs/latest/index.html

Apache Spark (2017). Cluster Mode Overview. Retrieved March 7, 2017, from http://spark.apache.org/docs/latest/cluster-overview.html

BIDData (2015). BIDMach Benchmarks. Retrieved February 5, 2017, from https://github.com/BIDData/BIDMach/wiki/Benchmarks

Blau, G. (2016). IBISWorld Industry Report 51121c: Business Analytics & Enterprise Software Publishing in the US. *www.ibisworld.com*. Retrieved September 26, 2016, from IBISWorld database.

Braun, M. (2014). What is Scalable Machine Learning? Retrieved Febuary 5, 2017, from http://blog.mikiobraun.de/2014/07/what-is-scalable-machine-learning.html

Canny, J,. & Huasha Zhao. (2013) Big Data Analytics with Small Footprint: Squaring the Cloud. *Proc. 2013 ACM SIGKDD Conf. on Knowledge Discovery and Data Mining*

Delforge, P., & Whitney, J. (2014). America's Data Centers Consuming and Wasting Growing Amounts of Energy. *Data Center Efficiency Assessment IP:14-08-a*, Retrieved from https://www.nrdc.org/sites/default/files/data-center-efficiency-assessment-IP.pdf

Diment, D. (2014). IBISWorld Industry Report 51821: Data Processing & Hosting Services in the US. *www.ibisworld.com*. Retrieved September 26, 2016, from IBISWorld database.

Gray, A. (2015, August 13). NVIDIA and IBM Cloud Support ImageNet Large Scale Visual Recognition Challenge [Web log post]. Retrieved October 11, 2016, from https://devblogs.nvidia.com/parallelforall/nvidia-ibm-cloud-support-imagenet-large-scale-visual-recognition-challenge/. Figure 2

He, K., Zhang, X., Ren, S. & Sun, J. (2015). Deep Residual Learning for Image Recognition. Retrieved April 14, 2017, from https://arxiv.org/pdf/1512.03385.pdf.

ImageNet Large Scale Visual Recognition Challenge 2012 Results. (2012). *Image-net.org*. Retrieved October 11, 2016, from http://image-net.org/challenges/LSVRC/2012/results.html#t1. SuperVision wins Task 1 by a large margin.

Jia, J., Kalipatnapu, P., Chiou, R., Yang, Y. & Canny, J. (2015). Implementing a GPU-based Machine Learning Library on Apache Spark. *Technical Report No. UCB/EECS-2016-51*. Retrieved April 14, 2017, from http://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-51.html

Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., & Byers, A. H. (2011). Big data: The next frontier for innovation, competition, and productivity. *McKinsey Global Institute report*. Retrieved from http://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/big-data-the-next-frontier-for-innovation

Mercer, C. (2016). Nine tech giants investing in artificial intelligence: Microsoft, Google, Uber and more are investing in AI: What is their plan and who are other key players? *TechWorld.com.* Retrieved Febuary 5, 2017, from http://www.techworld.com/picture-gallery/big-data/tech-giants-investing-in-artificial-intelligence-3629737/

Merrett, R. (2015). Where is machine learning heading in 2016? *http://www.cio.com.au/.* Retrieved Febuary 5, 2017, from http://www.cio.com.au/article/590834/where-machine-learning-headed-2016/

Naimat, A. (2016). The New Artificial Intelligence Market. *oreilly.com.* Retrieved Febuary 5, 2017, from https://www.oreilly.com/ideas/the-new-artificial-intelligence-market

Nvidia CES 2015 press conference: Tegra X1 [Press release]. (2015, January 5). Retrieved October 11, 2016, from https://www.youtube.com/watch?v=ao47RQvCZwg

OpenChai Overview. (n.d.). Retrieved October 11, 2016, from http://openchai.org. Customers and Features sections.

Russakovsky, O. (2015, January 30). *ImageNet Large Scale Visual Recognition Challenge* (Tech. No. V3). Retrieved October 11, 2016, from ArXiv.org website: https://arxiv.org/pdf/1409.0575v3.pdf. Table 8 and Figure 9.

Smart Phone Manufacturing in China (Rep. No. 4041a). (2016). *www.ibisworld.com*. Retrieved October 11, 2016, from IBISWorld database.

Thibodeau, P. (2014). Data centers are the new polluters. *Computerworld*. Retrieved from http://www.computerworld.com/article/2598562/data-center/data-centers-are-the-new-polluters.html

Tractica (2015). Artificial Intelligence for Enterprise Applications to Reach $11.1 Billion in Market Value by 2024. *tractica.com*. Retrieved Febuary 5, 2017, from https://www.tractica.com/newsroom/press-releases/artificial-intelligence-for-enterprise-applications-to-reach-11-1-billion-in-market-value-by-2024/

Truong, A. (2016, April 26). Why India Could Be The New China For Smartphone Companies. Retrieved October 11, 2016, from http://www.huffingtonpost.com/entry/india-smartphone-market-new-china_us_571f82c2e4b0b49df6a8fe42

VCloudNews. (2015). Every Day Big Data Statistics – 2.5 Quintillion Bytes of Data Created Daily. *VCloudNews.* Retrieved from http://www.computerworld.com/article/2598562/data-center/data-centers-are-the-new-polluters.html

Vieira, A. (2016). The Revolution of Depth. *medium.com*. Retrieved Febuary 5, 2017, from https://medium.com/@Lidinwise/the-revolution-of-depth-facf174924f5#.mansn7eyn

Zhang, S., Choromanska, A. & LeCun, Y. (2015). Deep learning with Elastic Averaging SGD. *NIPS 2015*. Retrieved April 14, 2017, from https://arxiv.org/pdf/1412.6651v8.pdf

Zhao, H. & Canny, J. (2014). Kylix: A Sparse Allreduce for Commodity Clusters. *Proc. Int. Conference on Parallel Processing (ICPP 2014)*. Retrieved April 14, 2017, from https://people.eecs.berkeley.edu/~jfc/papers/14/Kylix.pdf