Scaling Up Deep Learning on Clusters

Aleks Kamko

Scaling Up Deep Learning on Clusters

Master of Engineering Capstone Report
2016-2017

Aleksandr Kamko
Electrical Engineering and Computer Science
Data Science and Systems

Contents

Abstract

The Scaling Up project aims to develop a state-of-the-art machine learning framework that efficiently leverages the power of a cluster of machines. As data becomes increasingly more plentiful (Hilbert 2011), methods for efficiently leveraging computing power to crunch these numbers are becoming more critical. Typical industry datasets are on the order of 1 Terabyte and growing (Canny 2013), making them infeasible to process using a single machine. As a result, developing algorithms and frameworks for training statistical models in a distributed, cluster-accelerated setting is a hot area of research today.

Professor John Canny, our capstone advisor, has developed the BIDData Suite, a machine learning toolkit that expertly utilizes GPUs to achieve record-breaking ”roofline” performance on a single machine (Canny 2015). Our capstone focuses on extending BIDData's statistical models with the ability to train effectively in parallel on a cluster.

Our team has succeeded in developing multiple cluster-enabling modules within BIDData's codebase, including (1) an inter-machine communication framework, covered in Jiaqi Xie's technical report, (2) a network throughput monitor, covered in Quanlai Li's technical report, and (3) several distributed variants of practical machine learning models, covered in depth in Chapter 1 of this report.

Chapter 2 focuses on the issues that arise as a consequence of the growing trends of using machine learning to analyze massive datasets in industry, and how our project aims to alleviate some of these issues. Chapter 2 also provides an analysis of the market strategy for our industry partner, OpenChai, who is trying to bring the benefits of machine learning to lagging enterprise like healthcare and banking.

1 Technical Contribution

1.1 Problem Definition

The Scaling Up project aims to develop a state-of-the-art machine learning framework that efficiently leverages the power of a cluster of machines. As data becomes increasingly more plentiful (Hilbert 2011), methods for efficiently leveraging computing power to crunch these numbers are becoming more critical. Typical industry datasets are on the order of 1 Terabyte and growing (Canny 2013), making them infeasible to process using a single machine. As a result, developing algorithms and frameworks for training statistical models in a distributed, cluster-accelerated setting is a hot area of research today.

Professor John Canny, our capstone advisor, has developed the BIDData Suite, a machine learning toolkit that expertly utilizes GPUs to achieve record-breaking "roofline" performance on a single machine (Canny 2015). Our capstone focuses on extending BIDData's statistical models with the ability to train effectively in parallel on a cluster.

This report will focus on my work in developing distributed (cluster-enabled) variants of the Random Forest, Logistic Regression, and Sequence-To-Sequence statistical models in BIDData. A large part of this work was enabled by Jiaqi, who is the primary developer of the communication framework that enables the machines in our clusters to coordinate with each other.

1.2 Distributed Random Forest

For our first task, we chose to implement a distributed variant of the Random Forest (RF) model. This choice was motivated by the simple yet effective nature of RFs, and also because this model falls under the category of "embarrassingly parallel"[1], discussed below.

RF is a statistical model for classification and regression. An RF consists of multiple weak

[1] A problem that is "embarrassingly parallel" can be broken up into parts and solved in parallel such that no pair of workers need to communicate while solving their respective parts.

decision trees, making it a kind of ensemble method, i.e. an aggregate of weaker decisions.

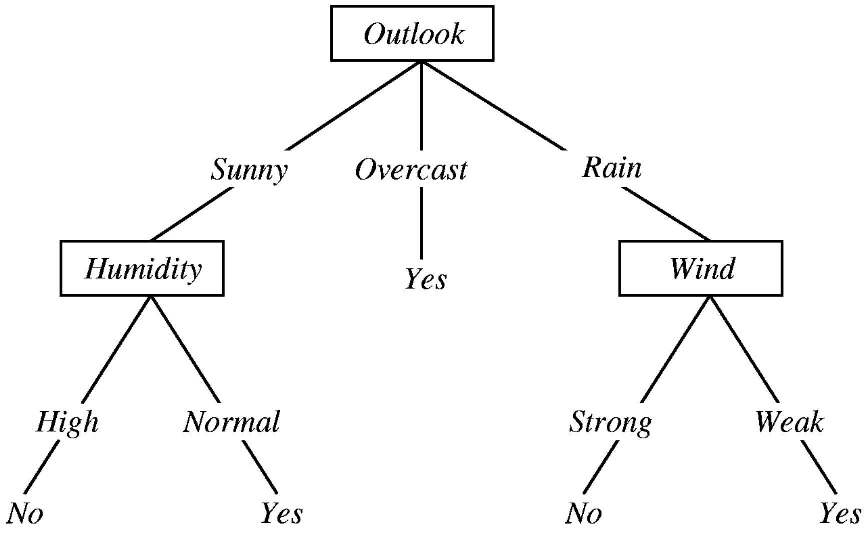

Figure 1: A simple example of a decision tree, the building block of Random Forests.

For a classifying forest, during inference, each decision tree in the forest is fed the same input, and the forest outputs the plurality vote of its decisions as the composite output of the entire forest. Despite their simple nature, RFs are surprisingly effective; they are often used in industry due to their quick inference speed and robustness to outliers in data (Stripe 2016).

Furthermore, Random Forests are relatively easy to parallelize because they are embarrassingly parallel. This means that we can run multiple instances of the Random Forest training procedure on multiple machines without having these machines communicate with each other *at all* throughout the training procedure. We simply place the dataset on each machine[2] and kick off the training process on each machine independently, just as if each machine were training in a single-machine setting. Once all machines finish training, we collect the forests constructed by each machine and concatenate them to construct a larger, aggregate forest with more inference power.

In theory, we can scale the distributed Random Forest algorithm linearly: if it takes 100 minutes for a single machine to train 10 decision trees, then it should only take 10 minutes

[2]Each machine gets the entire dataset; we do not need to divide the data in this case.

for 10 machines to train 1 tree each — assuming the time to distribute the dataset and to collect all trained subparts are both negligible, which is usually a safe assumption.

The effectiveness and "easy" parallelizability of Random Forests made them a good candidate for our first distributed model. Therefore, our challenge for implementing distributed RFs was mainly a systems and data distribution one: we had to become intimately familiar with both BIDData and Apache Spark, the latter being our chosen framework for cluster management and data distribution. Some of the work we did to run the system included: (1) making the existing BIDData Random Forest model serializable so that a copy could be sent to each machine in the Spark cluster, (2) tweaking the memory and processing parameters of Spark to avoid Out-Of-Memory errors and ensure proper data/work distribution in the cluster, (3) writing code to actually distribute the model and data, kicking off the learners on each machine and subsequently collecting all of the trained trees. I have spared the details here since much of the work was cosmetic and not particularly innovative.

Nevertheless, our results for the distributed Random Forests model are promising.

Table 1: Distributed Random Forest training time

Single-Machine	300s
4 Machine	140s
Ideal[3]	75s

Training time for 100 trees at depth 10 on the 201MB Year Prediction dataset (UCI 2011).
(Accuracy numbers are omitted because they are identical.)

Table 1 indicates that we achieve a close-to-linear performance boost on the Year Prediction dataset (UCI 2011) using our distributed implementation. The discrepancy from the ideal linear speed-up is likely due to data distribution time and model collection time, since the size of our benchmarking dataset lies at an unfortunate point where distribution time is

[3]An ideal speed-up would be at least linear. If we use N machines, we would ideally like a $\geq N$ times speed-up.

somewhat significant compared to training time. Prediction accuracy was roughly equivalent for both variants of the algorithm, barring some very small random perturbations. These results prove the feasibility of our distributed Random Forest implementation.

1.3 Distributed Logistic Regression (Distributed General Linear Model)

Following our successful distributed Random Forest implementation, we chose to focus on developing a distributed variant of the General Linear Model (GLM). This second choice was motivated by the wide applicability and popularity of GLM; it is a core component of any machine learning toolkit. Additionally, though the standard, single-machine variant of GLM is simple, developing a distributed variant is considerably difficult, unlike Random Forest models. This posed an interesting challenge for our capstone.

As its name suggests, GLM is a generalization of the linear model, i.e. "fitting a line to the data"[4]. For this report, I will be focusing on Logistic Regression (logit), one of the models described by GLM[5].

Logistic Regression is a probabilistic model that is used when the dependent variable of the data is categorical. In other words, Logistic Regression is used for classification. For demonstration, we consider the 0/1 binary case: during training, logit fits a "best line" to the data according to some parameters. Points on one side of the boundary are more likely to have a true category[6] of 0, and points on the other side are more likely to have a true category of 1 (see Figure 2 below for an example). During inference, a new point is assigned a probability according to how close it is to the boundary. The further away a point lies from the boundary, the more confident we are of that point's true category.

[4]Or, in higher dimensions, a hyperplane.

[5]Our work enables the distribution of any model described by GLM, but I focus on Logistic Regression for brevity.

[6]The true category of a data point is the real, underlying category of the data point which we are trying to predict but do not know in general.

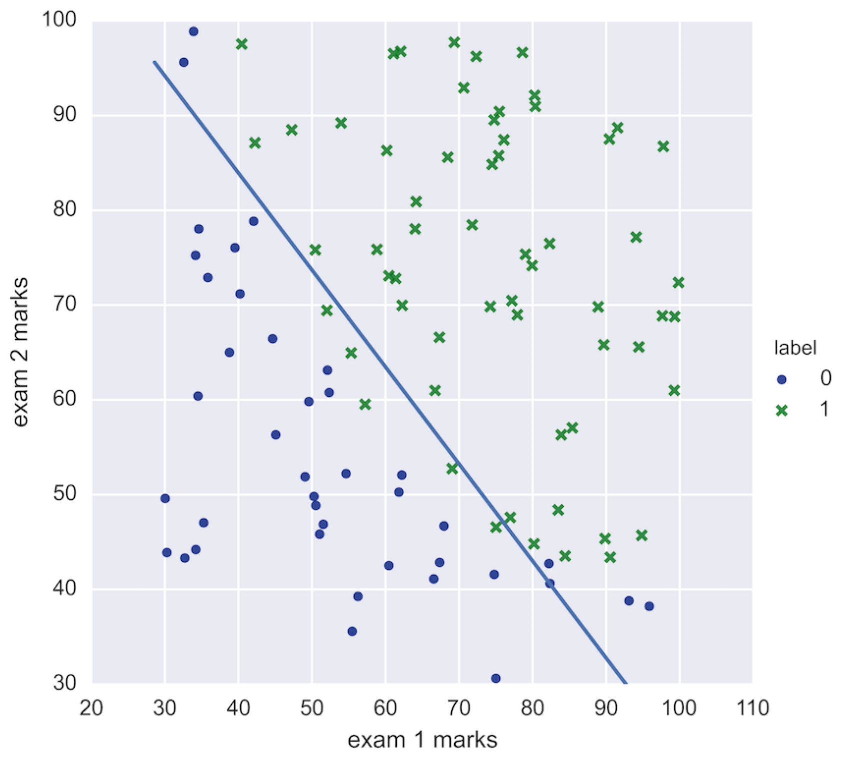

Figure 2: An example of Logistic Regression with 2 independent variables. Fitting a relationship between scores on 2 exams.

One popular use case of Logistic Regression is Click-Through Rate (CTR) Prediction: given information about an online advertisement and a particular consumer, how likely is the consumer to click the advertisement? This question ultimately boils down to a 0/1 categorical problem where we try to maximize our prediction accuracy so that we can serve advertisements more optimally to generate more revenue. CTR datasets present a compelling problem because their size is on the order of terabytes (Criteo 2015); most logistic regression systems would struggle to crunch through a dataset this large. Consequently, our capstone team focused on enabling training for large CTR datasets.

1.3.1 SGD - Stochastic Gradient Descent

Dataset size is a decisive factor for choosing a training method. Small logit models, for example, can be solved optimally via a single matrix-multiplication step on a commodity laptop. However, even medium size datasets in the gigabyte range are already unable to fit into memory[7] on a commodity computer, so we need to a take a different approach that avoids loading the full dataset at once. The most popular such alternative is SGD, or Stochastic Gradient Descent. In SGD, we partition our dataset into chunks called mini-batches and iteratively crunch through them. For each minibatch, we compute the gradient of a chosen loss function with respect to the mini-batch and then update our model by taking a small step in the direction of this mini-batch gradient.

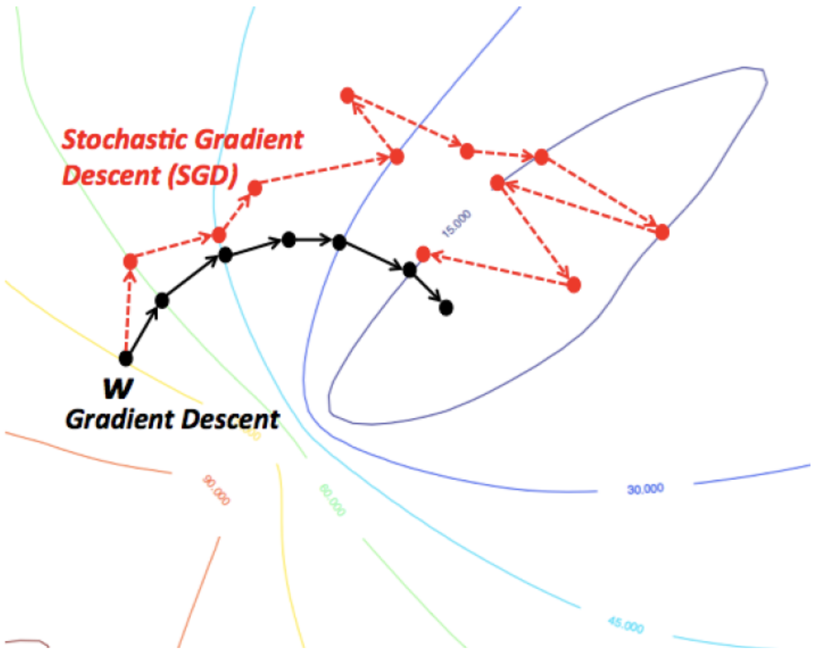

Figure 3: Noisy trajectory of an SGD-based iterative optimization vs. vanilla gradient descent, a much slower approach.

SGD takes the model in a "noisy" trajectory (see Figure 3 above) from the initial model parameters, but is nevertheless proven to eventually converge to the global optimal solution, given some mild assumptions.[8] (Shamir 2015).

But as previously mentioned, modern day CTR datasets are on the order of terabytes. Even with SGD and a powerful processor, taking a single pass through one of these datasets

[7]By "memory" here I mean volatile storage like RAM, not persistent storage like a hard drive.

[8]SGD has become the *de facto* approach for optimizing many statistical models. Its variants are used in everything from K-Means to Deep Learning.

would take many days. Consequently, we are interested in developing a method to speed up our training by processing these datasets in parallel using multiple machines.

1.3.2 Parallelization via EASGD - Elastic Averaging SGD

At first glance, it is not immediately clear how one can train multiple distinct Logistic Regression models and coherently combine their predictive capabilities. Suppose we partition our dataset and fit a best line to each partition separately. How can we then combine these independent line segments? One approach that may seem natural is to take some sort of "average" of these multiple logit models. Fortunately, this takes us a step towards the approach that we ultimately implemented: Elastic Averaging SGD, or EASGD.

In EASGD, we partition our terabyte dataset into distinct partitions, which I will denote "slabs", that can be reasonably handled with a standard SGD approach. We assign each slab to one worker in our cluster and kick off our workers to train on their respective slabs using some variant of SGD. Simultaneously, we start a coordinating master process. The master orchestrates the cluster workers to periodically[9] communicate their evolving models throughout the SGD optimization procedure. At each round of communication, called an *allreduce step*, we compute a global average of all of the workers' models. The master then instructs each worker to nudge its local model parameters[10] towards this average by some elastic factor. We continue this coordination process continuously and asynchronously, until we complete a desired number of aggregate passes[11] through the entire dataset. Finally, we take the average of all of our workers as our trained model.

We chose to use EASGD as our approach for distributing Logistic Regression because it has empirically shown very promising results[12] (Zhang 2015).

[9]The faster, the better.

[10]In the case of Logistic Regression, the model parameters are the slope of the line in each dimension of the dataset space.

[11]A single aggregate pass through the entire dataset corresponds to each worker completing a single pass through its slab.

[12]Although it has yet to be rigorously proven to converge.

1.3.3 Implementing EASGD on top of Spark

As described, the EASGD algorithm requires that the workers in our cluster are constantly communicating. This presents a significant design challenge because Spark restricts workers to only communicate at specific times during the distributed program[13]. To get around this limitation, we had to build an out-of-band communication framework that was able to mesh well with Spark's dynamic worker allocation. The framework is described in detail in Jiaqi's paper, but briefly, it consists of a fault-tolerant BIDData Master process that communicates with multiple BIDData Worker processes on other machines.

With this communication layer in place, I was able to patch the missing pieces in BIDData's EASGD and GLM code and implement a distributed GLM model on Spark.

Table 2: Distributed Logistic Regression training time

	Time (Seconds)	Validation Set Accuracy (AUC)
Single-Machine	1395	0.7808
4 Machine	433	0.7809
Ideal	348.75	

5 passes through the 12GB Criteo CTR dataset (Criteo 2014).

Over multiple trial runs, we were able to achieve a close-to-linear speed-up over single-machine Logistic Regression while achieving near-equivalent prediction accuracy on a smaller 12GB CTR dataset (Criteo 2014). These results are surprising since Logistic Regression is traditionally difficult to parallelize, proving the usefulness of our system on an industry standard dataset. Training on a larger, order-terabyte dataset should be feasible with our system.

[13] Inter-machine communication can be a significant performance bottleneck. Therefore, Spark is communication-restrictive to prevent the programmer from making poor decisions. Our use case is unique, however.

1.4 Distributed Sequence-To-Sequence Model

Our work on distributed Logistic Regression, especially the underlying communication framework, laid the foundation for our next task: developing a distributed neural network model. Furthermore, since EASGD is model agnostic, we are able to use it in this neural network setting. We chose to focus on the Sequence-To-Sequence (S2S) neural model because it is a recently invented and particularly relevant model.

A S2S model does what its name suggests: it takes a sequence of tokens as input and generates a corresponding output sequence. One practical use case of S2S models is language translation, which I will use to give a high-level description of how an S2S works (Figure 4 below provides a complementary illustration). An S2S consists of two Recurrent Neural Networks (RNNs), an "encoder" and a "decoder". We feed an input sentence word for word into the encoder to produce an "embedding" of the input sentence — a matrix that attempts to tightly encode the context of the input sentence. This embedding is fed into the first stage of the decoder, along with a "<START>" token, producing the first token of the output sentence and a new embedding. We then recursively feed the output token and embedding into the further stages of the decoder until it outputs an "<END>" token, indicating the end of the output sequence (Ram 2016).

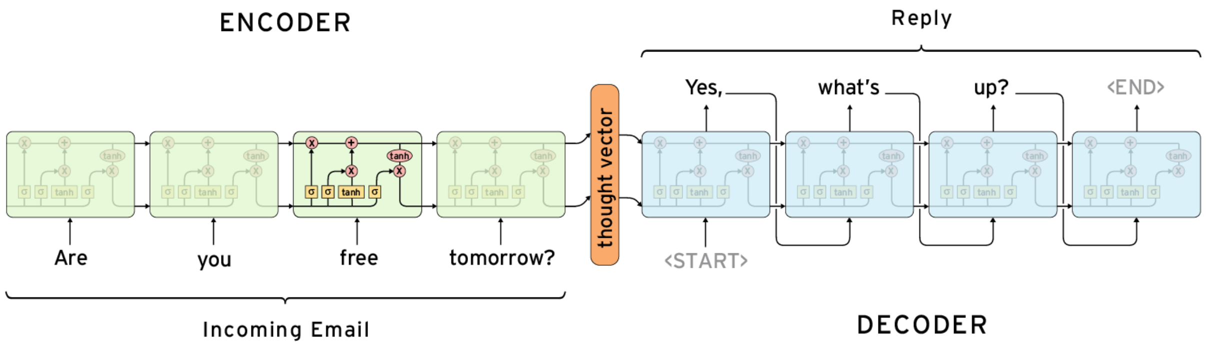

Figure 4: An S2S model for generating email replies. (Ram 2016)

S2S models are a recent and exciting development. In fact, Google Translate now uses a S2S model as its backend to achieve incredibly accurate translation (Le 2016). Other use cases include image captioning, dialogue generation, and word grapheme-to-phoneme conversion.

S2S models present an important problem because most models take days (or weeks) to train on a datasets that are large enough to produce good results. For example, the authors of the seminal paper on the S2S model trained for 10 days on an 8-GPU[14] machine in order to achieve good results on a French-to-English translation task (Sutskever 2014). And despite gradual hardware and software improvements since the Sutskever paper, training time has yet to be significantly reduced. As a result, a method that enables significant training speed improvement for S2S models would be a novel breakthrough.

1.4.1 S2S Results

We believe that our technology is a worthy contender for approaching this problem. Using EASGD for parallelization, we achieve impressive speed-ups in Sequence-to-Sequence model training. Below we summarize our results for training an S2S model on the CMU Pronunciation Dictionary Dataset (CMU 2014):

Table 3: Distributed Sequence-to-Sequence training time

	Time (Seconds)	Word Error Rate (%)
Single-Machine	3004	45.2
4 Machine	919	46.8
Ideal	751	

10 passes through the CMUdict dataset. (CMU 2014)

Some important caveats for these results: (1) we are not achieving state-of-the-art accuracy with this model because we have not yet tuned our model hyperparameters, and (2) data distribution and model collection time are not included in this benchmark because these

[14] Graphics Processing Unit

experiments did not use Spark. Nevertheless, these results presented in the second column are encouraging — BIDData is on the cusp of a novel and practical technique for improving the speed of S2S model training without sacrificing accuracy. Furthermore, we have set the groundwork for experimenting with larger S2S datasets and for running distributed variants of other kinds of neural networks (like Convolutional Neural Networks, used for image classification).

1.5 Conclusion

This report describes our work on extending BIDData to train distributed variants of the Random Forest, Logistic Regression, and Sequence-To-Sequence statistical models. It also describes our use of Elastic Averaging SGD, a contemporary method for parallelizing Logistic Regression, Sequence-To-Sequence models, and other neural models. Using EASGD, we achieve substantial training speed improvements over the respective single-machine algorithms while preserving prediction accuracy.

Extending these models to work in a cluster setting enables users of BIDData to process datasets with order-terabyte size at impressive speed. This was previously infeasible due to the memory and computational limits of single-machine algorithms.

Furthermore, our work is relevant for improving training speed of recently developed but cumbersome models, like Sequence-To-Sequence models applied to language translation.

2 Engineering Leadership

2.1 Project Context 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 $\geq$ 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.

To approach the third challenge, data privacy, our team is partnering with OpenChai to deliver BIDData to the consumer on a secure, standalone platform — a cloud-in-a-box. We finish this chapter with a market analysis of OpenChai's business strategy.

2.2 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).

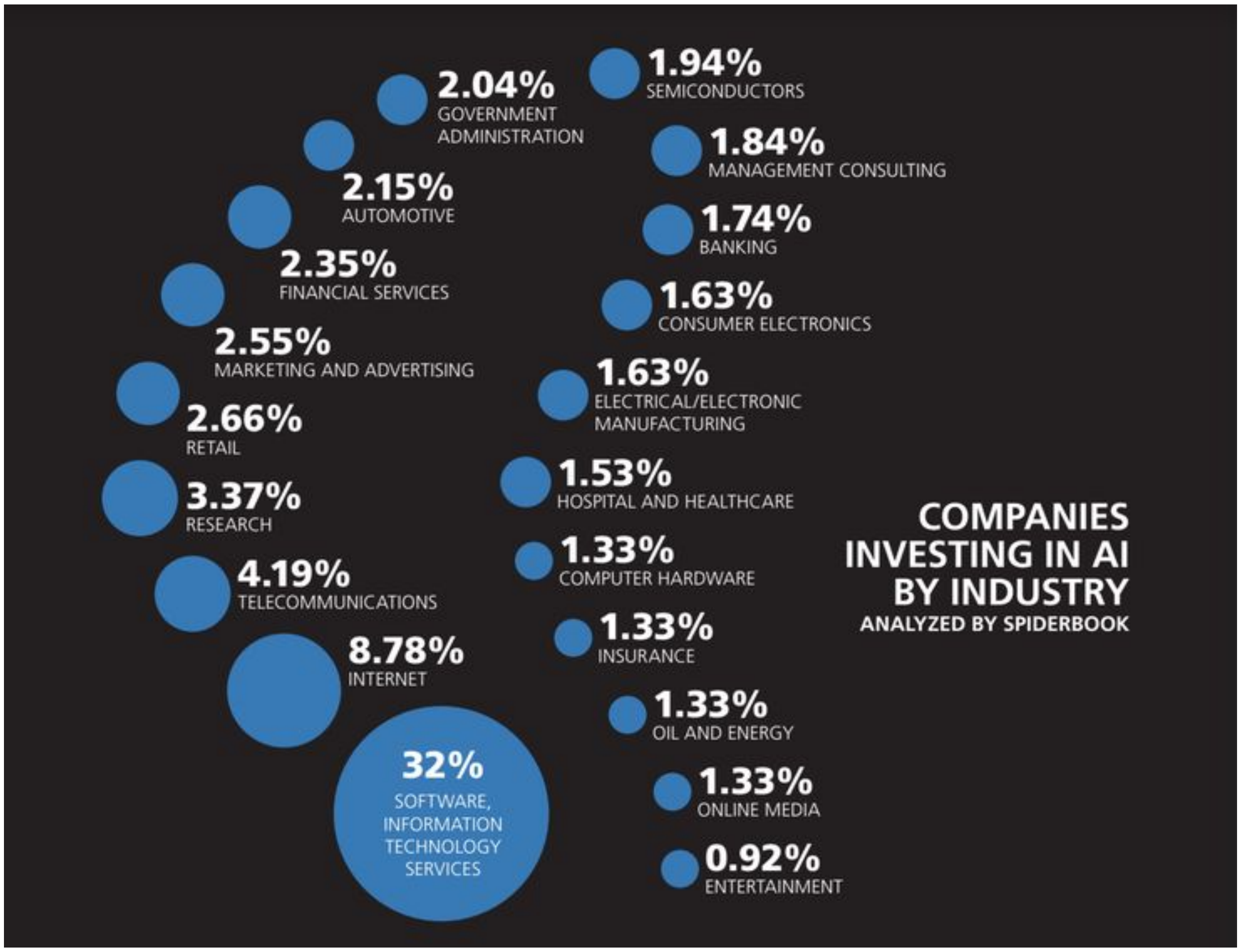

Figure 5: Companies Investing In AI By Industry (Naimat, 2016)

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).

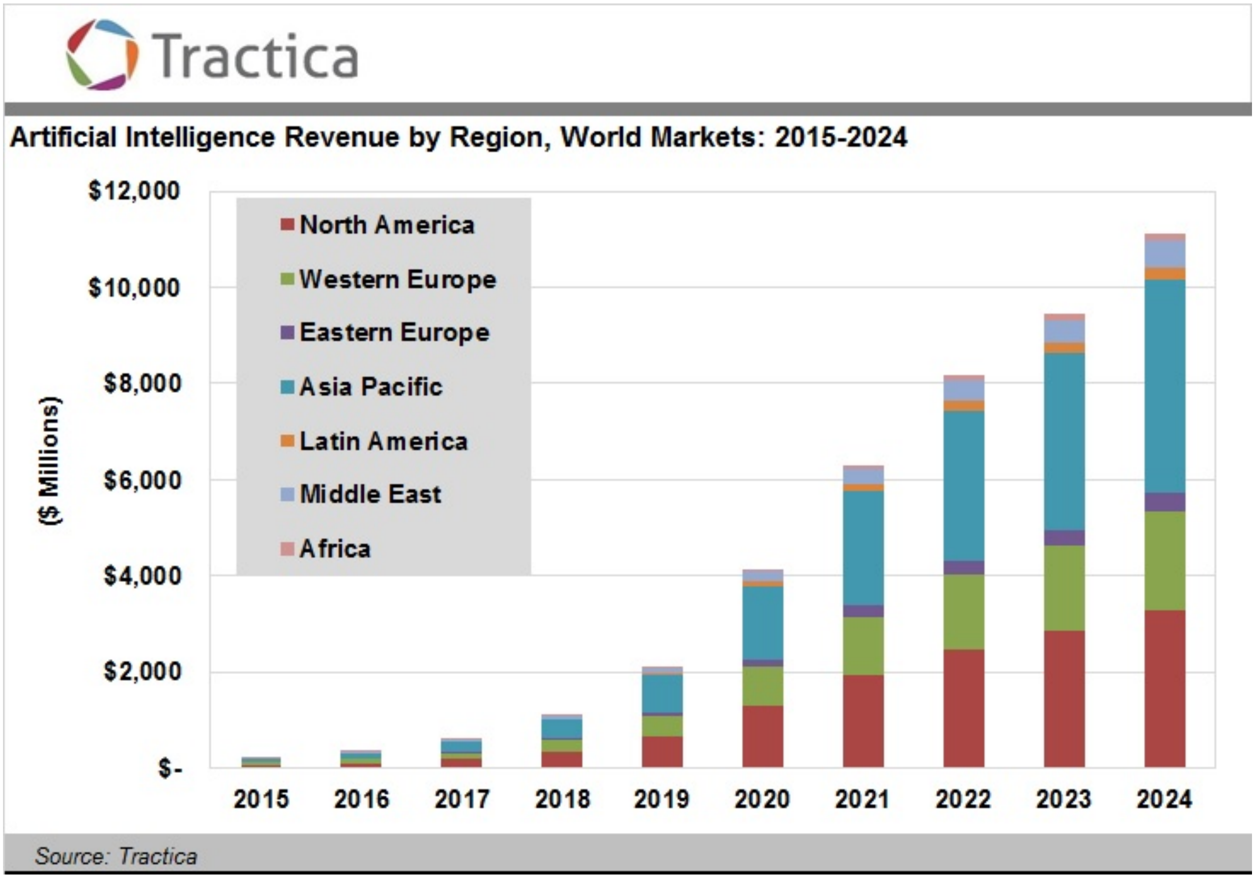

Figure 6: Artificial Intelligence Revenue by Region, World Markets (Tractica, 2015)

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

2.2.1 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).

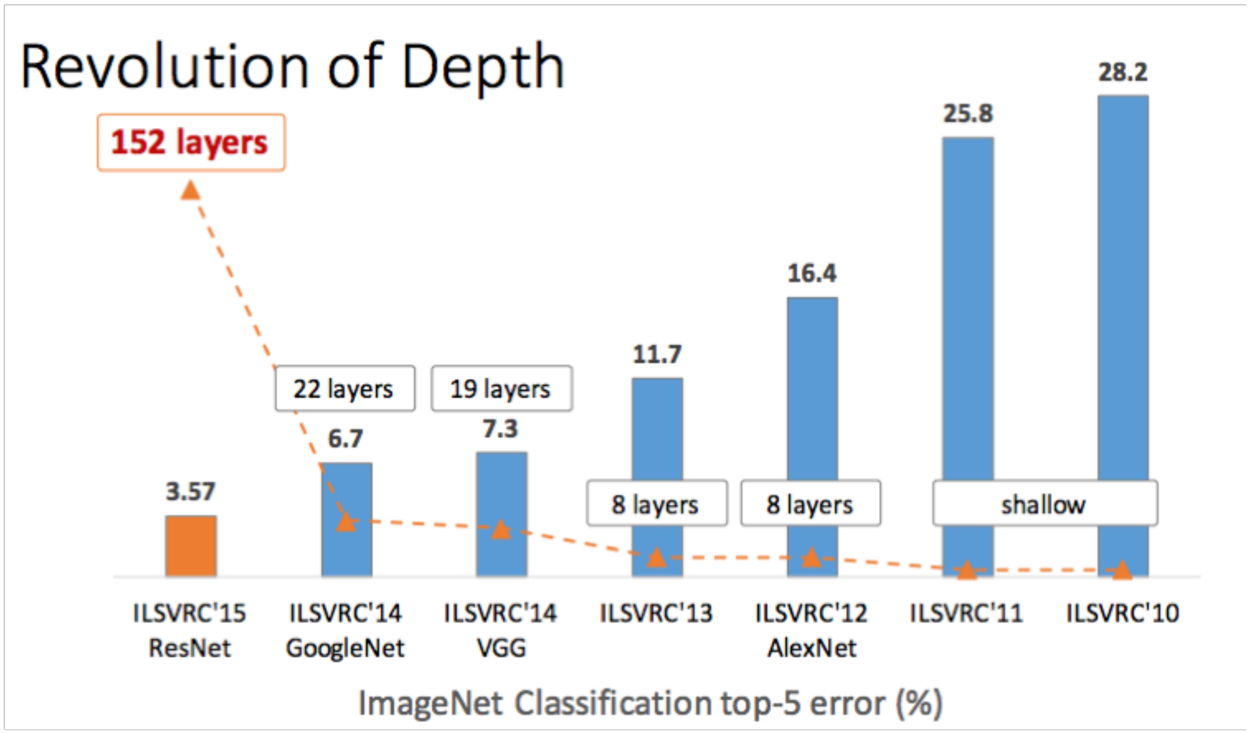

Figure 7: Revolution of Depth (He, 2015)

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[15] 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).

[15]Graphics Processing Unit

2.3 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.

2.3.1 Data Center & Cloud Service Industries

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.

2.3.2 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.

2.4 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[16] 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.

2.4.1 Smartphone Market Analysis

OpenChai's product is only feasible because of 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 8 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.

[16]Graphics Processing Unit

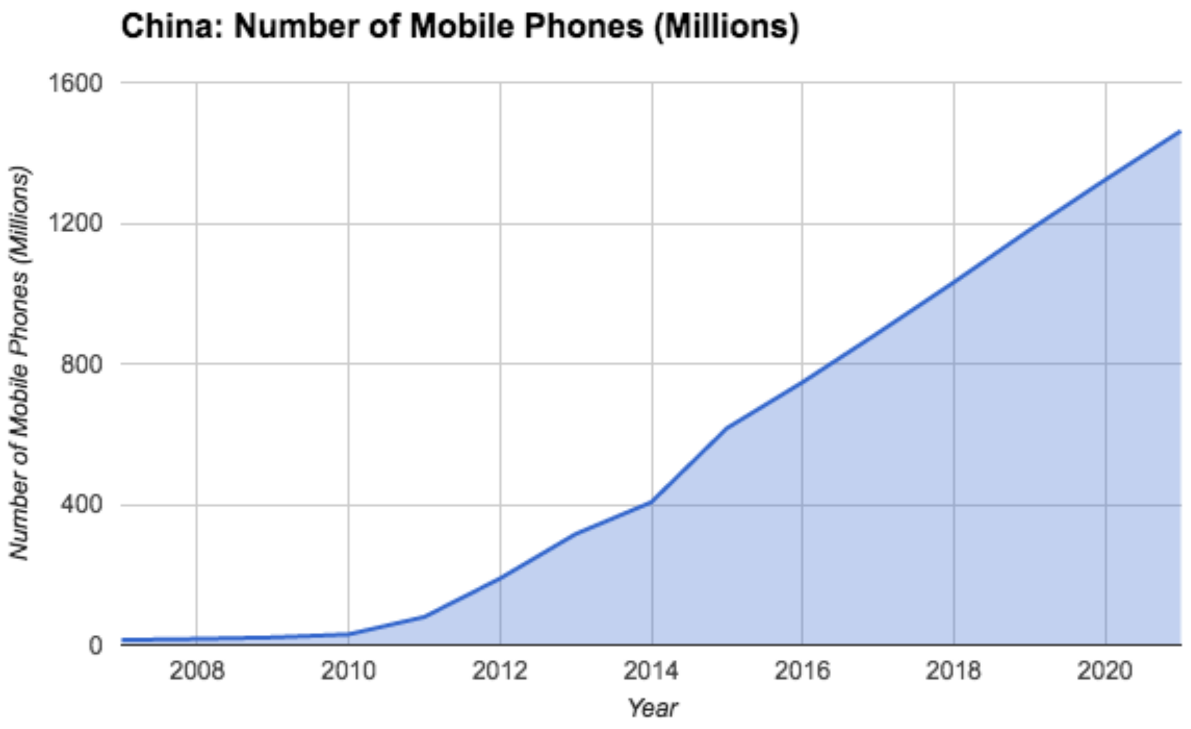

Figure 8: The number of mobile phones in China is growing quickly. (IBISWorld 2016, p.4)

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[17] and GPUs[18], 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.

[17]Central Processing Unit
[18]Graphics Processing Unit

2.4.2 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 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 9 below).

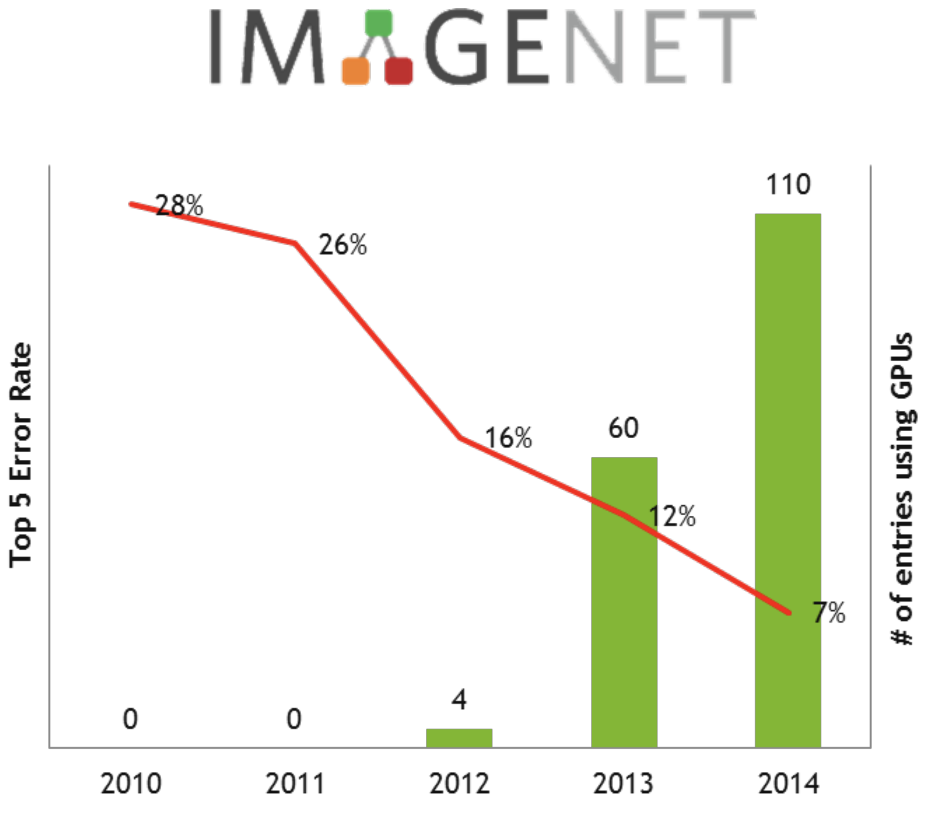

Figure 9: Following the "Krizhevsky result" of 2012, the use of GPUs in computer vision has exploded. (Gray 2015, Figure 2)

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.

2.4.3 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.

Works Cited

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

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

CMU. (2014) "CMUdict (Carnegie Mellon Pronunciation Dictionary)." GitHub. N.p., 2014. Web.

Criteo. (24 June 2014) "Kaggle Display Advertising Challenge Dataset." N.p., Web. http://labs.criteo.com/downloads/2014-kaggle-display-advertising-challenge-dataset/

Criteo. (18 Jun 2015) Criteo Releases Industry's Largest-Ever Dataset for Machine Learning to Academic Community. Criteo. N.p., Web.

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.

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.

Hilbert, Martin, and Priscila Lopez. (2011) The World's Technological Capacity to Store, Communicate, and Compute Information. Science 332.6025 (2011): 60-65. Web.

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

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.

Le, Quoc. "A Neural Network for Machine Translation, at Production Scale." Blog post. Google Research Blog. N.p., 27 Sept. 2016. Web.

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.

Ram, Suriyadeepan. "Practical Seq2seq." Blog post. Suriyadeepan Ram. N.p., n.d. Web. 31 Dec. 2016.

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.

Shamir, Ohad and Tong Zhang. (2015) Stochastic Gradient Descent for Non-smooth Optimization: Convergence Results and Optimal Averaging Schemes. Proc. 30th 2013 ICML Conf.

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

University of California, Berkeley College of Engineering

MASTER OF ENGINEERING - SPRING 2017

Electrical Engineering and Computer Science

Data Science and Systems

Scaling Up Deep Learning on Clusters

Aleksandr Yuriyevich Kamko

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, Electrical Engineering and Computer Science

2. Faculty Committee Member #2:

Signature: [illegible] Date 5/11/17

Print Name/Department: Joseph E. Gonzalez, Electrical Engineering and Computer Science