COMP90050 Survey Report: Large Scale Data Analytics and Distributed Machine Learning

by

Group 38

Saransh Srivastava - 1031073

Emanuela Arrigo - 833250

Waqar Ul Islam - 1065823

Rahul Babar - 937428

in the

Department of Computer Science

Melbourne School of Engineering

THE UNIVERSITY OF MELBOURNE

June 2020

Abstract

This report is concerned with providing a survey review on the topic of 'Large Scale Data Analytics and Distributed Machine Learning'. It is a survey on the research, technologies and systems created to assist the development of the topic. By understanding the history of large data analytics and distributed machine learning, the future directions of the field become clearer. Large scale data analytics is looked at through the lens of challenges with big data and how to overcome them and the research supporting the techniques proposed. The growing field of the internet of things, and the challenges associated with the vast amount of data collected is analysed. Parameter server framework is looked at in detail to understand how distributed ML systems operate and how to improve the systems which enable distributed ML, such as including flexible parallelism. The cutting edge of the field is analysed through the emergence of machine learning as a service and challenges faced with these services. Future directions of the field are also discussed.

Contents

Chapter 1

Introduction

The world has seen a rapid shift in the digital age over the last two decades. While the previous decade was dedicated to transforming offline information online, the present decade has seen a more and more efficient ways of storing that information, compiling, and transforming it into useful information. Various systems have been developed to store data in structured format via tables and charts, unstructured data such as a blog posts via semi-structured format in a key-value pair[1]. Businesses from all walks of life started moving from offline to online mode, to make use of their own information exchange. This led to a huge amount of data being stored in medicine, transportation, telecommunication, aerial, and retail.

While this is still an active area of research[2, 3], many quickly realized that to reach the true potential of a computer system and to make a technological leap is through extracting information from this data. This data was too large or frequent for conventional databases to easily interpret and process. With the ever-increasing power of new computer hardware, big data technology stacks were created harvesting the large volume of data while catering to the demands of variety, velocity, and veracity – the four V's of big data technology. Most recent application domains include online advertisement, computational biology, recommendation system, and other IoT technologies.

The field of large-scale data analytics emerged from this foundation, consisting of specialized algorithms and technologies which we will review in this survey. We will introduce the fundamental blocks of large-scale analytics and then present a recent system survey built on top of these technologies.

Arthur Samuel first coined the term ‘Machine learning’ in the 1950s [4] but the field never reached its true potential due to lack of enough data collection. The rapid development in recent years lead to an unforeseen increase in not only collection but also businesses’ curiosity to leverage the collected data into better decisions. Learning systems emerged as the true winner when it was able to solve problems that were earlier not feasible to solve due to the vast complexities of the problem.

In order to make these distributed datasets available for training of machine learning models, algorithms had to be developed that enable parallel computation, data distribution and are feature resilient.

We discuss some of the most popular architectures in this survey as well as some very promising research solving the challenges faced by them. The structure of the report is as follows. Background theory on the topic of large scale data and distributed machine learning in Chapter 2, the problems with big data computation and scaling and how to solve it in Chapter 3, the internet of things and the large data problems in Chapter 4, parameter server systems in Chapter 5, and machine learning as a services (MLaaS) in Chapter 6. As the successor of clustered distributed machine learning systems, we see MLaas as the cutting edge of the field, and we present our survey on the latest MLaaS comparison of different systems as well. These services allow users to lease the platform without owning any specific hardware and are increasingly becoming ideal for both small as well as big organizations. In the end, we provide active research involving privacy concerns regarding these MLaaS systems. Through exploring the frontier of developments in distributed machine learning and large scale data analytics, we provide insights into the future directions of the field in Chapter 7.

1.1 Team Information

Name	**Student Number**	**Email**
Emanuela Arrigo	833250	arrigoe@student.unimelb.edu.au
Saransh Srivastava	1031073	saranshs@student.unimelb.edu.au
Waqar Ul Islam	1065823	islamw@student.unimelb.edu.au
Rahul Babar	937428	rbabar@student.unimelb.edu.au

Table 1.1: Team and details

Chapter 2

Background theory

2.1 Large Scale Data Analytics Systems

2.1.1 MapReduce

MapReduce [5] is the most notable effort in large scale data processing, and began a data processing revolution. MapReduce consists of two steps: map and reduce. Map transforms an input key/value pair to an intermediate key/value pair based on rules defined by the user. Then the MapReduce library gathers these intermediate values and provides them to the Reduce function whose capabilities are defined by the user. Reduce is provided with the intermediate key and a set of values for the corresponding key and gathers these two variables and creates a potentially smaller set. By using the intermediate values, MapReduce can undertake these functions on data which is large to be stored in memory [5]. It provided a framework for processing large amounts of data on clusters, which supports parallel processing and is a fault tolerant implementation. MapReduce was originally designed for unstructured data, such as for Google's internet search capabilities however can be applied to structured data. Despite not being the most efficient for handling structured data, MapReduce became widely adopted due to the flexibility and the active open-source Apache Hadoop community.

2.1.2 Apache

The Apache environments are crucial to the large scale data analytics landscape, providing open-source large scale data analytics platforms and frameworks.

2.1.2.1 Hadoop

Apache Hadoop [6] is an open-source software framework which is designed for parallelizable large scale data analytics. Hadoop is made up of a distributed file system, HDFS, and a MapReduce engine. Hadoop offers a flexible, scalable approach to data analytics. Hadoop is ideal for running jobs on data which cannot be stored in memory and for performing batch processing. From Hadoop came YARN, (Yet Another Resource Negotiator), which separates the programming model and the resource management infrastructure to schedule per-application components[7].

2.1.2.2 Spark

Apache Spark [8] is a unified engine for distributed data processing, which uses in-memory caching for frequently used data. The in-memory computations is a point of difference for Spark and removed disk overheads found in Hadoop for completing iterative tasks. Spark uses resilient distributed datasets (RDD) which are fundamental in the Spark infrastructure, they are small read only sets of records and is the basis of the idea of perfect fault tolerance. RDD's allow Spark to distribute data over different machines in the network. The Spark framework is built upon MapReduce capabilities and extends them to provide better performance but still maintains the benefits of MapReduce. Due to the parallelizable capabilities of Spark, applications on Spark can run 100 times faster in memory and in some cases 10 times faster in disk compared to Hadoop[8]. These multiple jobs are handled through a master slave model in Spark. A master node is in charge of resource distribution and monitors multiple worker slaves. In case of a failure, the failed task can affect the execution time of other tasks as well. These failed tasks are known as stragglers, large numbers of stragglers can create a significant impact on the system. Speculative relaunch is used to mitigate the situation caused by the stragglers. Spark is a good data flow system however as the parameters of the model are stored in RDD, which is not ideal and the system continuously wants to receive the parameters and update the models, this creates an overhead. Spark predictive modelling can be used to gain good performance, which will give an advantage of 10 to 20 percent as compared to small standalone models[8]. Spark provides a rich variety for APIs in most common languages used for big data such as Python, Java, Scala and R.

2.2 Distributed Machine Learning background

2.2.1 Parallelism

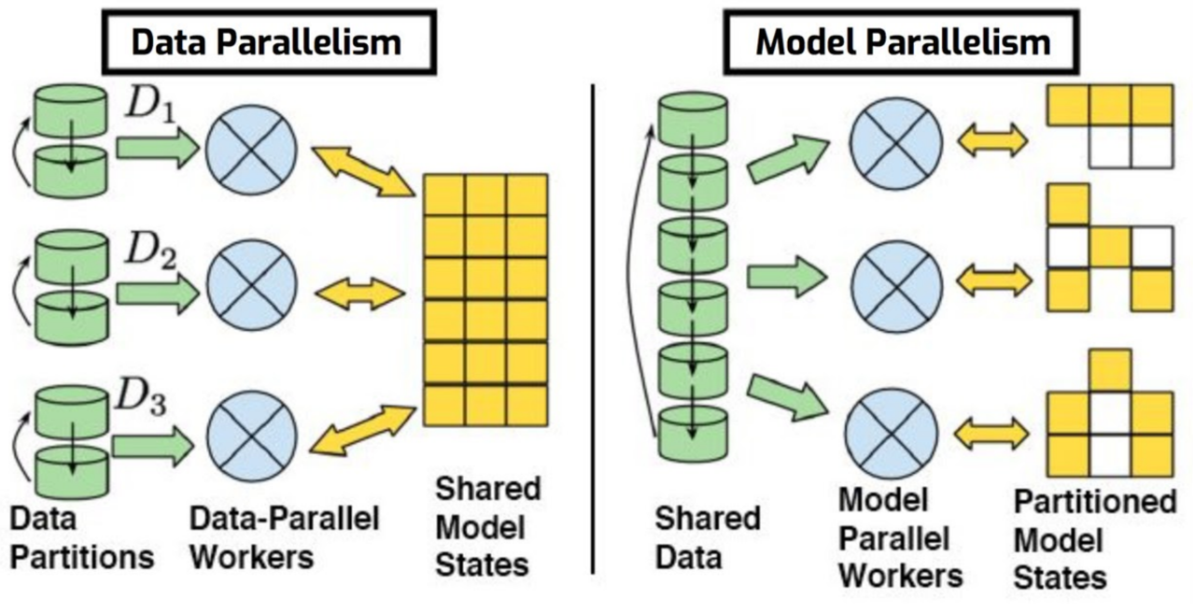


FIGURE 2.1: Two approaches to parallelism. Source [9]

Figure 2.1 visually shows the difference between the approaches to parallelism, data parallelism and model parallelism, the figure is from [9].

The increasing amount of data, produced by us every day is helping answer questions which we could not answer before. It is helping us to make more complex models, which also increases our computational requirements for the machine learning models. Training models such as deep neural networks is not easy without parallelism. Many parallelism techniques have been introduced in the field of machine learning to reduce the complexity of computations required in developing a model. We will discuss some commonly used techniques such as model and data parallelism, along with some advance frameworks introduced to create parallelism in training models.

2.2.1.1 Data-parallelism

Data parallelism is a commonly used technique in various frameworks such as TensorFlow and PyTorch. In this technique, the data is divided into batches and passed on to cores for computation. Each core independently computes the required parameters and after processing their estimates, parameter server aggregates the results. Data-parallel learning models are very useful for large-scale batch computations. However, this technique produces sub-optimal results, when it comes to iterative tasks, such as gradient descent. It creates a bottleneck if the number of parameters is large, which is inefficient for training. Researchers are also working on improving the data parallelism for existing machine leaning models by creating libraries such as MALT [10] which improves the shared memory model and communication between batches to improve performance.

2.2.1.2 Model-parallelism

Model parallelism is another frequently used technique for training machine learning models. Contrary to data parallelism, in this technique same data is divided over different cores. Each core independently computes the require parameters which are different then the parameters passed on to other cores, after computation all the estimates are aggregated by the server parameter. This approach is useful when you have large number of cores in the cluster and you must compute many parameters. This works best for iterative tasks as it reduces communication costs for synchronizing network parameters.

2.2.1.3 Manually Designed Parallelism Strategies

Other than data and model parallelism, experts of the field have also come up with different parallelism strategies with a combination of both data and model parallelism technique. One example is described in [11] where the authors use a combination of both. In this technique for convolutional and pooling layers, the author used data parallelism and model parallelism for connecting the layers. Such strategies have been used by many researchers. However, these strategies may improve results for a specific task but over all they still produce sub optimal solutions, hence not a one size fits all solution.

2.2.1.4 FlowFlex

Distributed machine learning frameworks such as TensorFlow and PyTorch, mainly use data parallelism. But they also provide users to use model parallelism by manually specifying the device placement for each operation. But as described before, a common problem in designing a customize parallelism technique for a system is that one parallelization strategy for one cluster may produce inefficient results on other clusters. FlexFlow [12] helps user's in identifying the best parallelization strategy. For applications such as deep neural networks (DNN), it creates an operation graph along with the topology graph for all the hardware devices available for the computation. Its random search to explore parallelism strategies with an execution simulator which accurately predicts DNN performance helps it produce better results than other parallelization approaches.

2.2.2 Distributed ML Systems

2.2.2.1 MLBase

MLBase [13] is a distributed machine learning system consisting of three parts- MLib, MLI and ML optimizer and developed on top of Apache Spark. It is one of the initial systems which was built keeping both researchers as well as other end users in mind. The system consists of four distinct parts 1) Pig Latin like declarative language to specify ML tasks, 2) Dynamic selection of learning algorithm – optimizer automatically selects best ML algorithm, 3) Iterative re-optimization - a set of high level operators are provided to enable ML tasks. Here the aim is to first provide with an initial solution and to keep optimizing in the background, 4) Distributed run time optimizer for data access patterns.

The aim of this paper is to expose the reader with different use cases this one system will be able to solve. MLBase, with its easy to use API's was able to provide solutions for classification, regression, collaborative filtering, dimensionality reduction, feature selection and data visualization and other exploratory analysis. One of the major advantages of such a system in classification problem was that the user did not need to choose an algorithm or parameters. The architecture of MLBase broadly consists of two parts – Logical learning part (LLP) and Physical learning part (PLP). The LLP is a combination of ML algorithms, feature techniques, parameters and data subsampling strategies. The optimizer tries to prune the search space of LLP to find the best strategy keeping test score vs. time comparable. The PLP stage consists of set of executable ML operations including filtering, feature scaling, sync/async MapReduce tasks and the stage returns overall quality assessment of the model. Because, this system was continuously refining its model in the background, it gave users an early interactive model to experiment.

MLI [14] is an API which was developed to make high-performance, scalable, distributed algorithms easy to develop and is included in MLBase. There is a disconnect between how academia and industry develop ML models and MLI has the goal of bridging the gap between proof of concepts and industry grade ML software. It is also addressing making these ML solutions scalable with the increased use of distributed data. Traditional data analytic systems such as MATLAB and R have not kept up with the demands of large-scale/distributed ML. MLI provides a solution to common ML problems such as feature tuning, model training and testing. The MLI interface consists of MLTable and LocalMatrix[14], each of which have their own API, and are system-independent, being able to be developed locally or in a distributed setting. MLI provides pre-defined interfaces for Optimization, Algorithms and Models. Developers can create their own protocols by using objects to build custom Optimizers, which are used by Algorithms to build Models. MLTable is how one loads data and feature extraction. MLI supports unstructured data and semi-structured data, which can then be transformed and used for modelling within MLI. LocalMatrix is how MLI applied linear algebra operations on the featurized dataset[14].

2.2.2.2 GraphLab

Due to the nature of MapReduce, not all ML algorithms suit it's assumptions and framework of the 'map' and 'reduce' steps, for example sorting is quite complex to implement in MapReduce. GraphLab [15] was developed especially for the sparse data structure and patterns in ML algorithms. GraphLab is a distbuted ML framework focused on iterative graph structure computations. The main element of GraphLab is the high level data representation, a directed data graph. The data graph consists of vertices and edges, denoted by G = (V,E,D) and the data relating to vertex i is donated by D_i, and data relating to the edge connecting i and j denoted by $D_{i->j}$.

The user defines the functions to use on the graph structure, there are update functions which are able to transform the data stored in the graph, and sync functions. Data consistency is handled through three consistency models, which the user chooses based on the needs of the program. GraphLab operates synchronously, meaning simultaneous updating of all vertex values. Alternatively, asynchronous systems update parameters when needed based on previous parameter values. Each vertex has the ability to read and write to adjacent vertices and edges in the graph due to GraphLab's use of sequential shared memory abstraction. However, due to structural constraints of GraphLab, it is unable to be generalised to situations which do not suit a graphical representation, such as automatic video segmentation [16].

2.2.2.3 Distributed GraphLab

Distributed GraphLab[16] is an extension of GraphLab to a distributed setting, it is a graph-parallel distributed framework. It makes GraphLab distributed through the shared memory feature of GraphLab, the creation of a distributed data-graph and relaxing scheduling requirements. Distributed GraphLab also introduced fault tolerance handling through a distributed checkpoint mechanism. The development of Distributed GraphLab addresses the lack of systems which handle asynchronous, dynamic graph-parallel computations. The introduction of asynchronous computations is a great benefit of Distributed GraphLab.

Chapter 3

Big Data Computation and Scaling Problems

The advances made in the big data retrieval field, has made a once challenging task of collecting and processing large volumes of data, a relatively simple task. Where this huge amount of data opens opportunities for data scientists, researchers and the like, it also brings computation and statistical challenges, when machine learning techniques are applied on the data as whole. The traditional machine learning algorithms can only be applied to a complete dataset as the datasets are scaling rapidly, it is less convenient and sometimes impossible to work on the entire dataset. Therefore, there is a need of developing systems that are scalable and can perform parallel processing.

3.1 Large Scale problems

Researchers are working on different techniques to address these large-scale computational problems. One method is partitioning the data along the feature space and applying the parallel block coordinate descent algorithm. Matrix decomposition is another methodology that can also be used in which sample space is segmented into small parts. The combination approach is suggested by Guo et al. (2016)[17]. This method includes decomposition of matrix and parallel block coordinate descent (PBCD) algorithm to make distributed computation for many of the most popular learning algorithms. This is a decentralized approach towards the computation but the result it produces is similar to its centralized solution.

The major problem in processing large-scale data is the big sample and feature space. The approach suggested by the authors in [17] is to decompose the feature space by using parallel block coordinate descent algorithm. This will decrease the computational cost of many famous machine learning algorithms such as SVM by Burges [18] and Logistic Regression [19]. The data sample is also huge, so for that we should apply this new method introduced by authors [17] for matrix decomposition and combination to generate the results.

Experiments on large datasets showed that this technique achieves better convergence rate as compared to other distributed learning algorithms. This method works for both regression and classification tasks.

3.2 Distributed machine learning problems

The distributed machine learning can be performed by using two different methods. Both methods have their positives and limitations, the first method uses a decomposition of first order derivative information, such as gradient,in order to convert the conventional machine learning algorithms, in distributed setup by Zinkevich [20]. When dealing with multiple constraints they use dual decomposition which is a distributed optimization technique [21], alternating direction method of multipliers (ADMM) [22] to merge the constraints into objectives and perform decomposition after it. As an example, we can see the finding of Mateos [23] in which he proposed a distributed algorithm for sparse linear regression. He first explained linear regression as a convex optimization problem and then used ADMM making the process distributed. This method can be generalized in terms of application domain, however its convergence rate is very low and in some cases it fails to converge.

The second method is about working on a specific structure of the problem in order to get an efficient distributed algorithm. As the authors were working on a specific nature of a problem, it helped them achieve the advantages of its convexity. This method has a high convergence rate, although the solution cannot be generalized in terms of application domain due to its specific nature.

3.3 Matrix inversion

The technique discussed in the paper [17] presents a solution which can remove the shortcoming of both methods. This technique uses the first and second order information such as the inverse of Hessian Matrix which makes it faster and does not deal with the specific structure of the problem making this solution applicable to a wider range of applications.

Matrix inversion approach described in this paper [17] reaches a lower bound which gives an efficient way of matrix manipulation. Large scale matrix inversion is a computational challenge which is addressed by using this technique.

A way to solve the equation in a distributed fashion and efficiently is describe below:

$$minimize\ \beta \sum_{i=1}^{N} \epsilon^2 subject\ to\ X\beta - Y = \epsilon \tag{3.1}$$

3.3.1 Steps

3.3.1.1 Divide and conquer

In this case m is the data size and n is a number of features of a dataset. According to their assumption both values are huge. To perform a divide and conquer strategy the author used a technique explained in the parallel block coordinate descent (PBCD) algorithm [24] and its convergence proof and rate is provided by Tseng in his paper about Convergence of a Block Coordinate Descent Method for Non-differentiable Minimization [25]. The matrix of data is divided into multiple columns and rows, where each column is denoted as $X = (X_1, X_2, X_3..., X_{N1}); X \epsilon \mathbb{R}^{m*n_i}$ and the rows are denoted as $\beta = (\beta_1^T, \beta_2^T, \beta_3^T ..., \beta_{N1}^T)$. In parallel block coordinate descent method, we assume that all block parameters will remain constant while we update the current parameter incrementally in every cycle.

3.3.1.2 Matrix Decomposition

We need to compute the inverse of the matrix and for that a new approach is proposed by the authors [17]. For computing efficiently a distributed approach will be used where all matrices will be divided into N_2 computing parts, as we know m is very large for each

X_i so it is impossible to load a complete matrix in the memory whereas n is small for each matrix so we can calculate the singular value decomposition(SVD) by Zhang [26] for each matrix easily. Each matrix contains the whole feature space for each data sample. After calculating the singular value decompose, we get

$$M = \sum_{j=1}^{N_2} V_i^j \Lambda_i^j (V_i^j)^T \tag{3.2}$$

Where V is orthogonal matrix, Λ a is a diagonal matrix with non-negative diagonal elements and M is the summation of each inverse element.

3.3.1.3 Matrix Merging Process

For merging the matrices the authors first represented what they have in the form of equation below: If we can represent M in such a form:

$$M = \ V\Lambda(V)^T \tag{3.3}$$

where Λ is a diagonal matrix with positive diagonal values and V is an orthogonal matrix, then, it is easy to compute the inverse of M. Before transforming Equation (4.2) into the form. By using a theorem which proves that if a matrix A is positive indefinite then A congruent to the identity matrix if A, we can merge the matrices. Thus repeating this step to merge all the remaining matrices.

3.3.1.4 Experiment Settings

The approach described above has been tested in a distributed environment using Hadoop against the performance of two different benchmark algorithms on task of regression and classification. The two datasets selected for classification is the Reputation dataset [27]. It contains 3,231,961 number of attributes and 2,396,130 total number of instances. The second dataset used for the classification is UJIIndoorLoc dataset [28] which contains 21,048 instances and 529 features. Likewise, two datasets are used for regression. The first dataset contains 53,500 instances with 386 attributes that is data Relative location of CT slices on axial axis dataset [29]. The second dataset is BlogFeedback dataset [30] which contains 60,021 instances and 281 features. The approach discussed in this paper [17] is tested against two benchmark algorithms. For the classification the first algorithm that is used was Consensus-Based Distributed Support Vector Machines (CB-DSVM)[31]. The second algorithm used was Parallelizing Support Vector Machines on Distributed Computers (P-SVM) [20]. Same is the case with regression which was also tested against two benchmark algorithms. For regression the first algorithm used was Distributed sparse linear regression (DS-LR) [23]. The second algorithm used for testing was Finito [32]. As every canonical distributed machine learning algorithm converges, the author compares the speed at which their approach converges with respect to others.

3.3.1.5 Experimental Results

As you can see in Table 3.1 that approach discussed in this paper achieves better test error performance and system execution time per cycle. All P-SVM,CB-SVM, Finito and DS-LR are slower at convergence than this novel approach.

Classification	Authors	P-SVM	CB-SVM
Dataset 1	6.46	18.56	25.85
Dataset 2	2.10	4.89	5.70
Regression	Authors	Finito	DS-LR
Dataset 1	0.53	4.55	2.43
Dataset 2	0.48	4.42	2.58

Table 3.1: Results [17] show the comparison between different approaches

Chapter 4

Internet of Things

Internet of things (IoT) is a technology, which is a network of all the devices or things in other words, which facilitates cutting-edge solutions. These devices or things could be anything from a car to a milk carton in a supermarket. Large scale data analytics is majorly used in IoT for processing. IoT is a vast concept but if it gets implemented, its usage could be revolutionary. A tremendous amount of work has already been done which can give a clear background of what potential it holds, along with its security and privacy challenges. There has been a wide range of establishments and its results have been quite astonishing. There are researchers all around the world, making this technology better, to cover almost every 'thing' and turn it into the Internet of Things. IoT uses quantum and nanotechnology to enable storage, sensing, and processing speed[33] and this makes these technologies an important field of research. The best example that we see in our daily lives is the application of the Smart Home System (SHS)[34]. Sensing the owner coming home, turning on the lights and heater/AC, opening the garage door, and so on are a few things that we can experience with SHS. IoT could not only be used in our personal lives but also the economic and industrial areas. However, using IoT in these areas could cause the greatest challenge and that is security.

IoT can be defined in various ways. Based on various definitions we can say that IoT is a huge network of each item, that shares information as a way of interaction to reach a decision. A hypothetical example could be the interaction of items in the refrigerator, which interacts with the user's mobile phone to add milk to the shopping list, as it is finished. This information exchange is done through the cloud. Now, this cloud network could be placed far from the user and hence, arises the challenge of the round trip time. For the example stated above, it is affordable to exchange information in its own time but if there is a network of all the cars being connected on a highway, there is a need to exchange information almost instantaneously. Latency plays a big role in this particular example and hence, fog computing can come in the picture which was created by IBM. We can define fog computing as the layer which sits in between the IoT device and the cloud. This reduces the round trip time to trade information and hence, would enable faster information processing[35]. Fog computing is used for trading smaller bits of information, just like an express counter in a supermarket. Therefore, when there is a need for extensive processing, the information has to be sent to the cloud[36].

4.1 Challenges

Whenever there is a discussion about IoT, its challenges or issues have always been a part of it. In the year 2015, due to the security issues of cloud computing, about 70% of the companies did not accept its implementation[37]. As fog computing is derived from cloud computing, its challenges remain the same[38, 39]. One possible technique an attacker can use for intruding in a fog or cloud network could be by using the Rogue cloud/fog node. This is an attack in which the intruder takes the place of a cloud or fog node and masquerade as a legit network node. Given in Table 4.1 describes the challenges associated with IoT and fog computing.

4.2 Inclination

Internet of Things (IoT) solely depends on exchanging information through cloud computing. This cloud computing consists of various nodes which could be a base station, wireless access point, set-top boxes, or even a simple router[40]. These devices are nothing but a medium for the information transfer and to do so, it uses the resources for computational processing. Now to process this information at the fog layer, it requires the resources to have a comparatively greater processing power[41, 42]. This could then facilitate and the benefits such as lower latency, lower change of intrusion as the area coverage is not wide enough, and so on. Equipping the fog nodes with such processing power could turn out to be a challenge and in this case to such a large extent, it could be expensive. Setting up this kind of network is a task that could take years and years to set up. Even after setting it up, there is an obligation of maintenance of the installed devices. Maintaining fewer devices regularly is affordable but maintaining such a huge

Challenges	
ISSUES	DESCRIPTION
TRUST	As the network is enormous, it is really hard to tell which device could be trusted. As the information is exchanged almost instantaneously, if it is shared with the wrong device, it could have serious effects.
DATA PRIVACY	Handling the authentication for such a network is not easy. Hence, there arises the challenge of accurate authenticity.
COMPUTATION POWER	If the information or data is once sent from the user's end, it would be almost impossible to reverse it if that particular information is being used instantly. Therefore, secured data storage is one of the key aspects of successful implementation.
LATENCY	The user must be able to access their information on the cloud reliably. There is a possibility for the man in the middle attack, thus can put the user in a threatening position.
ACCESS CONTROL	Network security has to be made with the state of the art technology. If there is an intrusion in the network, its effects could be colossal. For example, imagine the cars on the motorway exchanging incorrect information. This could lead to a fatal disaster.
SECURITY	The integrity and correctness of the data is hard to verify. If the information sent is incomplete or misleading, there is a chance that the system could make an inaccurate interpretation out of the provided information.
INTRUSION	The network security architecture must be fortified by using intrusion detection techniques. These techniques would be an important aspect to keep the network safe. If there is a breach in the system, the whole network could fall apart.

Table 4.1: Cloud and fog computing challenges

network regularly could be both time consuming and costly facilitating continuous work on hand[43]. There is a possibility that these fog nodes could be stolen physically and tap some unethical information to it. If it is a wireless connection, the route traffic could be attacked and if it is a wired connection, it is even more dangerous as the connection could be easily interfered with. A possible workaround this could be by using the user's mobile device as a cloud or fog node. By doing this, huge monetary savings could be achieved. A mobile device is already equipped with enough data storage space and other components such as RAM and a processor. A lot of research has been conducted to enable large scale data analytics using mobile phones. It is just the initialization cost that comes into the picture here on the mobile device, to enable it to be used as a cloud node.

4.3 Applications

4.3.1 Economy and Health-care

IoT is vastly being researched on to use it in the field of economy and health-care. Economic development, water quality maintenance, well-being, industrialization are some of the fields which can use IoT but are not limited to and are also striving for social, health, and economic goals of the United Nations advancement step[34]. IoT can monitor health problems such as blood pressure, diabetes, depression, obesity, and so on, which can turn out to be of utmost importance.

4.3.2 Vehicles and Smart city

Concepts such as smart cities, smart homes, and smart transport are now possible with IoT. Various technologies such as machine learning enables the system to continuously improve and make itself better. Cloud and fog technology directly facilitate the implementation of the concept of a smart city. Green technology is another field in which IoT could play an important role. Few of the big force vehicle companies such as Tesla, BMW, etc., are already using this technology to continuously monitor vehicle parts such as the engine, tires, oil, and other parts. Implementing IoT to manage traffic on the road could turn out to be groundbreaking as it would promote effortless and safer road ethics.

4.3.3 Industrial and agricultural automation

With an ever-increasing population, it is important to keep up with the increasing need for food. Hence, IoT comes into the picture to make it happen. Greenhouse technology helps the crops by providing appropriate climatic conditions, thus enabling the increase in its production. Doing it manually, would not be the optimal option and hence, IoT devices could be used here to continuously monitor the growth of crops and changing the requirements accordingly. Nowadays, industries are inheriting automation of machinery, and IoT is the one enabling that. It has affected a lot of areas such as stock management, quality check, logistics and supply chain optimization and management, and so on, positively.

4.4 Critical Analysis

Cloud and fog computing can make use of the large scale distributed analytics by making faster and accurate processing. By using a simple approach of using a mobile device as a node can drastically simplify the process. We tried to make a comparison of the three types to see how one can overcome the shortcomings of the other, based on our understanding. We have used good, average and, bad as a parameter to compare them. The comparisons are seen in Table 4.2.

Comparison			
	Cloud	**Fog**	**Mobile as a node**
DATA PRIVACY	Average	Average	High
COMPUTATION POWER	High	Average	Average/Low
LATENCY	High	Medium	Low
ACCESS CONTROL	Low	Medium	High
SECURITY	Low	Medium	High
INTRUSION	High	Medium	Low
TRUST	Low	Medium/High	High

TABLE 4.2: Comparison of cloud computing, fog computing and using mobile as a node

IoT as a whole can make lives easier and safer for humans. We are already using IoT in our daily lives but once it is implemented on a large scale it can make the world a better place.

Chapter 5

Parameter Server Framework

Distributed optimization is required to execute machine learning algorithms on large scale data. Traditional machine learning algorithms on single machines are unable to efficiently train models with millions of parameters. Since machine learning requires parameters and dividing the workload between worker machines, meaning worker machines have to share the updated global shared parameters to complete local computations to improve the model. The inter-machine data transfer uses valuable bandwidth and causes bottlenecks in the refinement of models.

An iterative convergent solution refers to models and algorithms which repeatedly update the model parameters until the parameters don't change (meaning converge). Common examples of iterative convergent algorithms include stochastic gradient descent (SGD) and alternating least squares (ALS). Iterative models include logistic regression, support vector machines, k-means, and neural networks. These algorithms and models are extensively used in practice in ML, however due to the global parameters have been difficult to translate to the distributed setting.

In Chapter 2, some general ML systems were touched upon, this chapter is focused on the distributed machine learning systems which use a parameter server architecture, however there is a comparison of the systems in 5.4 which includes systems included in the Chapter 2.

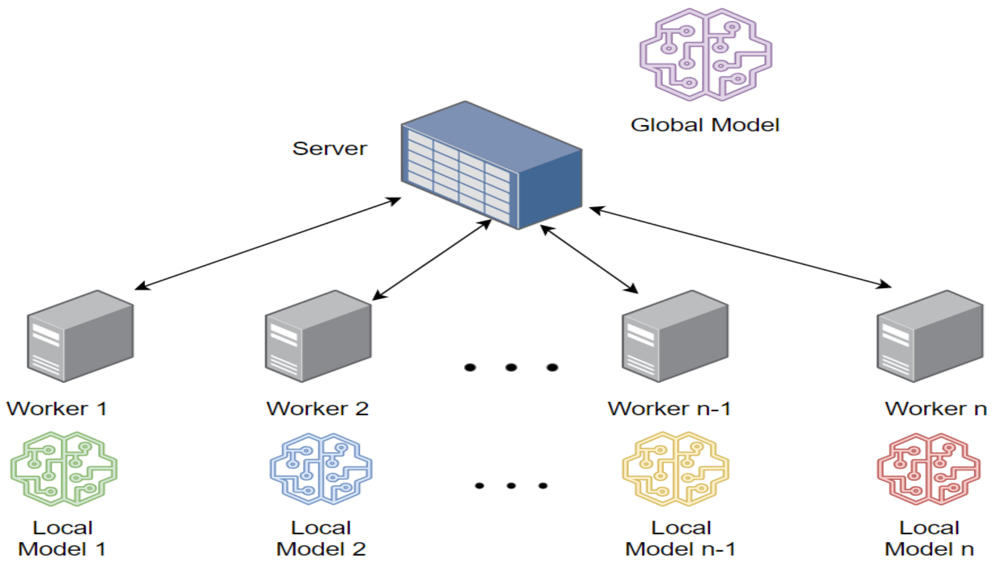


FIGURE 5.1: Parameter Server Framework Structure

5.1 Overview

Parameter server (PS) framework aims to address these problems to make scalable distributed machine learning possible. Figure 5.1 shows the structure of the parameter server framework, with workers maintaining local models on their section of the training data and integrating them with the global model. The servers provide storage of the calculated parameters. This framework is well suited to the repetitive nature of iterative convergent solutions.

Outlined in Li et al. (2014)[44], a PS based framework was developed for efficient machine learning (further details in Section 5.2). The algorithm proposed in Li et al. (2014) [44] solves parallel and asynchronous proximal gradient methods for non-convex and possibly non-smooth problems and defines conditions for convergence of such a system to a critical point. The motivating example provided is for general regularized optimization:

$$\underset{x}{\text{minimize}} \sum_{n=1}^{N} f_n(x) + h(x) \tag{5.1}$$

subject to $x \in \mathbb{R}^d$. The algorithm builds upon the proximal gradient methods by adapting it to spare high dimensional data. The first element is the empirical data loss and h(x) is the regularizer to be used.

Since the publication of Li et al. (2014)[44], parameter server frameworks have been improved upon that proposed. In Aytekin et al. (2016)[45] the ideas in [44] were extended to an 'asynchronous proximal incremental aggregated gradient' [45] algorithm to be implemented on a PS framework. This proposed solution solves problems of the form of Equation 5.1. This solution is an extension of that in [44] as it can handle general convex regularizers and convex constraints not just non-convex problems.

5.2 PS Systems

The PS structure has been utilized in numerous large scale distributed machine learning systems such as Parameter Server, PMLS, FlexPS and even TensorFlow.

In parameter server architectures, there are three consistency protocols used: Bulk Synchronous Parallel (BSP)[46], Stale Synchronous Parallel (SSP)[47], and Asynchronous Parallel (ASP). BSP updates parameters after every iteration hence every worker can see the updated values. BSP is the distributed implementation of the sequential algorithm (waiting for one stage to be completed until moving to the next stage) used on single machines. However this leads to problems with stragglers in BSP, which SSP addresses this problem through allowing faster workers to be m iterations ahead of the slowest workers in the system. ASP doesn't have any synchronisation requirements, it acts like an SSP with m set to ∞, whereas BSP is an SSP with m set to 0.

5.2.1 Parameter Server

From Li et al. (2014) an open-source implementation of the parameter server framework, Parameter Server was developed [48]. The Parameter Server architecture has both a server group and several worker groups, made up of several machines. In the server group, each machine is responsible for a portion of all the global parameters, with replication and migration of parameters being undertaken by the servers ensuring reliability and scalability. Worker machines are allocated a subsection of the training data, to compute local gradients or other measures of the data. To allow for communication delays, asynchronous tasks are used by overlapping communication with computation. Parameter Server offers continuous fault tolerance with the use of well defined behaviour of vector clocks, and can recover the system within one second of non-catastrophic failures [48].

5.2.2 Petuum/PMLS

Petuum [49], now called PMLS, is a general purpose framework to do distributed machine learning. It provides support for a wide range of machine learning algorithms and capabilities. Developed to address shortcomings of Hadoop such as it's difficulty of implementing important ML properties such as error tolerance and GraphLab being graph based limits what can be built in GraphLab, but the asynchronous property of GraphLab is desirable.

PLMS aims to exploit the repetitive nature of ML with a focus on making iterative convergence tasks more efficient. Each iteration updates the state of the model and produces a partial result which then updates the state of another model to create an output result. A worker thread requests the parameters in order to perform the required computation. Whereas the server thread is in charge of storing and updating the model. Server threads are responsible for all communication between the different threads.

A difference of PMLS compared to other dataflow systems such as Spark which uses BSP model is that PMLS uses SSP. Using the SSP model helps PMLS to execute the fast threads regardless of any slow threads or threads which failed to execute, decreasing the impact stragglers have on execution times.PMLS uses checkpoints, in case of any failure, you can resume the system from the last iteration saved in the system. It is more reliable than the Spark implementation. PLMS has a parameter server based module, Bösen, which operates at a data-parallel level, hence its inclusion in this chapter.

5.2.3 Multiverso

Multiverso [50] is a module which is a PS system included in Microsoft's Distributed Machine Learning Toolkit (DMLT). Multiverso operates with distributed key-values, using a globally shared table stored in the servers, where workers have access to the shared table and can use Get and Add operations. Multiverso differs in how it separates the workers and servers to different processes. By doing this, data can be loaded and trained in parallel. However, SSP is not supported in Multiverso, which is important for some implementations.

5.2.4 FlexPS

A shortfall of the above PS systems is the lack of support for altering the parallelism during run time. For dynamic workloads this feature is crucial to have efficient implementations especially for the iterative algorithms such as gradient descent and those mentioned earlier. FlexPS created by Haung et al. (2018)[51] was designed to be able to handle flexible parallelism control. Usually a parallelism degree (number of workers) is set and the choice of parallelism degree is important to set a value which balances computation time and communication overhead. A larger parallelism degree means more workers (reduced computation time) but it also means increased communication time (there are more workers pushing/pulling the updated values to/from the servers). The previously mentioned PS systems set a constant parallelism degree. To undertake flexible parallelism control, FlexPS utilizes multi-stage abstraction[51], where a machine learning task is broken into smaller stages with each stage having a distinct parallelism degree. FlexPS can be considered a complete general PS system as it supports static workloads, so processes which don't require flexible parallelism can also be undertaken in FlexPS. FlexPS can handle all three types of fault tolerance, BSP, SSP and ASP, as it's customizable key value datastore can support all three.

5.2.5 Deep Learning Systems

Underpinning deep learning systems such as TensorFlow and MXNet is the parameter server architecture. However, deep learning systems are concerned with providing the user with a high-level abstraction to build a dataflow graph, enabling users to focus on improving the graph optimization. Whereas the main concern of PS systems is low level communication between the workers and servers. We will briefly touch on the workings of TensorFlow and MXNet as they are widely used in research and industry and are based on PS architectures and do use PS architecture for their module management.

TensorFlow [52] is a dataflow system which uses the parameter server system architecture. TensorFlow allows mutable state nodes which represent different computations, with values moving through the graph being known as 'tensors'. Operations can be applied on these tensors which return a resulting tensor. TensorFlow has a client in addition to the traditional worker and master models, with the client being responsible for creating a session. In a session the computations are requested to the master, which assigns the tasks to workers. All workers are assigned a computational job by the master, and workers can work in parallel. A benefit of TensorFlow is the rich interface compared to PS as it has client API's in Python and C++.

MxNet [53] is another dataflow system framework which is a collaborative project of CXXNet, Minverva, and Purine2. Like TensorFlow, it also has mutable state nodes in a cyclic computational graph. Model parallelism is easier to implement as compared to TensorFlow, but it also supports data parallelism through multiple CPU and GPU. MXNet also enables users to train models in a synchronous and asynchronous mode. It has a dependency engine which checks for the dependencies while performing computation and this engine handles all dependencies accordingly, the components that are not dependent are executed in parallel. This make its execution speed faster than any other framework mentioned before. A middle layer of graph and memory optimization also maximizes its execution time. It uses a declarative programming language to show the computations however it has client API in languages such as Scala, Python, R and C++.

5.3 Other System Architectures

The parameter server framework is not the only available architecture, as it does suffer from some communication lag with all the workers communicating with one central server. Tang et al (2020)[54] provided an overview of the three architectures,

'All-reduce' which tries to address this bottleneck, where there is no central server and workers communicate with themselves. This approach is used by researchers wishing to reduce communication congestion however it does not suit applications when asynchronous communication is needed, hence difficult to apply the SSP communication protocol when using this model.

Lastly there is the 'gossip' architecture, where neighbour workers communicate to calculate distributed models based on chosen edges, not communicating with all workers like in 'all-reduce'. There is no consistency during the training, however once complete the model is globally consistent. The main difference to the PS architecture is the lack of global model and convergence is difficult to achieve.

When comparing these three architectures, if computation time is important but can use BSP, all-reduce is suitable, however if consistency and asynchronous communication is important PS is most suitable. Gossip architectures are beneficial for situations requiring low communication congestion. The other benefit of PS is the amount of systems which have been developed which use the architecture.

5.4 Critical Analysis

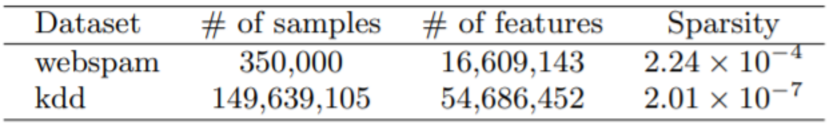
System	Shared Data	Consistency	Fault Tolerance	Flexible Parallelism
Parameter Server	Sparse matrix	Various	Continuous	No
PLMS	Hash table	SSP	Checkpoint	No
Multiverso	Table	BSP	RDD	No
FlexPS	Static vector	Various	Checkpoint	Yes
TensorFlow	Tensors	BSP	Checkpoint	No
MXNet	Graph	BSP	Checkpoint	No
GraphLab	Graph	Eventual	Checkpoint	No
MLBase	Table	BSP	RDD	No

Table 5.1: Comparison of PS and Distributed ML Systems and Features

As seen in Table 5.1, there are differences in the systems in how the data is represented, the consistency protocol used and how fault tolerance is handled, all important factors to consider when choosing a PS or ML system to use. The systems italicised are general ML systems and those in bold use the parameter server framework.

5.4.1 Flexible parallelism

Looking at the impact flexible parallelism has on the running time, an experiment was completed by the creators of FlexPS, Haung et al (2018)[51] (the data and figures are from that paper) comparing FlexPS with Multiverso and PMLS(Bösen). Comparing the computation time for a SVM with the SGD algorithm, a few variations of FlexPS were included, FlexPS- denotes no flexible parallelism, FlexPS-Opt has the optimum parallelism degree calculated offline, and FlexPS-Auto had parallelism degree calculated at runtime.



Dataset	# of samples	# of features	Sparsity
webspam	350,000	16,609,143	2.24×10^{-4}
kdd	149,639,105	54,686,452	2.01×10^{-7}

Figure 5.2: Datasets used in experiment. Source [51]

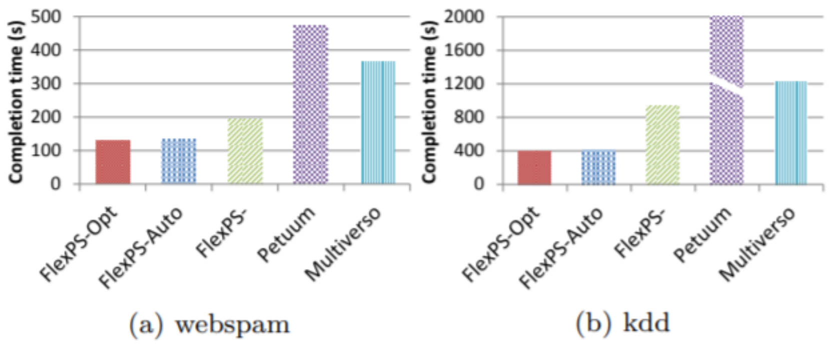


FIGURE 5.3: Completion time of SVM with SGD. Source [51]

	FlexPS-Opt	FlexPS-Auto	FlexPS-	Petuum	Multiverso
webspam	3877	3449	33376	94540	14676
kdd	31852	31275	70770	-	36540

FIGURE 5.4: Total worker time (sec) of SVM with SGD. Source [51]

A summary of the data used is provided in Table 5.2. Seen in Figure 5.3 PMLS performs the worst on this task, the FlexPS systems outperform the other two, with a reduction of 47% in computation time for the webspam dataset. The impact flexible parallelism has is seen in the improved performance in both datasets. Another measure for performance, is the total worker time (the sum of the working time of all the worker threads), a lower worker time is better, the worker time for this experiment is seen in Figure 5.4. Multiverso has less worker time than FlexPS-, however the other FlexPS systems have the lowest worker time. These results show that flexible parallelism does have an impact on performance and is an important factor to consider when considering systems, if compute time and worker time is important, implementing the system in FlexPS would be a good option.

5.4.2 Deep Learning Systems

Interested in seeing how deep learning systems performed compared to PS centric approaches and the architectural design for distributed machine learning algorithms on different platforms. It studies the effect on scalability, performance and usability of these platforms. The experimental results are from Zhang et al. (2017) [55].

Each framework is tested on two basic machine learning tasks: regression and image classification using neural networks. For regression a dataset of 10,000 samples was used with a feature set of 10000. For MXNet and Tensorflow synchronous stochastic gradient descent training was used with different datasets. Whereas for Spark, using its capability of batch processing, batch gradient descent is used for training the model. For PMLS, stochastic gradient descent algorithm is used. Each framework cluster is provided with three worker nodes and one master node, which will act as a driver node. For the image classification MXNet, TensorFlow and Spark were used with different models on MNIST dataset.

5.4.2.1 Results

It is found that MXNet is slightly faster than TensorFlow in the experiment setup except for the asynchronous training on single layer NN and two layers NN[55]. The computation speed of Spark decreases more significantly compared to the other two systems as the model size increases. According to experiment results of training a single NN and two-layer NN model, MXNet works faster than TensorFlow whereas the speed of Spark is the slowest. As the model size increases Spark's performance decreases, because it uses CPU more than other two platforms. Although the network utilization of Spark is less than the other two, it's mainly due to its batch processing feature. Due to this batch processing the memory utilization is very high as compared to MXNet and TensorFlow. The authors noticed that for complex machine learning tasks, basic data flow systems such as Spark fail to scale better. For more advanced tasks such as training deep neural networks, the parameter-server model is better because its supports mutable state and fast iterations. This is the reason it has been adopted by the machine learning community and more advanced systems such as TensorFlow and MXNet are built on top of it.

5.4.3 Critical Analysis Summary

From these experiments, the impact a parameter server approach has on efficiency is demonstrated, through the run time improvements over Spark, and how some improvements such as including flexible parallelism can further improve the systems. Comparison of systems is seen in Table 5.1, with key features highlighted, when considering which system to use, FlexPS would be suitable when working with data requiring a flexible number of working nodes, PMLS is useful when need a general purpose framework with a variety of modules. Parameter server based systems are widely used for distributed ML and there are opportunities for further development of systems at the frontier of distributed ML innovations, translating research into usable technologies.

Chapter 6

MLaaS

6.1 Machine learning as a service

Machine Learning will help shape the future world, therefore it will also produce a new set of challenges. Challenges that need to be addressed constantly whether these are technical ones, or the ones created by lack of infrastructure, data and computing power or data privacy. Machine learning models need quality data to produce good results however its always not easy to gather huge amount of data on your own. Even if you have your data the problem does not end here. You need a good infrastructure to have the required computing power to constantly processed and analyzed data into valuable information. This process requires a lot of resources. One solution to all these problems is Machine learning as a service (MLaaS). MLaaS provides range of cloud-based systems that provide solution to the problem of data pre-processing, training a model, evaluation, with further predictions. The emergence of cloud providers makes ML accessible to those who do not have access to expensive private computing clusters.

6.2 MLaaS: Cloud Providers

All the major competitors in the IT industry such as Microsoft Azure, Google, Amazon Web Services, and IBM Watson have launched their Machine learning services. These platforms are the latest trend of distributed machine learning which enabled users to solve complex problems. A brief overview about each of the technology is given below.

6.2.1 Amazon Machine Learning

Amazon Machine Learning [56] provides many automated solutions which can help user in deadline-sensitive operations. Data can be uploaded into their cloud using various techniques and all pre-processing operations are performed on data automatically. It also selects machine learning method automatically by looking the type of data uploaded in the system. It provides APIs to perform Predictive analytics on data in real-time or on-demand. The predictive analysis automated solutions are for people with basic knowledge of machine learning, because of this it provides less opportunities to make custom solution. The SageMaker tool [57] for is also a part of Amazon Machine Learning services. It helps you in creating your own solutions from the ground up, without server management hassle and it optimizes large dataset to work in distributed environment. It also enables user to build models using various machine learning libraries such as Keras, TensorFlow, Gluon, Torch and MXNet.

6.2.2 Microsoft Machine Learning

Microsoft has also introduced its machine learning services[58]. The products from Microsoft are flexible, it provides freedom to its user to create customize solutions and implementing out of the box algorithms. Mainly, we can have a look at the services mainly APIs and Azure Machine Learning Studio which is a powerful tool for both beginner and experts. Azure ML Studio is a graphical drag-and-drop tool which allows users to explore data, create pre-processing process, choose from a wide variety of algorithms to create models and validating modeling results. The graphical interface is modeled as a workflow system which has a learning curve but eventually benefits are more than you expect. It supports close to 100 methods that address classification, regression, anomaly detection, text analysis and recommendation. It also provides one clustering algorithm (K-means). It has a Cortana Intelligence Gallery where community members showcase their solutions. ML Azure services platform is for proficient data scientists, it provides support environment to build models, experiment with them using various popular frameworks like TensorFlow, scikit-learn and third-party tools such as Docker. It also contains extensive libraries which support different task such as computer vision, forecasting, text analysis, and hardware acceleration. Experimentation allows engineer to build model with specific configurations, compare them and deploy as required. You can also work on your previous models for improvements and modifications. The models are managed by Model management system which allows engineers to host, manage, version and monitor models that run on Azure. Workbench provides command-line environment to monitor the model development and lastly, Visual Studio Tools for AI which helps engineer to work with deep learning models. All in all, Microsoft services are a complete package with more to come in future. It is worth mentioning here that recently Microsoft trained world's largest Transformer language model with 17 billion parameters and open-sourced DeepSpeed - a deep learning library to make distributed training of large models easier[59].

6.2.3 Google Cloud Machine Learning

Like Microsoft and Amazon, Google also provides two main machine learning platforms, one for its inexperienced users and the other for the expert data scientists. Google Cloud AutoML [60]is cloud based platform for beginner, where user can upload data to cloud to train different models, these models then can be deployed using REST APIs. These services include image, video and NLP processing services. Google Cloud Machine Learning Engine is another platform suited for experienced data scientists; this is flexible in a sense that users can create custom models using cloud infrastructure. A variety of machine learning tools and libraries are available, TensorFlow is also a product of Google [61], which is powerful to use for deep neural network tasks.

6.2.4 IBM Watson Machine Learning Studio

Unlike other platforms mentioned before, IBM [62] has a single machine learning platform for beginners and experts. This single system provides two different type of ML methods, automatic and manual. Its Auto AI is fully automated from data processing, model training to model deployment. It can handle binary, multiclass classification and regression tasks. IBM SSPS software package is also a part of this system which allow user to transform data into statistical business information and train models with it. Neural Network Modeler is the same as SSPS but it is used for neural network models and text processing.

6.2.5 Summary

We can see that these powerful technologies are taking distributed machine learning to whole new different level, with accessible, immense computing power and quality datasets.

Through these you can solve complex real time problems, which makes machine learning more applicable. Azure has the most versatile tools in MLaaS, which covers many ML-related tasks. However, it this would not be wrong if we consider cloud-based machine learning is the future of ML with the growth in the fields like AI and IOT, it will only get more popular among the community.

6.3 Privacy concerns

With the growing popularity and leveraging the advantages of remote services provided by MLaaS architecture, an important aspect of the architecture is around data security. This involved security for both the parties - the consumer or user and the service provider. The user worries about not knowing what is happening with his or her private data circulating when his/her credit card information or the medical report is uploaded to one of the service provider website. Similarly, the service provider has trained a model on these distributed ML system and considered the parameters as an intellectual property of the company and would want to protect them from leaking.

Researchers are actively targeting these problems and tackling them broadly in two ways: data centric and model centric approaches. In this survey we will discuss each approach using the most recent research.

6.3.1 General setting

Both data centric and model centric approaches consider a general setting for the data. There are three stakeholders in the process: the user who shares the data, the service provider who consumes the data and some sort of honest-but-curious regulator who does not collaborate with the service provider and helps in computing the data without accessing the actual input information.

6.3.2 Data centric approach

Traditionally MLaaS technologies involved centralised data collection and aggregation. Shasabadi et al. (2020) [63] proposes in their work a way to allow users to privately train their data and privacy to users who use the model for predictions. The algorithms PriEdge distinguishes the images into private and non-private images. Private images are those which can reveal the information about the user such as a face image or handwriting. Preserving the privacy of users means that no other user can upload private image of a different user. The algorithm protects the images till the time they are not classified as non-private images.

The service provider using training data of N-users, aims to build an N-class classifier to predict if a new image belongs to any of these N classes. The algorithms thus allows each user to privately train a distinct class without using a public pre-trained feature extractor. A one class Reconstructive Adversarial Networks (RANs) is created by each user. RAN is based on GAN [64] but with a few differences: 1) the input into the RAN is a set of images rather than noise; 2) the aim of the reconstructor is to create the output as similar as possible to the input and 3) the negative log-likelihood is replaced by mean square loss function. This ensures that the algorithm is able to reconstruct high quality similar images.

Because each user trains in private, most complexity lies during prediction when all of the user data are combined for prediction. Private reconstruction of each class classifier is performed and then dissimilarity based prediction is performed. The service provider and the regulator together achieves substantial gain in complexity by performing the private prediction of each 1-class classifier using different protocols for different operations. Representing non-linear transformations as simple Boolean circuits help the algorithm to compute using Yao's Garbled Circuit [65] and linear transformations such as multiplication have an efficient and secure arithmetic circuit representation.

PrivEdge efficiently distinguishes handwritten texts of a user with overall precision of 91% and recall of 92%. However, its performance worse than non-private state-of-the-art algorithms by nearly 4%.

6.3.3 Model centric approach

Another way of ensuring privacy of data is by encrypting the data while transferring it to a model in a distributed setup. Xue (2020) [66] proposes a way of privacy preservation in decision tree algorithms in MLaaS. Decision tree algorithms are widely used in machine learning classification because of their ease of interpretation as well as power. In a MLasS paradigm, a service provider using a pre-trained decision tree can input a client detail as a feature vector and obtain a prediction for that client. Similar to the above case: two security concerns arises in such scenarios - 1) The privacy of the client personal data; 2) The security of the decision tree classification parameters.

At a high level, both service and client obtain their public-private key and register with the trusted authority (TA). The client using variant of Paillier's encryption converts feature vectors into ciphertext. Based on the novel comparison protocol, the client and service provider evaluate the decision tree and secretly share the comparison results at each decision node. After evaluation, the service provider returns the encrypted classification result to the client, who can recover it using its private key.

6.3.3.1 Performance evaluation

The time cost for a high dimensional input vector and large depth tree model was nearly 2x when the bandwidth used was 0.08 MB. The algorithm is highly efficient for sparse trees as well because the service provider has to convert them to complete binary tree. When number of decision nodes increases from 10 to 100, the computation time and bandwidth required for the client and the service provider increases linearly as well. Additionally, when the dimension of feature vectors is fixed, the communication overhead grows linearly with the number of decision nodes.

Chapter 7

Conclusion

In this survey we had an in depth look at the research in large scale data analytics and distributed machine learning. The field has evolved from MapReduce, and has a lot more growth ahead, as distributed machine learning provides an answer to scalability problems associated with large scale data.

We have seen how large scale data analytics is used in IoT. Progression in the analytics field could directly affect the field of IoT. The issues with IoT using cloud and fog computing are surely challenging but if researches break through them, the results would be game-changing for the betterment of the world.

Parameter server frameworks provide an efficient, scalable, easy to implement approach to distributed machine learning, with new systems and improvements constantly being developed. The solutions are becoming more usable for the end users, which means distributed machine learning is more accessible than ever. PS underpins systems such as TensorFlow and has proved to be a reliable framework, despite some problems such as communication bottlenecks, there are solutions to address these problems.

Lastly, we looked at ML as a service and the present solutions available in the market. These solutions are not only easy to use but also reduces the time to release a product in the market. The distributed ML is an evolving field and it will be interesting to see what turns it take in the future.

7.1 Future Works

According to experts around 44 zettabytes of data is produced daily by the users all over the world and it will increase every year [67]. It will be a challenge to store, maintain and process this amount of data. These data intensive problems were finally solved by data clouds and open source technologies such as Hadoop which is enabling us to process all this data. Most of the famous cloud services such as Amazon, Azure and Google cloud is giving pay as you go service which makes these services more accessible for the people, it also resolved problems like scalability and lack of infrastructure. These data cloud technologies are now evolving into hybrid and multi-cloud environments to cater issue like privacy and information theft. The current systems are more about capacity, read-write efficiency, and scalability. The redundant blocks of data improve fault tolerance in case of system failures and it also increases concurrent performance. Streaming models have been created and are constantly improving to support application which needs data continuously. Elastic Search is another technology, which improved online interactive query systems. In the future we will see rise in fast data or actionable data. This type of data can be processed in real time. These streams of data can be processed very fast and it can create results within milliseconds, this is more beneficial for organizations as it will improve access, customer experience and decision making in businesses. Analyzing distributed machine learning technologies and their challenges, we realize the future of distributed machine learning needs to address the challenges of privacy of user data. As future work, we wish to keep a close eye on federated learning [68] which is a privacy preserving learning algorithm. It is pushing boundaries of distributed machine learning by observing mobile phone user behavior, adaptive pedestrian behavior in autonomous vehicles and using wearable device sensor information to predict health events. With the introduction of Nvidia's GPU and Google's TPU, more deep learning workloads will be handled solving more complex problems in the future.

Bibliography

[1] Seref Sagiroglu and Duygu Sinanc. Big data: A review. In *2013 international conference on collaboration technologies and systems (CTS)*, pages 42–47. IEEE, 2013.

[2] Ying Liu, Anthony Soroka, Liangxiu Han, Jin Jian, and Min Tang. Cloud-based big data analytics for customer insight-driven design innovation in smes. *International Journal of Information Management*, 51:102034, 2020.

[3] Patrick Mikalef, John Krogstie, Ilias O Pappas, and Paul Pavlou. Exploring the relationship between big data analytics capability and competitive performance: The mediating roles of dynamic and operational capabilities. *Information & Management*, 57(2):103169, 2020.

[4] samuel. URL https://en.wikipedia.org/wiki/Arthur_Samuel.

[5] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. volume 51, pages 137–150, 01 2004. doi: 10.1145/1327452.1327492.

[6] T White. *Hadoop: The Definitive Guide*. 01 2010.

[7] Vinod Vavilapalli, Arun Murthy, Chris Douglas, Sharad Agarwal, Mahadev Konar, Robert Evans, Thomas Graves, Jason Lowe, Hitesh Shah, Siddharth Seth, Bikas Saha, Carlo Curino, Owen O'Malley, Sanjay Radia, Benjamin Reed, and Eric Baldeschwieler. Apache hadoop yarn: yet another resource negotiator. 10 2013. doi: 10.1145/2523616.2523633.

[8] Matei Zaharia, Reynold Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael Franklin, Ali Ghodsi, Joseph Gonzalez, Scott Shenker, and Ion Stoica. Apache spark: A unified engine for big data processing. *Communications of the ACM*, 59:56–65, 11 2016. doi: 10.1145/2934664.

[9] Illia Polosukhin. Data vs model parallelism, 2017. URL https://www.slideshare.net/ilblackdragon/optimizing-distributed-tensorflow.

[10] Hao Li, Asim Kadav, Erik Kruus, and Cristian Ungureanu. Malt: Distributed data-parallelism for existing ml applications. In *Proceedings of the Tenth European Conference on Computer Systems*, EuroSys '15, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450332385. doi: 10.1145/2741948.2741965. URL https://doi.org/10.1145/2741948.2741965.

[11] Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks. *CoRR*, abs/1404.5997, 2014. URL http://arxiv.org/abs/1404.5997.

[12] Zhihao Jia, Matei Zaharia, and Alex Aiken. Beyond data and model parallelism for deep neural networks. *CoRR*, abs/1807.05358, 2018. URL http://arxiv.org/abs/1807.05358.

[13] Tim Kraska, Ameet Talwalkar, John C Duchi, Rean Griffith, Michael J Franklin, and Michael I Jordan. Mlbase: A distributed machine-learning system.

[14] Evan Sparks, Ameet Talwalkar, Virginia Smith, Jey Kottalam, Xinghao Pan, Joseph Gonzalez, Michael Franklin, Michael Jordan, and Tim Kraska. Mli: An api for distributed machine learning. 10 2013. doi: 10.1109/ICDM.2013.158.

[15] Yucheng Low, Joseph E. Gonzalez, Aapo Kyrola, Danny Bickson, and Carlos Guestrin. Graphlab: A distributed framework for machine learning in the cloud. *ArXiv*, abs/1107.0922, 2011.

[16] Yucheng Low, Joseph Gonzalez, Aapo Kyrola, Danny Bickson, Carlos Guestrin, and Joseph M Hellerstein. Distributed graphlab: A framework for machine learning in the cloud. *arXiv preprint arXiv:1204.6078*, 2012.

[17] Hongliang Guo and Jie Zhang. A distributed and scalable machine learning approach for big data. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, page 1512–1518. AAAI Press, 2016. ISBN 9781577357704.

[18] C.J.C. Burges and Chris J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, January 1998. URL https://www.microsoft.com/en-us/research/publication/a-tutorial-on-support-vector-machines-for-pattern-recognition/.

[19] Su-In Lee, Honglak Lee, Pieter Abbeel, and Andrew Ng. Efficient l1 regularized logistic regression. volume 21, 01 2006.

[20] Martin Zinkevich, Markus Weimer, Lihong Li, and Alex J. Smola. Parallelized stochastic gradient descent. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 2595–2603. Curran Associates, Inc., 2010. URL http://papers.nips.cc/paper/4006-parallelized-stochastic-gradient-descent.pdf.

[21] David Sontag, Amir Globerson, and Tommi Jaakkola. Introduction to dual decomposition for inference. January 2010. URL https://www.microsoft.com/en-us/research/publication/introduction-to-dual-decomposition-for-inference/.

[22] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2010. ISSN 1935-8237. doi: 10.1561/2200000016.

[23] Gonzalo Mateos, Juan Bazerque, and G.B. Giannakis. Distributed sparse linear regression. *Signal Processing, IEEE Transactions on*, 58:5262 – 5276, 11 2010. doi: 10.1109/TSP.2010.2055862.

[24] Joseph K. Bradley, Aapo Kyrola, Danny Bickson, and Carlos Guestrin. Parallel coordinate descent for l1-regularized loss minimization. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 321–328, Madison, WI, USA, 2011. Omnipress. ISBN 9781450306195.

[25] Paul Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, 109:475–494, 2001.

[26] Xianda Zhang. Matrix analysis and applications. 01 2004. doi: 10.1017/9781108277587.003.

[27] Justin Ma, Lawrence Saul, Stefan Savage, and Geoffrey Voelker. Identifying suspicious urls: An application of large-scale online learning. page 86, 01 2009. doi: 10.1145/1553374.1553462.

[28] J. Torres-Sospedra, R. Montoliu, A. Martínez-Usó, J. P. Avariento, T. J. Arnau, M. Benedito-Bordonau, and J. Huerta. Ujiindoorloc: A new multi-building and multi-floor database for wlan fingerprint-based indoor localization problems. In *2014*

International Conference on Indoor Positioning and Indoor Navigation (IPIN), pages 261–270, 2014.

[29] Franz Graf, Hans-Peter Kriegel, Matthias Schubert, Sebastian Pölsterl, and Alexander Cavallaro. 2d image registration in ct images using radial image descriptors. volume 14, pages 607–14, 09 2011. doi: 10.1007/978-3-642-23629-7_74.

[30] Krisztian Buza. *Feedback Prediction for Blogs*, pages 145–152. 10 2014. doi: 10.1007/978-3-319-01595-8_16.

[31] Pedro A. Forero, Alfonso Cano, and Georgios B. Giannakis. Consensus-based distributed support vector machines. *J. Mach. Learn. Res.*, 11:1663–1707, August 2010. ISSN 1532-4435.

[32] Aaron Defazio, Justin Domke, and Caetano. Finito: A faster, permutable incremental gradient method for big data problems. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1125–1133, Bejing, China, 22–24 Jun 2014. PMLR. URL http://proceedings.mlr.press/v32/defazio14.html.

[33] Konstantinos Gatsis and George J Pappas. Wireless control for the iot: Power, spectrum, and security challenges. In *2017 IEEE/ACM Second International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pages 341–342. IEEE, 2017.

[34] Sachin Kumar, Prayag Tiwari, and Mikhail Zymbler. Internet of things is a revolutionary approach for future technology enhancement: a review. *Journal of Big Data*, 6(1):111, 2019.

[35] Redowan Mahmud, Ramamohanarao Kotagiri, and Rajkumar Buyya. Fog computing: A taxonomy, survey and future directions. In *Internet of everything*, pages 103–130. Springer, 2018.

[36] Luis M Vaquero and Luis Rodero-Merino. Finding your way in the fog: Towards a comprehensive definition of fog computing. *ACM SIGCOMM Computer Communication Review*, 44(5):27–32, 2014.

[37] Shanhe Yi, Zhengrui Qin, and Qun Li. Security and privacy issues of fog computing: A survey. In *International conference on wireless algorithms, systems, and applications*, pages 685–695. Springer, 2015.

[38] Dimitrios Zissis and Dimitrios Lekkas. Addressing cloud computing security issues. *Future Generation computer systems*, 28(3):583–592, 2012.

[39] Clinton Dsouza, Gail-Joon Ahn, and Marthony Taguinod. Policy-driven security management for fog computing: Preliminary framework and a case study. In *Proceedings of the 2014 IEEE 15th International Conference on Information Reuse and Integration (IEEE IRI 2014)*, pages 16–23. IEEE, 2014.

[40] Feng Xia, Laurence T Yang, Lizhe Wang, and Alexey Vinel. Internet of things. *International journal of communication systems*, 25(9):1101, 2012.

[41] Arwa Alrawais, Abdulrahman Alhothaily, Chunqiang Hu, and Xiuzhen Cheng. Fog computing for the internet of things: Security and privacy issues. *IEEE Internet Computing*, 21(2):34–42, 2017.

[42] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16, 2012.

[43] Mohamed Labbi, Nabil Kannouf, and Mohammed Benabdellah. Iot security based on content-centric networking architecture. In *Security and Privacy in Smart Sensor Networks*, pages 179–199. IGI Global, 2018.

[44] M. Li, D.G. Andersen, A. Smola, and K. Yu. Communication efficient distributed machine learning with the parameter server. *Advances in Neural Information Processing Systems*, 1:19–27, 01 2014.

[45] Arda Aytekin, Hamid Feyzmahdavian, and Mikael Johansson. Analysis and implementation of an asynchronous optimization algorithm for the parameter server. 10 2016.

[46] Leslie G. Valiant. A bridging model for parallel computation. *Commun. ACM*, 33(8):103–111, August 1990. ISSN 0001-0782. doi: 10.1145/79173.79181. URL `https://doi.org/10.1145/79173.79181`.

[47] Qirong Ho, James Cipar, Henggang Cui, Jin Kyu Kim, Seunghak Lee, Phillip B Gibbons, Garth A Gibson, Gregory R Ganger, and Eric P Xing. More effective distributed ml via a stale synchronous parallel parameter server. *Advances in neural information processing systems*, 2013:1223—1231, 2013. ISSN 1049-5258. URL `https://europepmc.org/articles/PMC4230489`.

[48] Mu Li, David G. Andersen, Jun Woo Park, Alexander J. Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J. Shekita, and Bor-Yiing Su. Scaling distributed machine learning with the parameter server. In *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, pages 583–598, Broomfield, CO, October 2014. USENIX Association. ISBN 978-1-931971-16-4. URL https://www.usenix.org/conference/osdi14/technical-sessions/presentation/li_mu.

[49] E. P. Xing, Q. Ho, W. Dai, J. K. Kim, J. Wei, S. Lee, X. Zheng, P. Xie, A. Kumar, and Y. Yu. Petuum: A new platform for distributed machine learning on big data. *IEEE Transactions on Big Data*, 1(2):49–67, 2015.

[50] Microsoft. Multiverso. URL https://github.com/Microsoft/multiverso/wiki,year=2015.

[51] Yuzhen Huang, Tatiana Jin, Yidi Wu, Zhenkun Cai, Xiao Yan, Fan Yang, Jinfeng Li, Yuying Guo, and James Cheng. Flexps: Flexible parallelism control in parameter server architecture. *Proc. VLDB Endow.*, 11(5):566–579, January 2018. ISSN 2150-8097. doi: 10.1145/3177732.3177734. URL https://doi.org/10.1145/3177732.3177734.

[52] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, page 265–283, USA, 2016. USENIX Association. ISBN 9781931971331.

[53] Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems, 2015. URL http://arxiv.org/abs/1512.01274. cite arxiv:1512.01274Comment: In Neural Information Processing Systems, Workshop on Machine Learning Systems, 2016.

[54] Zhenheng Tang, Shaohuai Shi, Xiaowen Chu, Wei Wang, and Bo Li. Communication-efficient distributed deep learning: A comprehensive survey. *ArXiv*, abs/2003.06307, 2020.

[55] Kuo Zhang, Salem Alqahtani, and Murat Demirbas. A comparison of distributed machine learning platforms. pages 1–9, 07 2017. doi: 10.1109/ICCCN.2017.8038464.

[56] Amazon. URL https://docs.aws.amazon.com/machine-learning/latest/dg/what-is-amazon-machine-learning.html.

[57] sagemaker. URL https://aws.amazon.com/sagemaker/?nc2=h_a1.

[58] azureml, . URL https://azure.microsoft.com/en-us/services/machine-learning.

[59] azureai, . URL https://azure.microsoft.com/en-us/overview/ai-platform/.

[60] googleml, . URL https://cloud.google.com/automl/.

[61] googleai, . URL https://cloud.google.com/ai-platform/docs/technical-overview.

[62] ibmwat. URL https://dataplatform.cloud.ibm.com/docs/content/wsj/analyze-data/ml-pipelines.html.

[63] Ali Shahin Shamsabadi, Adrià Gascón, Hamed Haddadi, and Andrea Cavallaro. Privedge: From local to distributed private training and prediction. *IEEE Transactions on Information Forensics and Security*, 2020.

[64] Gan. URL https://en.wikipedia.org/wiki/Generative_adversarial_network.

[65] Yao. URL https://en.wikipedia.org/wiki/Garbled_circuit.

[66] Liang Xue, Dongxiao Liu, Cheng Huang, Xiaodong Lin, and Xuemin Sherman Shen. Secure and privacy-preserving decision tree classification with lower complexity. *Journal of Communications and Information Networks*, 5(1):16–25, 2020.

[67] Fw. URL https://www.microsoft.com/en-us/research/lab/microsoft-research-asia/articles/the-present-and-future-of-data-intelligence/.

[68] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *arXiv preprint arXiv:1908.07873*, 2019.