BIG DATA SYSTEMS FOR ARTIFICIAL INTELLIGENCE

A PREPRINT

Vibhatha Abeykoon*
Luddy School of Informatics,
Computing and Engineering
Indiana University Bloomington
United States
vlabeyko@iu.edu

December 15, 2019

ABSTRACT

Artificial intelligence (AI) is an inevitable reality, evolved on modern-day scientific breakthroughs. AI itself is a collection of knowledge and experience. In the current research, the closest approximation towards AI is made by developing applications with deep learning and reinforcement learning. In designing AI-enabled solutions another inevitable component is the system design. Intelligence wrapped around knowledge requires a learning curve on larger datasets. Enabling intelligent solutions with a larger amount of data raises some concerns related to streamlining dataflow for specific AI algorithms. Dealing with such larger datasets, the state of the art solution is to use big data systems. Separating a big data processing framework with AI-enabled systems is hard thing to do. Due to this inability to decouple AI workflow with big data systems, the challenge is to identify how big data systems can co-exist with AI-enabled systems. This is an emerging topic in recent research. Another important aspect is to see how to keep existing big data systems keeping up with the growing AI-enabled systems. Big data systems play a major role in data organization aspect. Besides, in unifying data organization, learning and evaluation process, an AI-enabled system must be linked with a streamlined workflow. Another important thing is supporting AI-enabled systems with the state of the art big data systems. With emerging data collection, improving training time is a vital task. Distributed training, data organization and AI algorithm development with rich application interfaces enables a streamlined workflow for scientists. Once an AI model is optimized to do the expected task, the next important task is to enable low-latency inference. Combining big-data systems with an inference workflow is vital. In terms of improving systems considering the hardware stack, both training and inference of AI algorithms must take place in the state of the art efficient hardware. In facilitating this with big data systems, hardware capabilities must be taken into consideration when designing big data systems or when designing interfaces for existing big data systems. In this study, we do a deep analysis of understanding the current status of AI-enabled systems and the future of the AI-enabled ecosystem. The core of our study is based on how academic research influenced early day AI. Specially, we take a close look at how artificial intelligence is efficiently designed by private co-operations like Google, Facebook, Uber, Tesla, Microsoft and Amazon. From the observed conclusions, we suggest a workflow for AI systems on top of big data systems.

***K*eywords** Stream Processing · Batch Processing · Deep Learning · Reinforcement Learning · Edge Inference

1 Introduction

Instead of using a classical constrained-based system, modern-day science has taken a step towards AI-enabled algorithms to solve problems. Deep learning and reinforcement learning are the two pillars dominating the current research. Deep learning systems are highly coupled with larger datasets with higher dimensions. Reinforcement

*https://www.vibhatha.org

learning systems are also coupled with larger datasets, but they are mostly involved with experience-based agent training. Running more simulations and trying to create an older version or more experienced version of the agent from its younger or less experienced version is the end goal. Memory optimizations and computation optimizations are high valued assets to enable an efficient AI system. The most challenging problem is to design systems to enable efficiency depending on the nature of AI algorithms. Depending on the nature of the computation intensity of larger datasets, modern-day hardware has been evolved from CPU to GPU[1] at a particular stage of the evolution of AI systems. Now, this trend has been again evolved from GPU to TPU [2]. Depending on each hardware device-specific AI platforms are being designed by most of the commercial entities like Google[3] and NVidia [4]. Depending on the hardware capabilities, the design specific platforms are also evolving. Tensorflow [5] from Google, Pytorch [6] from Facebook-AI [7], MXNet [8] from Amazon (AWS AI) [9] and CuDNN [10] from NVidia AI [4] are a couple of highly used deep learning and deep reinforcement enabled systems. These AI systems have been designed for training and inference purposes. All these frameworks can be currently classified as general-purpose AI algorithm development platforms. The inspiration behind most of the prominent AI platforms designed by industrial entities is with the inevitable necessity of understanding the data for the increasing demand of a larger number of services in satisfying various aspects of human life. Even though most of these AI platforms are designed by the industrial entities, the strong driving force in enabling machine intelligence comes from academia. Geoffrey Hinton is one of the prominent authors in modern-day artificial intelligence. Deep Learning review on Nature [11], fast learning algorithm for deep belief nets [12] and dimensional reduction with deep neural networks [13] are major contributions among the other topics [14],[15], [16].

Uber [17] focuses their attention mostly on fine-tuning the customer experience on taxi and delivery mechanisms. Uber-Eats and Uber for travelling are the main products involved with AI. Autonomous driving is one of the challenging problems that is being solved by Uber. Also, they focus on food delivery as well. In these scenarios providing the estimated time of arrival by considering current traffic flow, weather and other factors are taken into consideration. Tesla on the other hand provides autonomous driving for vehicles by inventing new AI technologies[18]. Safety for the driver and external entities are most prominent aspects considered in improving the user experience. Object detection, obstacle avoiding, keeping track on the lanes and the curbs of the road are vital to achieve these goals. A variety of sensors provide input to a stack of deep learning systems to make decisions. Microsoft is another entity working on improving AI experience. AI in Microsoft focuses on medical applications, agriculture, production automation, etc. Most of the AI-enabled workflow systems are made available with the Azure web services. Most of these services are focused on vision, language and speech[19]. Amazon is another entity focusing on customer experience via AI. Amazon Alexa is a prominent natural conversing AI-bot. It uses a variety of techniques from voice recognition, speech recognition, complex sentence interpretation, etc. Most of these technologies are coupled with edge computing and cloud computing aspects. Providing minimum latency for AI systems are vital when it comes to deploying an advisory system like Alexa.

Deep learning application on solving challenging scientific problems has become a norm. Earthquake prediction, high-energy particle physics, weather prediction, climate modelling, precision medicine[20], deployment of clean fusion energy are some of the most challenging problems. The other aspect is mining data from human interactions to improve existing business entities and generate new business entities. This is one of the most prominent goals from social media related research done in Facebook, Google, Amazon, Alibaba[21], etc. Christoph Angerer looks at the deep learning systems under two main categories [22]. First method is designing a system not with a rules but with data. The improvement over a larger dataset can evolve to a higher accurate and flexible system. The second aspect is where a learning system is vital when we need to create scientific applications where we use AI as a surrogate model for existing solutions. Taking a climate simulation or any physical model consideration, there are a hundreds of thousands of parameters governing the behaviour. These models have high precision model definitions through mathematical models. Instead of an exact solution, an approximate solution can be developed with parameter-tuning using deep learning systems. These approximate models can be designed by modelling a physical model by tuning a larger number of hyper-parameters with a larger amount of training data. The objective is to generate an approximate model with training data and compare its accuracy with the existing understanding with the real-world data. Once the surrogate functions are well trained, a fine-grain physical model can be replaced with a parameterized model designed with the power of artificial intelligence. In the physics cases the physic models can generate training data. The idea is to speed up the experiments depending on these physical models. AI can speed up such simulations and provide efficient systems to design far complex models.

Understanding AI and being a part of the AI evolution comes with the understanding of requirements of evolving world. The ultimate goal of AI focuses mainly on food, water, health, business and education. Dynamic voltage and frequency scaling in NoCs has been recently evaluated with supervised and reinforcement learning [23]. Monitor health of canine [24] (Billion dollar pet industry has to catch up with the wearable in the market (FitBark). Digital advisors for helping making decisions. For instance teaching how to use a particular device. This can be done with a rule based step by step system. But with an intelligent system, tracing the progress of a learner and understanding the specific areas

he or she needs help, can be done with intelligent systems [25]. AI-enabled safety-critical systems is a vital area of research. When it comes to the generative adversarial networks it can generate fake data which is close to the real data. The generation of fake content to compromise a system dependent on it's intrinsic values. The adversarial examples involved in training another network can be affected by a wrongly trained system by gaining access to input data and tampering with it, model parameters modification and poisoning the training data with adversarial information[26].

In designing intelligent systems, the most important thing is designing an entire system including data organization, AI modelling, AI deployment and AI model evaluation with unobserved external factors. With modern electronic devices, one of the most prominent component in day to day life is a digital advisor. Amazon Alexa[27], Google Assistant[28], Siri[29] and Cortana[30] are some of the prominent digital advisors. The growth of digital advisors increased with the growth of technology trends. From pre-programmed chatbots to natural conversational chatbots were a reality with the growth of technology towards intelligent applications. Figure 1 shows the technology growth against time. In the dawn of the 21st century, primitive AI-enabled applications started appearing. But with the evolving machine learning, deep learning and reinforcement learning algorithms, digital advisors tends to get the maximum out of AI.

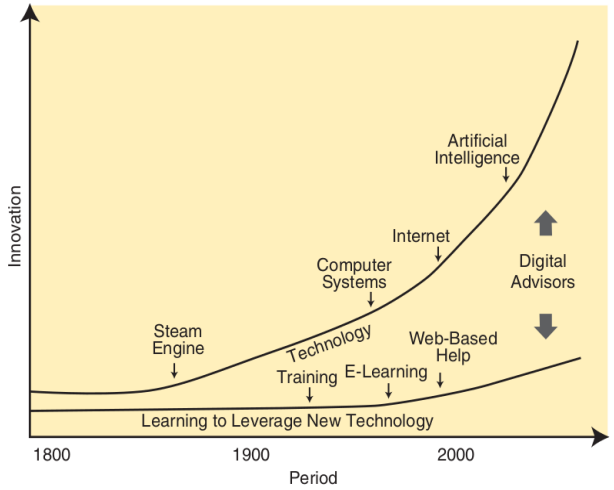


Figure 1: Digital Advisor Technology Growth Rate with Respect to Time. Source [25]

All these intelligent applications are driven by the power of knowledge and experience. For gaining more knowledge, more training has to be done with more data. This implicitly involves the necessity of big data systems. The main focus of this study is to showcase how existing big-data systems support the AI-enabled applications and how big data systems must evolve in meeting emerging requirements of AI-enabled applications. In addition, a study on linking AI platforms with big data systems is also discussed.

2 Workflow for AI Systems

The core of an AI system involves with organized data, algorithm modelling, external system design and AI model deployment[31]. These steps are all sequential components of a complete AI pipeline.

2.1 Data Organization

Data organization is the most vital component in AI systems. Without organized data, no conclusions can be made. Data organization has a couple of important components. The raw-data can take the forms of unlabelled data, schema-less data, sparse data, etc. Processing row-data by labelling, adding schemas and representing sparse data with a memory optimized format can lead to the first step towards data organization.

'Road to AI' article in scientific computing world magazine [32], describes a four stage process where data is being vetted for better use.

- Identification
- Preparation
- Ingestion

- Storage

In terms of data organization steps, *identification* is vital for extracting better features from data sources. The source of data is not always structured, because the data sources are not providing uniform schemes. Some times data arrives from IoT devices, databases or by other data streams like Kafka[33] message broker. In such scenarios, making data into a meaningful format is vital. In this data *preparation* segment, primitive data analysis and pruning are done with algorithms like PCA. Data *ingestion* is all about data gathering and application of necessary data points to the AI or other analytics platforms. In this stage, the data structure, source and size of data must be taken into consideration. Besides, the involvement of streams of data for non-re-playable data sources is vital and adding edge level data pre-processing is very important in formulating schemas. In the final step, the nature of the *storage* of the processed data is vital when it has to be fed to the AI system. Here good data vs cold data selection has to be done. Good data is the data stored in the right storage medium depending on the importance. This refers to a faster dataflow from the storage to the AI system. Cold data is the data stored in a slower storage medium. This can be in the cloud, as the latency is less critical.

. Next important thing is feeding data for the AI system. Dataflow for an AI system must consider the following factors;

- Training dataflow
- Cross-validation dataflow
- Testing dataflow

The training dataflow most of the time deals with retrieving data from existing storage or processing data in-memory.

2.1.1 In-Memory Oriented Dataflow

When the data size for training or predictive task is under the in-memory capacity of the physical resources, in-memory computation strategies can be taken into consideration. When working with different systems, it is always better to keep a common data interface understood by all systems involved (discussed more in 6.4).

2.1.2 Storage Oriented Dataflow

Once the data size reaches beyond the in-memory capacity of the system resources (cluster memory capacity), data must be queried from the storage and fed back to the AI system in terms of a data pipeline. State of the art solutions in the dawn of this decade was to use Hadoop oriented solutions, but this interest has moved towards data structures with columnar format depending on different application requirements (discussed more in 6.4, 4.4).

2.1.3 Touch of HPC

High-performance computing resources and the software stack extending from HPC world has become a vital factor in improving the performance of in-memory dataflow management and distributed training. Deep learning on HPC has been deeply analyzed by many types of research throughout the last decade. Among these work, the large data movement with HPC systems in deep learning is one of the most important problems that have to be tackled [34], [35]. Understanding the correct HPC operator can provide a boost in dataflow for the AI system. In addition, in building a stable system and understanding bottlenecks in scaling AI systems is vital. Awan et.al [36] discusses this problem in HPC environments to understand how various deep learning workloads affect the system. Another important aspect of HPC is to enable existing AI platforms to run efficiently on HPC hardware. Biswas et.al discusses on how Tensorflow can be accelerated using RDMA-based GRPC [37]. The AI platform-driven system design is one of the most important aspects that can accelerate modern-day AI systems.

2.2 AI Modelling

AI modelling is the most challenging task in building AI systems. The modelling stage is mutually exclusive from the external system design as long as the structure of the model is known. When the structure of the model is not known and it has to be designed with the help of simulations. In such a situation, an external system can come to provide such assistance. Such an approach can be taken when there is a rough estimate on the model. When simulating a couple of models with similar core model definition, a workflow has to be designed to select the best configuration. This can be further enhanced by an auto-tuner which can be enabled via a stochastic gradient descent or similar optimization algorithm. The overall loss and the accuracy against the configuration matrices can be recorded and configurations can be auto-tuned upon the feedback from the algorithm. The aforementioned workflow can be abstracted using a dataflow model for algorithm selection.

2.3 System Design

Core system design for an AI system must be highly efficient in training and inference stages. In the evolving software stack which is highly coupled with the hardware stack, designing just an API for building deep learning systems won't be the ultimate solution. The ultimate goal must be to design the software stack along with the improving hardware stack. In recent years, with the dawn of the deep learning and reinforcement learning systems, the hardware stack dominated by CPU based computation moved to GPU based computation. GPUs are better with graphics processing and also very good with matrix related computations. This created a ripple effect in AI system designing. Due to high computational intensity on image processing, language processing, etc. GPU oriented AI system designing became a dominant trend. In recent years, TPU related research has provided another optimization to the software stack. From various researches, it has been shown that TPUs can improve beyond GPUs in AI system modelling at training stage [38], [39]. With such advancements, low latency systems can be designed for training and cross-validation. State of the art AI SDKs like Tensorflow and Pytorch are powered by TPUs and GPUs while CuDNN is powered by GPUs. For an AI system design, another important aspect is to visualize the model and understand feature extraction while training takes place. The data provenance on inference stats like inference time, loss function value, accuracy (top-1, top-5) can be very important to optimize the AI algorithms with the support of the system design wrapped around it.

2.4 Deployment

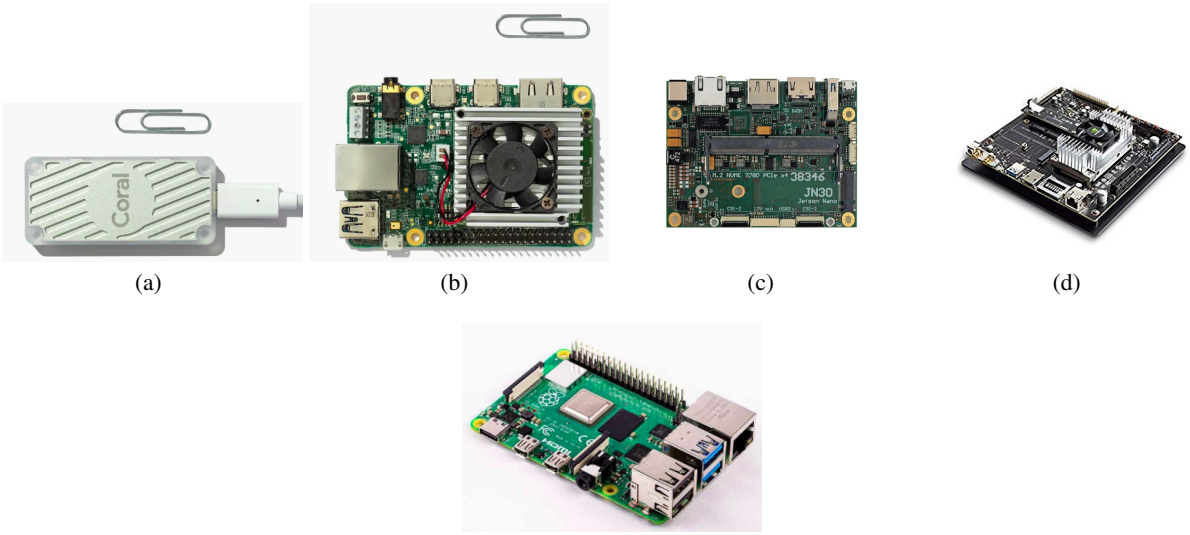
AI system deployment is the end goal of all these steps. Providing AI services with increasing demand is a challenge. The scale-able solution is vital in providing such services. Low-latency is one of the major factors affecting the quality of the services. As same as training, the inference is also very important to provide scale-able solutions. Moreover, the hardware on which both training and inference run is an influential factor. Most of the AI SDKs like Tensorflow, Pytorch and CuDNN have optimized the software stack along with the hardware stack. The reality is training done on servers in a cloud or a static location. But the inference takes place in billions of places in billions of devices. Providing faster hardware and optimizing existing mobile and IoT hardware is vital to improving the efficiency of the deployed AI services. In deep learning, quantization and forming mobile-friendly networks are the key factors towards efficient inference. In addition, network compression and graph optimizations are also done by the tools like TensorRT[40], TensorflowLite[41], Pytorch Mobile[42], etc. With mobile compression, two major actions take place. First one is low-latency inference from INT-8 and FP-16 quantization schemes and smaller model size fitting devices with limited memory[43, 44, 45]. Apart from mobile devices like smartphones, watches and tablets, there are IoT requirements with the increasing remote sensing activities. There is specific hardware produced for meeting the aforementioned requirements. Edge TPU devices[46] (shown in 2 (a) and (b)) and Edge GPU devices[47] (shown in 2 (c) and (d)) have been the solutions in the modern-day research. The software stack has also been optimized to fit for these hardware devices. (For CPU devices Raspberry-PI[48] (shown in 2 (e)) like devices are available to run mobile compatible versions, but hardware acceleration is not there as in Edge TPU or Edge GPU devices.)

3 Role of Big Data Systems

This section is focused on understanding how big data systems can satisfy the requirements in AI systems as described in section 2. System building is a vital factor when it comes to integrating machine intelligence to domain science problems or business problems. With the increase in usage of mobile devices, the data collected also increases in an exponential rate. So, big data processing has become an inevitable component since the last decade. And it will be the most demanding component that powers AI in modern-day science. The quest is to provide a streamline dataflow for these learning systems. Existing big data systems like Apache Spark[49], Apache Hadoop[50], Twister[51] have been major batch processing systems. In-stream processing Apache Storm[52], Apache Flink[53], Heron[54], Alink[55] can be recognized as state of the art streaming systems. Until the mid of this decade, all these big data systems were mainly focused on data processing, querying and storage. With the emergence of AI, all these data processing engines have been adopting certain practices to provide support to such systems. In most of the research conducted in recent years, the main focus has been to provide a streamline data source for AI systems. Big data systems can be influential with AI systems in three main ways. The first one is the training of an AI algorithm. The next phase is to use the trained algorithm at scale. The other important aspect is supporting transient learning. This empowers building AI systems upon AI systems.

3.1 Training System Requirement

This section discusses how big data systems can enable the necessities discussed in sections 2.1, 2.2 and 2.3. Deep learning and reinforcement learning are the major components in modern-day AI. Deep learning highly dependent on high volume and high dimensional data. Reinforcement learning also depends on such larger datasets and also it



(a) (b) (c) (d)

(e)

Figure 2: (a) Edge TPU Accelerator (b) Edge TPU Dev Board (c) Jetson Nano (d) Jetson Tx2 (e) Raspberry PI 4. Note all the figures are not in real scale. Figures in (a) and (b) with scale. Figure (c) includes a device with size 70 mm $\times$ 45 mm. Figure (d) includes a device with size 5 cm $\times$ 9cm. Figure (e) includes a device with size 85.6 mm $\times$ 56.5 mm. Depth of the devices are relatively smaller among all devices.

depends on gaining experience by simulating itself with a set of actions focusing towards a higher reward. All these deep learning and reinforcement learning systems are being powered by software like Tensorflow, Pytorch, MXNet, CuDnn, etc. Creating a streamlined data pipeline to feed these systems is a challenging task. Enabling large in-memory batch processing and scaling out for thousands of machines is vital when training larger systems. Most of these frameworks are wrapped around a C++ core with Python. In a pythonic environment, processing very large datasets with numpy become infeasible, when most of the raw data is not formatted to expected schemas. In addition, memory-bound issues are always coupled with pythonic data structures. Data organization is a vital task in training AI systems. In terms of Big data systems, they are well designed to do this heavy-weight task by distributed computing on batch and stream mode. And also the querying capability with SQL-based interfaces in big data systems enable the data organization much easier. Once the data is well-organized the next challenge is coupling the link between dataflow from a big data system to the AI system. Once this is enabled a complete workflow is therefrom data organization to an AI system. This link can be enabled with distributed file systems like HDFS or modern-day storage formats like Parquet (discussed in 6.4). Another way is to spawn the processes within the big data system by abstracting data pipeline with training pipeline. For very large scale experiments on hundreds and thousands of petabytes of data, this won't be the solution. It rather works well with distributed cache or distributed file systems. But for medium-sized data sets, in-memory dataflow and AI process launching via a big data system can enable a streamlined workflow. Another important aspect is loading cross-validation data which governs the evaluation metric of the AI algorithm. Passing data for cross-validation loop by overlapping I/O and training is another advantage that can be provided for the AI systems. Big data systems can provide a streamline data pipeline to enable this. At the training stage, the most important metric to evaluate the progress of the algorithm can be enabled with data provenance. These capabilities are external attributes to the AI system. But a virtual coupling between these attributes can be enabled through a big data system. The vital nature of a big data system for AI application training monitoring is a must to obtain an efficient AI ecosystem.

3.2 Predictive System Requirement

In this section, we discuss how big data systems can facilitate a streamlined workflow to enable tasks discussed in section 2.4. Once an AI model is trained for the expected accuracy, the next step is deploying the model. For model deployment and enabling low-latency inference, distributed dataflow in streaming mode, lambda or kappa (discussed in 6.3) mode is a vital task. Once you obtain millions of requests from thousands and millions of edge devices, a streamline dataflow model has to be designed to enable low-latency inference. With predictive systems, data provenance is a vital factor for evaluating the model in terms of accuracy and performance. Collecting stats on dataflow operations like gather or reduce must be evaluated to understand the bottlenecks in the predictive systems. Besides with the time, if

a model seems to be under-performing, these stats must also be collected. These evaluation metrics can be encapsulated via a big data system. The main reason for such an ethic lies behind the challenges in stream processing, data storage, distributed cache and data management components. The expert systems doing all these tasks are big data systems.

3.3 Transfer Learning System Requirement

In AI systems, learning a new skill from an existing skill is quite useful in optimizing system design and maintenance. In supporting such tasks, big data systems can enable linking older versions of trained deep learning or reinforcement models with transfer learning. In transfer learning, feature extraction is one of the core requirements. A pre-trained model can be taken and a sub-set of layers can be extracted from it. For instance, excluding a softmax layer, a dimension-reduced representation of a larger image dataset can be effectively obtained. Big data systems can enable a streamlined workflow to enable such a re-training process on top of already trained AI models. The effect of big data systems for this component is merely for data pipelining. Most of the work is done within the mutation of the AI model which is governed by the AI system itself.

4 AI at Work

So far the study was focused on how AI has been progressed throughout the time and where AI has positioned itself at the moment. Understanding these ideas help to improve big data systems and evolve on top of it. In this section, the main focus is to analyze how industrial business entities have collaborated for the evolving AI and how AI has been positioned to solve the problems in modern human life.

4.1 Google AI

Google is one of the main entities working on artificial intelligence. Alpha Go Zero, Alpha Zero, Tensorflow, Apache Beam, TPU and Edge TPU are some of the most impacting research focuses that has shaped AI.

4.1.1 Alpha Go Zero

Go[56] is one of the most famous Chinese board games. This game is considered to be one of the most complex games among other board games like chess or checkers. Google Deepmind is one of the pioneers worked on solving this game with AI. Alpha Go Zero was the AI solution provided. The core of Alpha Go Zero is designed with deep reinforcement learning techniques. The main focus is to not to train the Alpha Go Zero with more data, but with more experience on playing the game. Alpha Go Zero itself is an agent which tries to play the game by increasing the possibility of winning. As deep reinforcement learning is used the objective is training an agent such that each move played in the game increases the reward given towards it. An agent is capable of being strong with play when it has more experience through playing many games. At the beginning of training Alpha Go Zero, it was trained with 100,000 known games played by different amateur players. At this stage, after training for this much games, it is a better version than that of the version played 1000 games. The main challenge is the need of more games when it is needed to be trained beyond the capabilities of an amateur player. So far the main challenge is creating infrastructure and algorithms to facilitate 100,000 games. Without more data, the only way to improve the AI to exceed a human player is to make the AI play itself. The idea is to generate an older and more experienced version of the existing AI player. Alpha Go Zero has been trained 30M games with itself in such a way that it creates a version of itself by being better than the old version. So every time it plays, a newer version is made, outperforming the previous version. There have been many studies done on understanding the nature of an AI game player exceeding the best level of a human player. A most important study is to understand the level of a human player through AI techniques, most preferably by deep reinforcement learning and this has been discussed by Hassabis et.al [57]. Besides, intensive searches are being done on deciding positions for a game as stated in the study by Silver et.al [58]. The AI modelling itself is computation-intensive as there are exhaustive tree searches that are being happened to find a better move. But these explorations are not like brute-force searches as seen in classical chess engines. They are driven by human intuition and intuition gained by experience.

4.1.2 Alpha Zero

Alpha Zero[59] is the generalized framework which unifies the superhuman player level in playing the game, Go, Chess and Sogi (Japanese Chess)[60, 61]. The difference in this generalized framework is the number of searches are very minimum when it is compared to IBM DeepBlue[62] and Stockfish[63], Elmo[64] (state of the art engine for Sogi). The number of searches by a human grand-master is around 100 moves and the number of searches by a state of the art chess engine is around 10 million. But the Alpha Zero only use 10,000 searches per move. These computation must take place as fast as possible and the underlying software and hardware influences this. Stockfish and Elmo have

used 44 CPU cores while Alpha Go Zero and Alpha Zero has used 4 first class TPU cores and 44 CPU cores. The performance of Alpha Zero in different game formats is shown in figure 3.

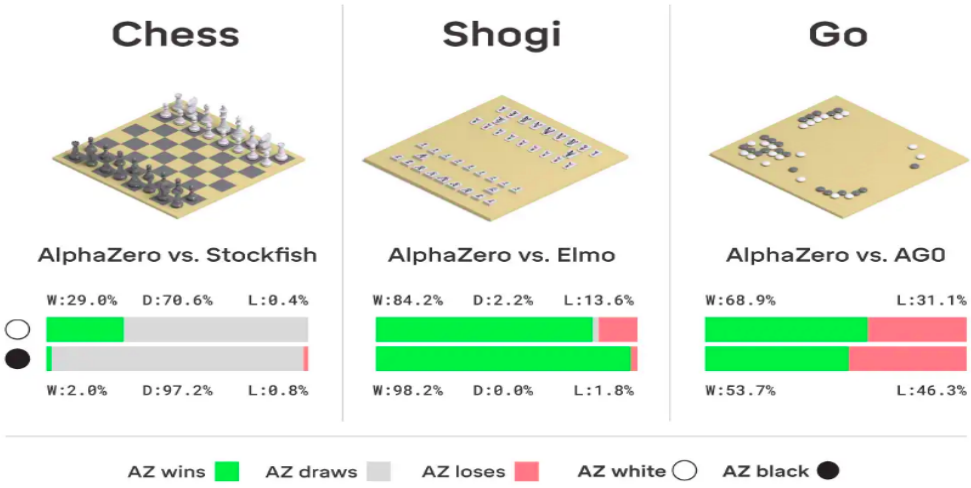


Figure 3: Alpha Zero Performance in Sogi, Chess and Go with the State of the Art Game Engines. Source [61]

With this understanding the evident fact is the need of a big data system in the expansion of multiplayer games can be vital in handling large streams of data in both batch and streaming mode.

4.1.3 Google Assistant

Google assistant[28] is an AI-powered virtual assistant with a voice base interface designed for interacting with human beings. A tool like this goes with a variety of AI models. Content identification, voice recognition, speech recognition, translation and signal denoising major skills needed for such a tool.

4.1.4 Apache Beam

Apache Beam[65] is the software tool which unifies both batch and stream processing[66]. The underlying concept is commonly known as Google-Dataflow. In big data systems, the most challenging task is to keep batch processing and stream processing in a unified manner. Also, due to the existence of a large amount of big data systems, providing support to existing consumer base on each system is a vital task. Apache Beam encapsulates all these requirements within the core of Google Dataflow model where one can program using PCollections or similar API endpoints in Apache Beam. There are runners in Apache Beam for other big data systems like Apache Spark and Apache Flink. In developing state of the art AI-systems, a tool like Apache Beam can be vital to unify data, AI models and providing services at scale.

4.1.5 Tensorflow

Tensorflow[5] is one of the main software development tool kit (SDK) to develop AI models. Tensorflow provides a variety of API support for developing deep learning, deep reinforcement learning and machine learning applications. This software stack is optimized to run on CPU, GPU and especially on TPU. In current research domain, Tensorflow is performing far better on TPUs than GPUs. The computation shift from GPU to TPU was enabled by the support in Tensorflow. In order to reduce the training AI models, Tensorflow has provided distributed training on CPU, GPU and TPU devices.

4.1.6 TensorflowLite

With the increasing demand for efficient predictions or inference on mobile devices, Tensorflow has been optimized to convert heavy models to fit into mobile devices. TensorflowLite[41] is the software stack that enables the deployment of quantized mobile-friendly networks at scale. The uniqueness of this platform is that it supports mobile operating systems like Android, IOS, Raspbian and also support Google Coral edge devices specialized on TPUs (described in section 2.4 and 3.2).

4.1.7 Hardware Oriented Software Stack

Tensor Processing Unit (TPU) is one of the main hardware accelerators developed to improve deep learning research at scale. TPU devices are a special type of hardware like GPU, but very different in architecture. TPU has been designed to do training in large scale. And also it is much faster than GPU in training (scales well beyond 1 TPU core[38]). The software stack developed with Tensorflow provides software abstraction on TPUs to design deep learning and reinforcement learning applications. In deploying trained models with high efficiency, Google Edge TPU hardware and TensorflowLite software stacks can be used to harness the power of TPUs (discussed in detail in 2.4 and 3.2).

4.2 Facebook AI

Facebook is another entity driven towards AI. The work done in Facebook relates to developing software tools and hardware tools. And also there are other supporting tools to enhance the AI development experience.

4.2.1 Pytorch

Pytorch[6] is one of the main software development kit (SDK) for AI modelling. This is a production from Facebook AI. Pytorch provides support to design dynamic graph definitions in designing deep neural networks. The dynamic graph execution is one of the most significant aspects of Pytorch. In the beginning, Pytorch was only designed as a research tool on Facebook. The production was done on Caffe2[67]. When deploying an AI application at scale, a research-level application had to be converted to Caffe2 format. In order to reduce the overhead in application development, Pytorch 1.0.0 was introduced by unifying these APIs (the currently available version is Pytorch 1.3.0). In deep learning application development, the main overhead in the development stage is the training time. In order to minimize the training time, Pytorch has enabled distributed training mode[68]. Pytorch supports MPI[69], Gloo (Facebook distributed runtime)[70] and NCCL (Nvidia) [71]. With the variety of distributed environments, the AI models can be trained efficiently on Nvidia GPUs, HPC hardware and also in Google TPU hardwares[72]. For low-latency inference, Pytorch Mobile[42] provides an efficient workflow for deploying mobile-friendly AI models for inference purpose. In addition to training and inference, training with data encryption is a vital element in dealing with sensitive data[73, 74, 75]. Crypten[76, 77] is one such framework built on top of Pytorch to make a software interface available for such application development. Another important aspect of AI modelling is understanding what happens in the models in the training stage. Captum[78, 79] is a model interpreting tool designed on Pytorch. In optimizing a model to obtain better accuracy, learning meta-data like cross-validation accuracy, top-5 accuracy and gradient overlay are important hyper-parameters. Capture encapsulates these requirements for both text and pictures related AI system development. Detectron[80] is an object detection AI software toolkit. And Fairseq[81] is a translation, summarizing and language modelling AI toolkit developed with Pytorch.

4.2.2 Poker Game with AI

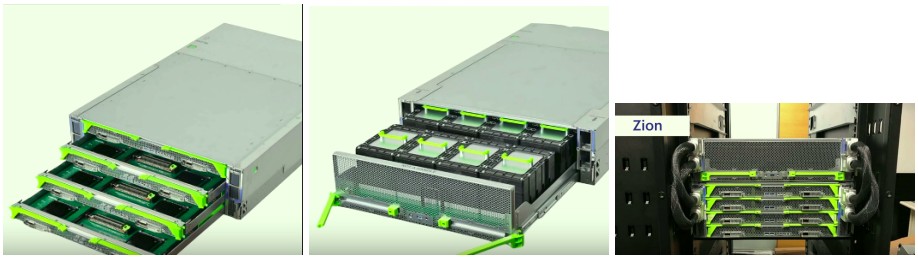
In AI-oriented game development, Facebook has partnered with Carnegie Mellon University to produce an AI-based Poker game which can play with 5 human players. Among 2 player dominant AI-based games like Alpha Go Zero and Alpha Zero, this AI-based Poker gaming engine is the first to solve a multiplayer game. It uses Monte Carlo counterfactual regret minimization for introducing self-play as same as Alpha Go Zero improved itself by playing with itself[82].

4.2.3 Facebook Hardware Stack for High Performance

Facebook has also designed new hardware devices to facilitate efficient training and inference at scale. Efficient video processing, image processing and content processing techniques are vital to performing quality analyze on the content added to the social network by millions of users. The hardware stack builds for this is known as Zion platform. Figure 4 shows the hardware stack used in Facebook research.

4.2.4 Facebook ONNX

From the sections 4.2.1, 4.1.5, it is evident that there are multi-disciplines in writing AI systems. ONNX is a shared model exchange which facilitates deploying AI systems to build on different software stacks using one middle-ware. Deploying AI models with multiple software backends takes time and resources. A middle-ware like ONNX can be used to unify multi-discipline AI workflows to create a streamline inference workflow. And also with the support of the Glow compiler, the output from the ONNX can be optimized to fit into different hardware for inference in different vendors. Figure 5 shows the overview of the ONNX platform with other AI platforms.



(a) (b) (c)

Figure 4: (a) CPU Chassis with 8 CPUs (b) Accelerators for Inference (c) Zion Platform Connecting CPU Chassis and Accelerator Chassis in the Rack. Source [83].

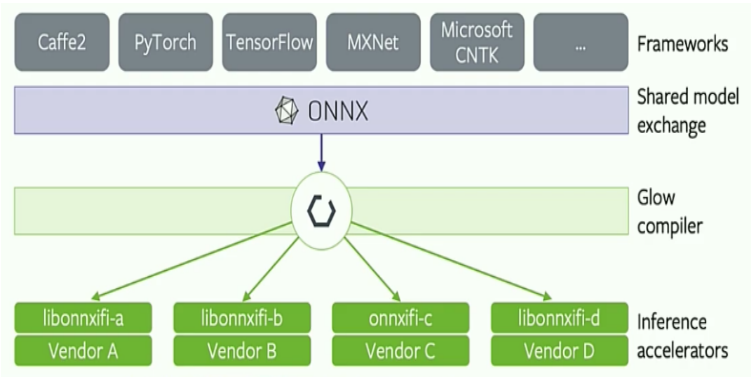


Figure 5: ONNX Platform With Other AI Platforms. Source [83].

4.2.5 Facebook AI @Scale

Scaling AI systems is one of the main challenges for many of the business entities. With the increasing consumers and devices, data management has to be optimized for higher efficiency. Most of the increasing demand is focused on text, video and image processing. In increasing efficiency, computation and memory management optimizations are vital. In AI systems, there are many layers responsible for various tasks. The lower levels are more data-oriented and depend on memory. The higher levels are more intense on computations. Understanding the overheads in each layer will allow optimizing systems for higher efficiency. Concerning computation, understanding common computations done within a larger AI system enables in reducing redundant execution. Once such common computation done in one machine is useful to a computation done in another machine, the final result can be sent through the wire. This involves lower-level optimizations on understanding the data-parallelism or model-parallelism of distributed training of such AI models. And also providing hybrid AI modelling support for *data-parallel* and *model-parallel* training is vital to optimizing modelling time and resource usage. Considering memory, using better data schemes for supporting sparse data and dense data is vital in optimizing applications. In such scenarios usage of embedding, tables[84] to encapsulate the data representations can be very effective to represent data. Instead of re-vectorizing data, with embedded table lookup, more time can be saved to find data representations for computations. With such optimizations memory bandwidth, sensitive problems can be solved [85]. Figure 6 shows the memory and computation based layer categorization in a deep neural network.

4.3 Tesla AI

Tesla AI is more focused on facilitating the autonomous experience for vehicles. Such an autonomous task needs a vast amount of sensory inputs. Most critical thing is to infer and take actions instantly. For autonomous driving, object detection, object identification, obstacle avoiding, predicting the flow of objects and many other external sensations are required to make accurate decisions. With AI modelling, the obvious solution is to create an ensemble solution for each of the components. For instance, a deep learning network for object detection and object identification can be used. But when the number of tasks scale to a larger extent, having separate AI model doesn't provide a scale-able solution. Common characteristics involved in these tasks must be identified and knowledge has to be shared among the networks to provide an efficient and scale-able solution. This provides a generalized network to execute multiple tasks with

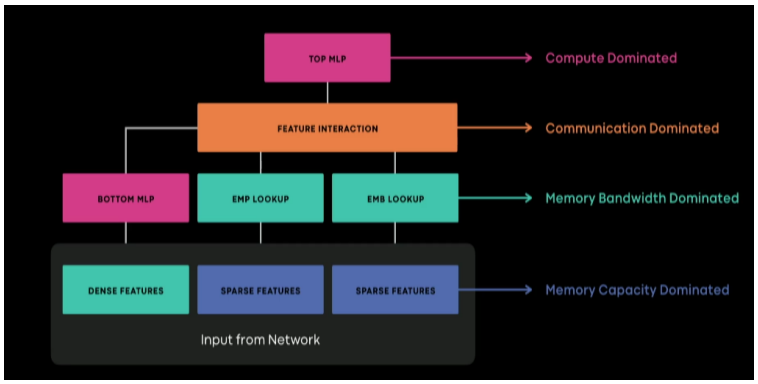


Figure 6: Neural Network Layer Categorization on Memory and Computation. Source [85].

multi-disciplines. Tesla has produced Hydra-Nets, a version of a complex AI system which enables the aforementioned functionality. A reinforcement learning network can help to link up unseen connections or find unseen features in a given event. This is vital to make automated AI. When complex decisions like steering or moving the car to a specific location has to be done, some features need to be burrowed from different hydra-nets. For instance, the depth of the picture is very important in identifying the obstacles, but at the same time, it needs to burrow some features related to the shape of the road if you need to steer. Each hydra-net is specific for different tasks. A recurrent set of tasks can be used in such a setting. Figure 7 shows the structure of the multiple networks forming a Hydra-net.

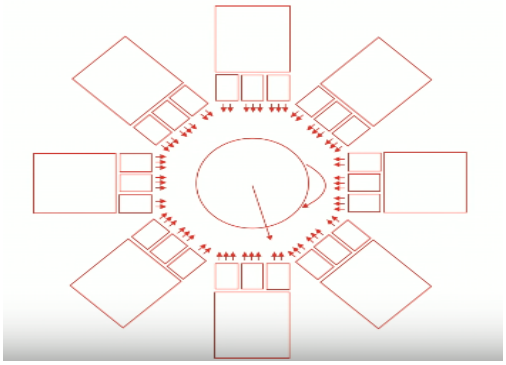


Figure 7: Architecture of Tesla Hydra-Net. Source [86]

As shown in the figure, there are 8 different tasks done by these hydra-nets. Each hydra-net extracts features and all these features are sent for a 2nd layer to do another evaluation. The speciality of each net performing a unique task is vital, but having an overall understanding to formulate a clear goal is vital when performing the global task. So the decision making is done by analyzing the features extracted by each of these systems. All these networks are recurrent neural networks. So this means it is heavy training. The main issue is these models are very large, so what happens is the amount of data that has to be processed doesn't fit the memory. In this case, the single node training won't be practical. In such a scenario, distributed training is vital. In these Hydra-Nets, 48 different networks are being used to provide 1000 tensor outputs as predictions. The estimated time of training is 70,000 GPU hours. It is clear from these stats that the model parallelism and the data parallelism must be used. Hydra-Nets consume Pytorch model parallel and data-parallel AI model designs to do the heavy lifting in distributed training. In addition to that, there are special types of chips called Dojo, designed to improve the inference performances[86].

4.4 Uber AI

Uber AI division focuses on a couple of disciplines which go in parallel with transportation. The Uber Taxi services have also been linked with a delivery system linked with a large restaurant network. In facilitating such services, traffic data understanding, estimated time evaluation on services and locating customers and other service centres are vital. Uber Driver demand prediction is a very important feature. This is vital in understanding where more drivers need to be and try to arrange resources such that a surge can be handled. Another one is the estimated meal arriving time. ETA

calculation is a very important thing. Suggestions for texting with the driver. Making things much easier when there is no room to do heavy typing. In a rainy day, this is great. Involving with large data, the main contribution from Uber to the state of the art AI ecosystem is the Horovod[87] platform. Horovod is a distributed training platform developed to facilitate distributed training for Pytorch, Tensorflow, MxNet and Keras. It uses OpenMPI[88] backend to do model synchronization in distributed training. For the heavy data pre-processing Horovod has been wrapped with Apache Spark (PySpark) to facilitate an effective workflow.

Dealing with large data pre-processing and facilitating a larger amount of services, data querying becomes a very intriguing challenge. According to Uber research, most of the data are stored contain thousands of columns, but the queries from users only require one or 2 columns to generate the expected results. Storing data in an optimized way is vital to provide efficient services. When this data is transformed into column architecture, the retrieved values are in transposed. Here the row data in the normal state becomes column data. So when you just need a few rows from the original setting, this becomes a few columns after converting to row setting. This way of data access is very efficient. This optimization is vital when such queries are present.

Understanding the business statement is vital to design a data processing system for AI applications. In analyzing more on data storage in Uber, the largest datasets are stored in a key-value store and then there is an incremental pull of data for every 30 minutes. Once the data is being pulled, it is being processed as a batch and sorted in timely passion. In Uber rides, sometimes the trips need to be updated due to the pricing of the trip. So, in this case, these records need to be retrieved. So these records are going to be updated consistently for a given period. Some transaction update can take some time to complete its process. Depending on the number of data associated with the transaction, the update process can be costly. A refund in a complex trip where there were multiple drivers and multiple trips were involved, there has to be a way to retrieve these records fast from historical data. The data has to be labelled. Once a transaction is being completed and it must be made immutable after a certain period. This kind of policy enables data being not subjected to any change.

Aforementioned business statement involves a larger portion of AI system building on data organization. Having batch and streaming data processing involved to most of the transactions causes a huge overhead in dataflow management. Most of the data processing in Uber is adopted on Hadoop based technologies. But with the complex data querying involved with incremental processing followed by batch processing, a novel framework has to be introduced to encapsulate all the business logic. In addressing this, an effective solution on top of the existing framework was designed. This framework is known as Marmaray[89]. In incremental processing and batch processing pipelines, there can be an update for an existing key-value pair. In such a scenario the record can be viewed in two formats. One format is the latest mode, which provides the data set with the final updates. The other data retrieval mode is the incremental mode, it provides the view of the latest given a previous checkpoint timestamp. Hudi[90] is a framework designed within Uber to facilitate such complex tasks[91]. Figure 8 shows the Lambda architecture in Hudi.

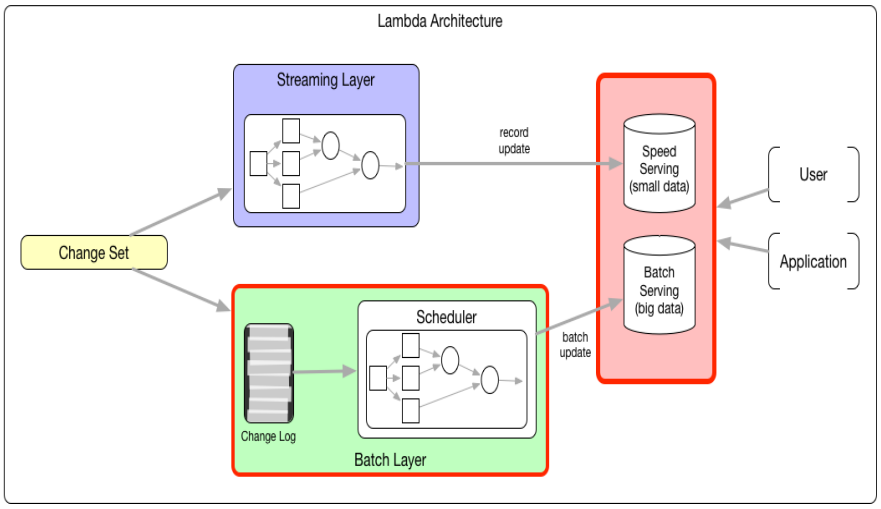


Figure 8: Lambda Architecture in HUDI. Source [90]

4.5 Microsoft AI

In modern research, Microsoft AI has been mostly focused around providing scale-able solutions via Azure cloud. In addition to that, the literature on Microsoft AI shows that their main focus is towards solving problems by providing AI as services. These are some of such influential services on AI. Vector search is an AI-powered search instead of classical index-based search. By analyzing user inputs, approximate results are being retrieved by the use of AI algorithms. Here deep learning models are being used to represent data as vectors. The distance between vectors designates the similarity of two entities. "Approximate nearest neighbour (ANN) algorithms search billions of vectors, returning results in milliseconds"[92]. Snip Insights [93] is an AI-based tool developed for identifying famous people or landmarks. Such applications are very important for people who are travelling and exploring the world. Snow leopard trust [94] is a research project on enabling a safe environment for snow leopards. In this research, scientists are using "camera traps to spot snow leopards in their natural habitats with minimal disruption". This camera traps records hundreds of thousands of images. The necessity is to identify and analyze these images. By facilitating image processing techniques developed on AI provides support for such a labour-intensive task. Ethics in AI [95] is one of the most important aspects that need to be considered. In this research, the focus on providing technology support for people with disabilities. This research questions and answers the ethical duty of AI to serve humanity. In addition to this, Cortana is another Microsoft AI advisor which is built on top of a larger ecosystem of AI technologies.

4.6 Amazon AI

Amazon is another business entity involved in AI-related research. The main applications from Amazon focus on user experience improvement on shopping, delivering, searching and conversational intelligence. Understanding user requirements via searches and existing purchases is a qualitative and quantitative study. Using classical statistical models limit the deep understanding. In recent research from Amazon, the main focus is on deep learning-based research. Amazon Alexa is one of the main research outputs from these AI-related research. This contains a larger area of expertise to produce a natural conversational AI. At the moment, Alexa also serves as a digital advisor with the capability of linking up with external services like weather, news, music, video, smart devices, etc. Home automation and voice navigated instructions to control electronic services is one of the outcomes of this AI advisor.

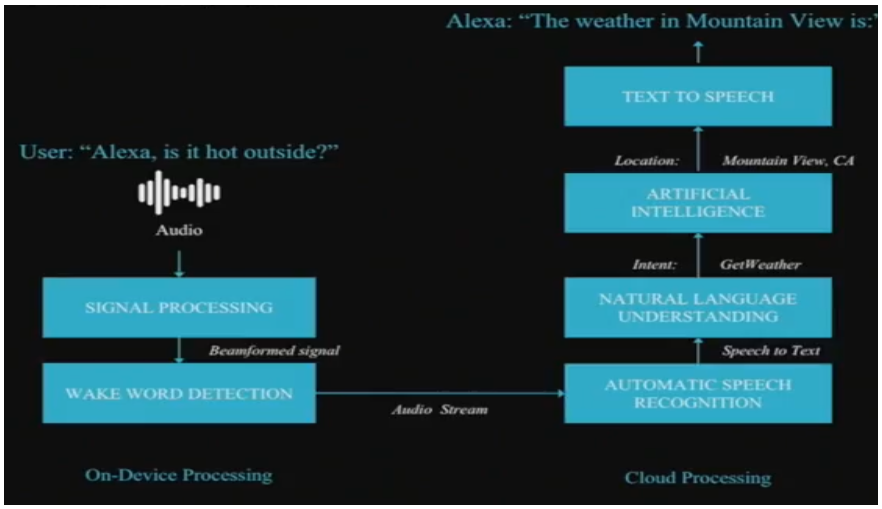


Figure 9: Amazon Alexa Workflow. Source [96]

Natural conversation is one of the most important aspects to provide a streamlined service. For such services, AI models developed using deep learning for signal processing and denoising are used. In addition to this, data pre-processing at the Edge is one of the most effective usages of network bandwidth. The Alexa AI or the inference models are hosted on the cloud, so the synthesised speech is sent to the cloud with pre-processing. The task offloading from Edge to Cloud is an important aspect that is considered in Alexa to provide low-latency responses. In the cloud, this data is being inferred with relevant AI models and results are communicated the user as voice. Signal denoising and voice recognition are very important technologies at the user end. In addition to this understanding complex sentences and infer the meaning from such sentences is vital for a digital advisor like Alexa. Figure 9 shows the workflow associated with Amazon Alexa.

Amazon SageMaker[97] is another tool developed to cover a subsection of AI, machine learning. This tool is a complete workflow manager which enables data cleaning, machine learning algorithm modelling, auto-tuning, monitoring, deploying and maintenance. It covers the life-cycle of a machine learning related project. Most of the functionalities are around auto-tuning and automating most of the redundant work in machine learning algorithm modelling. The workflow of SageMaker can also be used to develop AI workflow management.

5 IBM with Big Data Stack

There are a set of application areas identified by IBM research on data analytics and simulation workflows. Data analytics is mostly done by big data systems. The simulation workflows are mainly done by high-performance systems. In addition to that, there are some applications and research interests overlapping both of these areas. The convergence of the high performance and big data stack provides a highly productive and high-performance region. High productivity enables powerful APIs for data organization. And also the power of high-performance computing literally enables the high performance.

Under the data analytics category, there are tasks related to data organization, data processing and interpretation. Data organization takes place at the source of the data pipeline. When dealing with the data generated by different sources, handling unstructured or semi-structured data is a very important aspect of the first step towards data organization. Spark offers a structured streaming API for handling structured data. This enables schema for the data stream, and this is very useful in the higher levels of the data pipeline. For any other big data system, it is vital to have structured data processing capability. Down the data pipeline, the data filtering, grouping takes place. Depending on various application requirements it is vital to understand how to provide such capabilities. These steps basically cover the data organization and data processing section of data analytics. This is the generic data-oriented workflow in a big data system. Due to the exponential growth of data, centralized data processing won't become a practical approach. In the decentralized data processing, providing big data systems to extend the data processing capabilities from cloud to edge is vital. Task offloading from cloud to edge is enabled by this way. So the overhead caused by sending raw data over the network can be eliminated by such a process. In addition to that, data summarizing or data compression is also a very important task that needs to be done on the edge. For such cases, PCA, TSNE[98] or lightweight DNN models can be used.

The data simulation category is highly associated with various model simulations in domain science problems. Most of the data associated with this area are structured. The reason is most of these models are designed for domain science-related research. The expected outcome is not known but the structure of the outcome is known as the data sources are the endpoints of the designed simulators. In such cases, the overhead is data organization is minimum. The strength of this category mostly depends on doing computations much faster. Exascale experimenting is one of the strengths in the data simulation related applications. The objective of this application category is to enable high-performance applications.

Enabling better systems for understanding data and making new interpretations of data is the main goal. Both data analytics systems and data simulation systems have strengths and weaknesses. For instance, data analytics systems are strong in data organization but not as competitive as data simulation systems in processing data with high efficiency. This effect is vice-versa. The overlap of these systems is very valuable to build a unified system which has the strengths of both data analytics and data simulation systems. The challenge is understanding how such frameworks can be designed aligned together to support efficient applications.

With this study by IBM on the overlap of the data analytics and data simulation frameworks, the design of such a system comes under four main layers.

- **Application Level:** Involves with both big data applications and applications and community codes from various domain sciences.
- **Middleware and Management:** In this layer, as per big data model, it contains the distributed data management, data processing APIs and all the software developed for these purposes. For HPC mode, there are MPI, OpenMP related application development libraries and supporting development tools.
- **System Software:** For big data model, virtualization, containers and other cloud services related tools can be noted as core components. For HPC model, specific container management tools like Singularity, Shifter can be denoted. But all the components lie upon the Linux OS variant.
- **Cluster Hardware:** Cluster hardware is the lowest layer where the network connections and network storage for both HPC and big data system stands. For big data applications, Ethernet switches, local node storage, etc can be denoted as sub-components. For HPC model, Infiniband, Ethernet switches, GPU accelerators and other In-situ processing components can be noted.

As the idea is the converged software development, the best way to support the overlap is by providing support for the big data stack to match with the high-performance application development. The reason is the necessity always come with easiness to develop applications. That is one of the most significant qualities of the big data systems. So the overlap of these two domains must provide MPI as a service, big data tools as a service, web app as a service, GPU as a service and similar components that could make an impact. Hereafter the term **"The Overlap"** refers to the system design with the overlap of the aforementioned system disciplines. Figure 10 shows the Apache Spark related application stack developed by IBM.

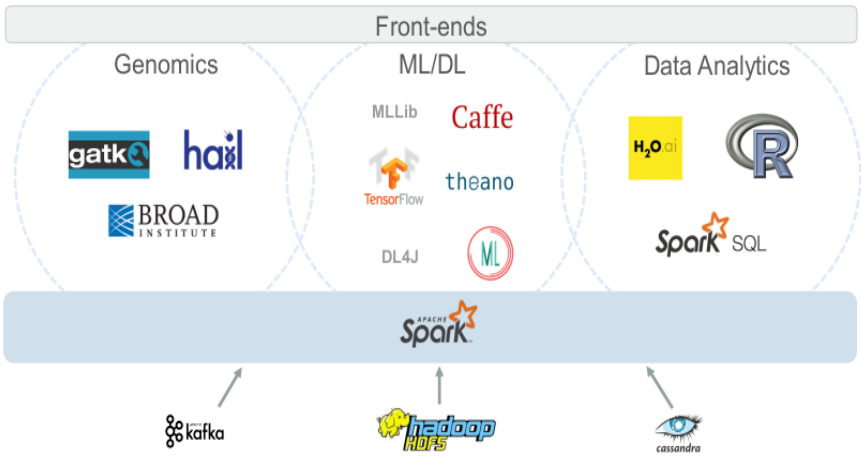


Figure 10: Apache Spark related Application Stack

Figure 10 shows the generic layout of a system designed with the overlap of big data and HPC model. Under this unified framework, there are three main tracks of system design.

- **Linking with Scientific Applications:** There are many tools designed for climate modelling, earthquake analysis, genomics analysis, etc. Re-writing them in the new APIs created by the overlap. So the best model is to support the applications via creating interfaces to connect the overlap and these scientific applications.
- **Machine Learning and Deep Learning Support:** Many frameworks are supporting the development of streamlines machine learning and deep learning application. So rather than re-writing these codes, it is better to link them with the existing frameworks.
- **Data Analytics Support:** SQL, R and frameworks like H2O offers a variety of APIs to do data analytics in a streamlined passion.

The main concern with linking up the high-performance programming models with the big data stack is the language barrier. For the cause of much easier programmability, JVM oriented languages are extensively used in developing big data frameworks. But almost all the high-performance applications are designed with C++/Fortran or FPGA. Apart from the language boundary, linking up JVM oriented programs to support accelerated hardware devices like GPUs or TPUs is a great challenge. The best way is to design linking programmes between each of these programming stacks. But still, the data movement issue has to be handled when data has to be copied from CPU to GPU for optimized computations involved with matrix or vector computations. Spark provides optimizers for JVM related application development with Tungsten and Catalyst projects. Similar optimizers are necessary for other big data frameworks to support optimum performance. The first line of optimizations comes under the language level optimizations on JVM. IBM has optimized several Java implementations (OpenJDK and OpenJ9).

5.1 JVM Optimization

OpenJ9 is one of the prominent work from IBM for JVM optimization. This has been tested with Apache Spark and the results show that there is a significant improvement in performance with OpenJ9. Some of the improvements are as follows.

- Java object models with smaller footprints

- Effective Garbage Collection
- Efficient JVM lock contention schems
- JIT (Just-In-Time compilation)
- GPU-enabled JIT
- Shared classes technology

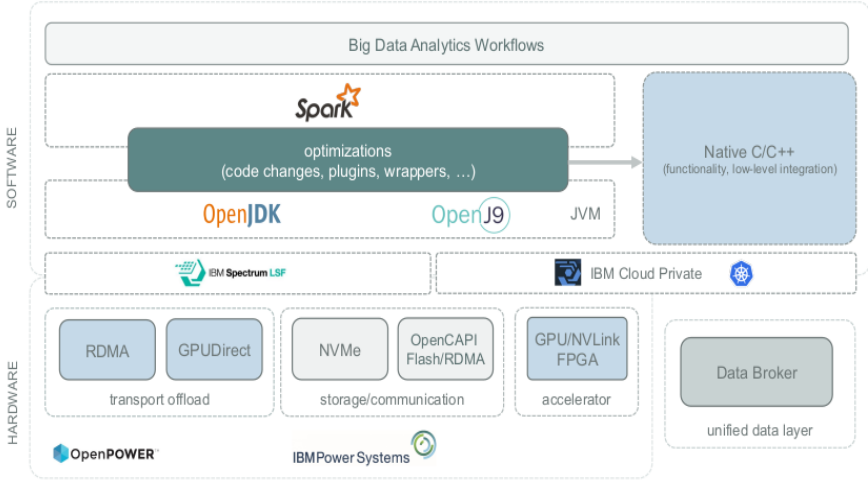


Figure 11: Language Optimization Support for Apache Spark

Figure 11 shows the language level optimization added big data system evolved around Apache Spark. Another task of the language level optimization is to provide native C/C++ performance by using JNI for tasks to get better performance. Underneath the language optimization layer, there exists the scaling capability obtained from IBM Spectrum and IBM Cloud Private. Underneath this layer, there exist communication, storage/communication and accelerator layer. The communication layer is associated with RDMA and GPUDirect. The storage/communication layer is associated with NVMe, OpenCAPI and Flash/RDMA. The accelerator layer is associated with GPU/NVLink and FPGA. Apart from these, there are connectors to online message processing via message brokers.

5.2 GPU Acceleration

IBM has worked on a GPU-enabled version of Apache Spark, called IBMSparkGPU. The compute-intensive workload is being allowed to run on GPUs. Here the support for Spark Mlib, SparkSQL and GrapX has been given to run on GPUs. Besides, the GPU code generation is done using Tungsten. And the other way of supporting GPU Acceleration is providing support for GPU-enabled data analytics or simulation platforms. For instance, Tensorflow, Pytorch and CuDNN (with Rapids) already support GPU with their underline architecture. So it is always easier effective to support such platforms to harness the power of GPU for big data analytics. Optimizations associated with these system design always couple with JVM. And the optimizations are done for JVM as mentioned in section 5.1, have enabled better performance for Apache Spark.

5.3 Data Broker

IBM-Databroker is the message broker level optimization added for the overlap. It is nothing but an in-memory key-value store enabling applications to share data using one or more namespaces. In a traditional dataflow model program, if a message broker is not used, the data is fed to the analytics engine in terms of File (I/O) or sockets. With a file-based approach, the latency can be very high, as the data has to be load from the disk into the memory. This becomes a task with a longer latency. Sockets, the issue is the same, the latency is very high. The advantage of a data broker is that a data broker can be used to link the distributed data over multiple nodes using the DRAM. This is much effective than using I/O oriented methods. In this setting, the latency is relatively lower than that of an I/O oriented method. Another advantage of using a data broker is app discovery can be effectively carried out. To improve the performance in Spark a data broker mechanism has been introduced by IBM. In Spark for shuffling the data, blocks are stored on disk. The main overhead lies in the OS as the I/O operations are intensive. With a data broker, a simpler shuffle is implemented with No File I/O, disk access and no sorting.

6 Distributed Data Paradigm

Distributed data paradigm contains business entities and various resources they make. Apart from the large business entities, there are other academic and non-academic related entities developing such systems.

6.1 K3s and K8s For Containing

In scaling applications with the virtual machines, Kubernetes or K8s related software stack provides a much easier way of scaling applications. Kubernetes provides support in managing containerized workloads and services. There are two types of workloads associated with modern-day systems. Things that run on larger hardware like data centres and things that run on much smaller hardware like decentralized Edge-devices. In both these scenarios, containerization is vital. K3s becomes the lightweight Kubernetes for Edge-devices. In scaling deep learning models in the Edge devices, K38s can be used to do better scaling and maintaining models[99].

6.2 Streaming for AI with Message Brokers

Streaming machine learning and streaming for deep learning inference need state of the art message brokers to make the workflow streamlined. In such cases, a few of the current state of the art streaming brokers are Apache Kafka[100], ZeroMQ[101] and RabbitMQ[102]. Among these, Apache Kafka provides a streamlined API in Java and Python. This enables easier application development and connector design for other big data systems. When connecting multiple systems which work on different disciplines, message-brokers are the best way to provide access to different services. There are many layers involved in these large big data ecosystem. There is a huge issue in combining all these layers together to enable a complete workflow. Linking each of these components, Extract, Store, Transformation, Load and Store is a challenge. That is the main reason most of the application developers use message brokers to link different services and systems[103].

6.3 Lambda for Incremental Processing

Lambda architecture uses both batch processing and stream processing to enable the increasing requirement for low-latency application development with the rise of data contained in the big data applications. This is more like a definition of data processing rather than a system design.

6.4 Storage Handling

In the earliest days of handling big data, Hadoop Distributed File System (HDFS) was the most prominent way of managing the distributed data. With the increasing demand for data, the MapReduce came into the picture. MapReduce concept is all about processing the distributed data in terms of a mapping process and reduction process. In a mapping process, data is being filtered or transformed. In a reduction process, data summarizing is being handled by arithmetic or algebraic expression. All these processes are happening in a distributed manner and HDFS supports the data management, basically, data read, write and delete. But with the increase of various applications, there are requirements on retrieving a part of a data record. For instance, a record contains sub-records. Generally, these records are representing in row formats. The issue with the row format is, the data storage is not well compressed and not I/O efficient when records have to be retrieved in such a way that, a few elements in a row of data records are useful for the query. In supporting such requirements, a different storage mode can be efficient. With memory architecture, a columnar data structure with more data compression can be used to do effective querying. Parquet is one such columnar data storage format and the Apache Arrow is a cross-platform in-memory columnar data format. These new trends have improved data storage and data handling across multiple big data platforms.

6.5 Databricks

Databricks is the Apache Spark-based entity designing a variety of solutions for the big data stack. They support all of the states of the art data processing functions like data storage, data querying, data analytics in both streaming and batch formats, machine learning and deep learning. Databricks support application development especially on Jupyter notebooks with PySpark, a python wrapper for Apache Spark Scala core. This allows most of the application developers to design applications with much ease and it allows connecting with most of the state of the art python-based APIs like PyArrow, Tensorflow, PyTorch, MXNet, Pandas and many other such libraries. In addition to that, the MLFlow system designed to evaluate machine learning and deep learning algorithm training and testing have been a new addition. It provides an API abstraction linked with Tensorflow to evaluate the training process. All these additions to the Apache Spark big data system enables application development much more streamlined than the other big data

systems. In addition, another new addition is the structured streaming API of Apache Spark supporting schema-based data processing with the SQL-enabled streaming API.

6.6 Harp

Harp[104] is one of the high performance communication library designed on top of the Java language. Due to usability in Java, extending to ML and similar workflow development can be enhanced by Harp communication collectives. Like MPI, Harp offers similar collectives like, reduce, allreduce, gather, allgather, broadcast and another Harp specific, rotation. Harp-DAAL[105] is an extension to Intel DAAL[106] library which facilitates deep learning and machine learning performance improvement on Harp.

6.7 Twister2

Twister2[107] is a big data system enabling both batch and stream processing with the deployments on Kubernetes, Mesos, Slurm and also a MPI backend. Twister2:Net[108] is the core of Twister2 framework. It enables writing both batch and streaming applications with a unified API. Twister2 also supports machine learning applications. Twister2 has a dynamic task graph design which enables designing complex applications in a streamlined passion. Twister2 default runtime is developed on top of MPI ISend and IRecv communication protocols. This allows Twister2 to do collective communication much faster than other state of the art big data systems.

6.8 BigDL

BigDL[109] written on PySpark. These frameworks also support developing deep learning applications but now most of the applications have been developed on Tensorflow and Pytorch.

6.9 Weka and Moa

Weka[110] is one of the most prominent machine learning designed by the academic institute, University of Waikato. This is a complete library machine learning with data processing and data organization. Weka has also extended the classical batch mode applications into streaming applications by using Moa[111]. Moa is a streaming machine learning library built with the core of Weka.

6.10 Apache Samoa

Apache Samoa[112] is another streaming machine learning framework which is inactive at the moment. Apache Samoa supported creating a complete streamlined workflow for streaming machine learning algorithms. It also supports the streaming workflow with Apache Flink and Apache Storm.

6.11 H20

In addition to this, there are some other Deep Learning related tools like. In addition H20 is also another distributed library which supports development of deep learning and similar workflow applications with in-memory distributed computing capability[113].

6.12 GraphX

Apache Spark's Graphx[114] is one of the earliest high performance graph processing library built on big data stack. It supports graph processing with the core of Apache Spark. Graphx allows to do distributed iterative graph processing within a single system.

6.13 Snap

Snap [115] or Standford Network Analysis Platform is defined as a general purpose network analysis and graph library. This library is written in C++ and graph implementations designed with this tool scales for hundreds of millions of nodes with billions of edges. This framework also has a Python interface called SNAP.py.

6.14 Petuum

Petuum[116] is a distributed machine learning library which supports both data and model parallelism. This framework supports matrix factorization, Lasso, SVM, Convolutional Neural Networks, Latent Dirtchlet Allocation, Logistic regression, KMeans, etc. This is one of the most prominent machine learning libraries which support model parallelism.

6.15 DeepDriveMd

DeepDriveMD[20] is a deep-learning-oriented protein fold simulation library with adaptive simulations. This tool has better performance over other protein fold simulation systems because of the usage of deep learning analysis. It enables the effective learning of the latent representations and drive adaptive simulations.

6.16 Deep Learning and Machine Learning Frameworks with Middle-ware Support

The deep learning or machine learning systems consumes a vast range of other resources. In the universe of these complex systems, this AI component recognized as an ML component in figure 12, is just a tiny piece. There are other layers which facilitates the functionality of the ML system.

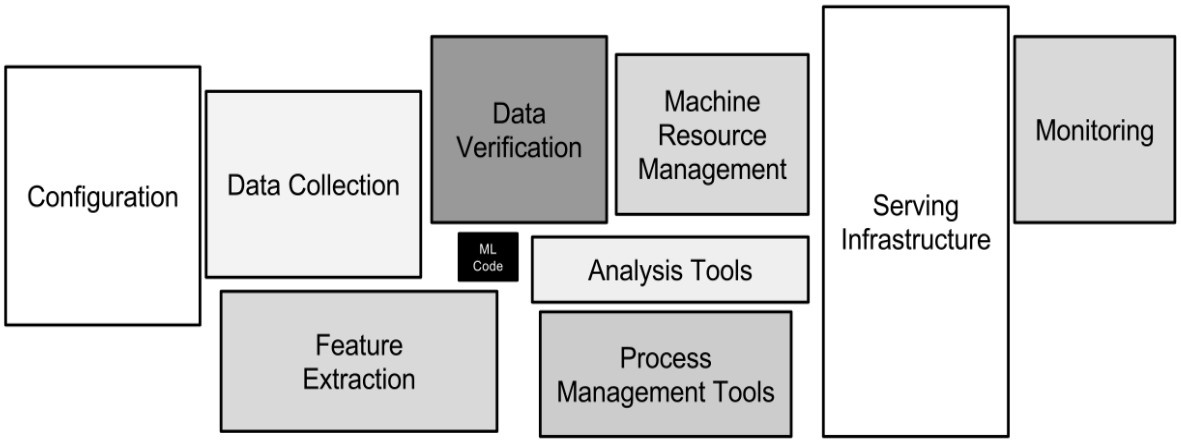


Figure 12: System Overview of an Intelligent System

In the table 1 shows the AI frameworks along with the third-party and core systems supported in scaling and providing support for multiple applications.

In addition for graph neural networks a framework called Deep Graph Library[117] also supports with the state of the art platforms like Amazon SageMaker, Apache MXNet, Pytorch and Tensorflow.

Active or Inactive nature is decided upon the recent releases of the library.

7 Breaching the Programming Language Barrier

The main reason for big data systems not being popular among scientists is the learning curve of programming languages like Java, Scala, C++, etc. Most of the big data systems are designed with Java and Scala while high-performance systems are designed with C++. This is where python becomes the most helpful programming language in breaching the gap between a domain scientist and a big data system. Apache Spark, Apache Flink and many other big data systems provide a Python API. In addition to these big data systems, there are other systems designed by both HPC and Big data community to provide high-performance data solutions to scientists.

7.1 Dask

Dask is a distributed computing platform written for Pythonic applications. Dask support data organization with easy-in-memory loading, support for Numpy and Scipy. Besides, Dask is a frontier python framework which supports

Table 1: AI Framework and Middleware

AI/BigData Framework	Middleware	AI Major	Status
Scikit-Learn	Dask	Machine Learning	Active
Tensorflow	Apache Spark, Google XLA	Deep Learning	Active
Pytorch	Apache Spark	Deep Learning	Active
MXNet	Horovod	Deep Learning	Active
Weka	MOA	Machine Learning	Active
Spark Mlib	Apache Spark	Machine Learning	Inactive
Apache Samoa	Apache Samoa	Streaming Machine Learning	Inactive
Keras	Tensorflow, Theano	Deep Learning	Active
Graphx	Apache Spark	Graph Computation	Active
Snap	C++ and Python	Graph Computation	Active
Petuum	Distributed Model and Data Parallel Native System	Machine Learning	Active
H20	Multithreaded MapReduce	Machine Learning	Active
BigDL	Apache Spark	Deep Learning	Active
Deep Graph Library	Native Python Libraries Amazon Sagemaker Apache MXNet	Graph Neural Network	Active

many other python-based data analytics tools like, Scikit-Learn, Scikit-Image, Rapids, etc. Because of the high usability of python and support of Dask for multiple frameworks, this is being used by many researchers to do domain science research.

7.2 Epython

Epython[118] is a lightweight python API developed for application development in micro-architectures. In this design, the interpreter and the runtime resident has an actual memory size of 24KB. And this implementation works both with many-core processors executed independently and co-processors with some extra shared memory between the host.

7.3 PyComps

PyCOMPS[119] is a framework built on Python to support parallel computation workflows. The APIs in PyCOMPS allows users to write the programmes sequentially but use functions with asynchronous tasks by annotating them. This allows domain scientists to write parallel programmes with much ease.

7.4 Dislib

Dislib[120] is a distributed computing library providing distributed algorithms for scientific computations. This library is highly focused on machine learning algorithms. This is a library designed on top of PyCOMPS7.3 library.

7.5 Parsl

Parsl[121] is a Python oriented framework designed to run programs on any compute resource from laptops to supercomputers. This framework also supports annotation oriented methods to do parallel programming and the APIs have abstracted the parallel logics from the user so that parallel programmes can be written in a streamlined passion.

8 Conclusion

AI systems consisting machine learning, deep learning and reinforcement learning systems are always coupled with data and simulations. In designing larger ecosystems capable of solving state of the art scientific and non-scientific problems, the big data frameworks associated with data processing can be very vital in designing such systems. From this study, the conclusion made is that machine learning, deep learning and reinforcement learning are vital, but these applications cannot exist without the support of high-performance big data systems.

References

[1] Developer resources for deep learning and ai | nvidia. https://www.nvidia.com/en-us/deep-learning-ai/developer/. (Accessed on 12/02/2019).

[2] Cloud tpu | google cloud. https://cloud.google.com/tpu/. (Accessed on 12/02/2019).

[3] Google. https://www.google.com/. (Accessed on 12/02/2019).

[4] Artificial intelligence computing leadership from nvidia. https://www.nvidia.com/en-us/. (Accessed on 12/02/2019).

[5] Tensorflow. https://www.tensorflow.org/. (Accessed on 12/02/2019).

[6] Pytorch. https://pytorch.org/. (Accessed on 12/02/2019).

[7] Facebook ai. https://ai.facebook.com/. (Accessed on 12/02/2019).

[8] Apache mxnet | a flexible and efficient library for deep learning. https://mxnet.apache.org/. (Accessed on 12/02/2019).

[9] Ai with aws machine learning. https://aws.amazon.com/ai/. (Accessed on 12/02/2019).

[10] Nvidia cudnn | nvidia developer. https://developer.nvidia.com/cudnn. (Accessed on 12/02/2019).

[11] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[12] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[13] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[14] Geoffrey E Hinton. Learning multiple layers of representation. *Trends in cognitive sciences*, 11(10):428–434, 2007.

[15] Geoffrey E Hinton. To recognize shapes, first learn to generate images. *Progress in brain research*, 165:535–547, 2007.

[16] Geoffrey Hinton. Where do features come from? *Cognitive science*, 38(6):1078–1101, 2014.

[17] Uber ai home | uber ai. https://www.uber.com/us/en/uberai/. (Accessed on 12/02/2019).

[18] Autopilot | tesla. https://www.tesla.com/autopilot. (Accessed on 12/03/2019).

[19] Ai platform | microsoft azure. https://azure.microsoft.com/en-us/overview/ai-platform/. (Accessed on 12/03/2019).

[20] Hyungro Lee, Heng Ma, Matteo Turilli, Debsindhu Bhowmik, Shantenu Jha, and Arvind Ramanathan. Deepdrivemd: Deep-learning driven adaptive molecular simulations for protein folding. *arXiv preprint arXiv:1909.07817*, 2019.

[21] Alibaba. Machine learning platform for ai: Data mining & analysis - alibaba cloud. https://www.alibabacloud.com/product/machine-learning. (Accessed on 12/02/2019).

[22] Robert Roe and Chirstoph Angerer. Deep learning drives new science. *Scientific Computing World*, 2019.

[23] Quintin Fettes, Mark Clark, Razvan Bunescu, Avinash Karanth, and Ahmed Louri. Dynamic voltage and frequency scaling in nocs with supervised and reinforcement learning techniques. *IEEE Transactions on Computers*, 68(3):375–389, 2018.

[24] Anna Zamansky, Dirk van der Linden, Irit Hadar, and Stephane Bleuer-Elsner. Log my dog: perceived impact of dog activity tracking. *Computer*, 52(9):35–43, 2019.

[25] Mark Salisbury. When computers advise us: How to represent the types of knowledge users seek for expert advice. *Computer*, 52(9):44–51, 2019.

[26] Apostolos P Fournaris, Aris S Lalos, and Dimitrios Serpanos. Generative adversarial networks in ai-enabled safety-critical systems: Friend or foe? *Computer*, 52(9):78–81, 2019.

[27] Amazon. Amazon alexa - wikipedia. https://en.wikipedia.org/wiki/Amazon_Alexa. (Accessed on 12/02/2019).

[28] Google. Google assistant, your own personal google. https://assistant.google.com/. (Accessed on 12/02/2019).

[29] Apple. Siri - apple. https://www.apple.com/siri/. (Accessed on 12/02/2019).

[30] Microsoft. Cortana - your personal productivity assistant. https://www.microsoft.com/en-us/cortana. (Accessed on 12/02/2019).

[31] Robert Roe and Loren Dean. Beyond the 'i' in ai. *Scientific Computing World*, 2019.

[32] Rob Johnson. Road to ai. *Scientific Computing World*, 2019.

[33] Apache kafka. https://kafka.apache.org/. (Accessed on 12/03/2019).

[34] Ammar Ahmad Awan, Karthik Vadambacheri Manian, Ching-Hsiang Chu, Hari Subramoni, and Dhabaleswar K Panda. Optimized large-message broadcast for deep learning workloads: Mpi, mpi+ nccl, or nccl2? *Parallel Computing*, 85:141–152, 2019.

[35] Dhabaleswar K Panda, Ammar Ahmad Awan, and Hari Subramoni. High performance distributed deep learning: a beginner's guide. In *PPoPP*, pages 452–454, 2019.

[36] Ammar Ahmad Awan, Arpan Jain, Ching-Hsiang Chu, Hari Subramoni, and Dhabaleswar K Panda. Communication profiling and characterization of deep learning workloads on clusters with high-performance interconnects. *IEEE Micro*, 2019.

[37] Rajarshi Biswas, Xiaoyi Lu, and Dhabaleswar K Panda. Accelerating tensorflow with adaptive rdma-based grpc. In *2018 IEEE 25th International Conference on High Performance Computing (HiPC)*, pages 2–11. IEEE, 2018.

[38] Kun Yang, Yi-Fan Chen, George Roumpos, Chris Colby, and John Anderson. High performance monte carlo simulation of ising model on tpu clusters. *arXiv preprint arXiv:1903.11714*, 2019.

[39] Gu-Yeon Wei, David Brooks, et al. Benchmarking tpu, gpu, and cpu platforms for deep learning. *arXiv preprint arXiv:1907.10701*, 2019.

[40] NVidia. Nvidia tensorrt | nvidia developer. https://developer.nvidia.com/tensorrt. (Accessed on 12/04/2019).

[41] Google. Tensorflow lite. https://www.tensorflow.org/lite. (Accessed on 12/04/2019).

[42] Facebook. Home | pytorch. https://pytorch.org/mobile/home/. (Accessed on 12/04/2019).

[43] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[44] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In *Advances in neural information processing systems*, pages 4107–4115, 2016.

[45] Minje Kim and Paris Smaragdis. Bitwise neural networks. *arXiv preprint arXiv:1601.06071*, 2016.

[46] Google. Frequently asked questions | coral. https://coral.ai/docs/edgetpu/faq/. (Accessed on 12/04/2019).

[47] NVidia. Embedded systems developer kits & modules from nvidia jetson. https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/. (Accessed on 12/04/2019).

[48] En Li, Zhi Zhou, and Xu Chen. Edge intelligence: On-demand deep learning model co-inference with device-edge synergy. In *Proceedings of the 2018 Workshop on Mobile Edge Communications*, pages 31–36. ACM, 2018.

[49] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster computing with working sets. *HotCloud*, 10(10-10):95, 2010.

[50] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, Robert Chansler, et al. The hadoop distributed file system. In *MSST*, volume 10, pages 1–10, 2010.

[51] Jaliya Ekanayake, Hui Li, Bingjing Zhang, Thilina Gunarathne, Seung-Hee Bae, Judy Qiu, and Geoffrey Fox. Twister: a runtime for iterative mapreduce. In *Proceedings of the 19th ACM international symposium on high performance distributed computing*, pages 810–818. ACM, 2010.

[52] Ankit Toshniwal, Siddarth Taneja, Amit Shukla, Karthik Ramasamy, Jignesh M Patel, Sanjeev Kulkarni, Jason Jackson, Krishna Gade, Maosong Fu, Jake Donham, et al. Storm@ twitter. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 147–156. ACM, 2014.

[53] Paris Carbone, Asterios Katsifodimos, Stephan Ewen, Volker Markl, Seif Haridi, and Kostas Tzoumas. Apache flink: Stream and batch processing in a single engine. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 36(4), 2015.

[54] Sanjeev Kulkarni, Nikunj Bhagat, Maosong Fu, Vikas Kedigehalli, Christopher Kellogg, Sailesh Mittal, Jignesh M Patel, Karthik Ramasamy, and Siddarth Taneja. Twitter heron: Stream processing at scale. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 239–250. ACM, 2015.

[55] alibaba/alink: Alink is the machine learning algorithm platform based on flink, developed by the pai team of alibaba computing platform. https://github.com/alibaba/Alink. (Accessed on 12/03/2019).

[56] Wikipedia. Go (game) - wikipedia. https://en.wikipedia.org/wiki/Go_(game). (Accessed on 12/04/2019).

[57] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[58] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

[59] Google. Alphago | deepmind. https://deepmind.com/research/case-studies/alphago-the-story-so-far#alphago_zero. (Accessed on 12/02/2019).

[60] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

[61] Google Deepmind. Alphazero: Shedding new light on chess, shogi, and go | deepmind. https://deepmind.com/blog/article/alphazero-shedding-new-light-grand-games-chess-shogi-and-go. (Accessed on 12/04/2019).

[62] Ibm100 - deep blue. https://www.ibm.com/ibm/history/ibm100/us/en/icons/deepblue/. (Accessed on 12/04/2019).

[63] Stockfish (chess) - wikipedia. https://en.wikipedia.org/wiki/Stockfish_(chess). (Accessed on 12/04/2019).

[64] elmo (shogi engine) - wikipedia. https://en.wikipedia.org/wiki/Elmo_(shogi_engine). (Accessed on 12/04/2019).

[65] Google. Apache beam. https://beam.apache.org/. (Accessed on 12/04/2019).

[66] Tyler Akidau, Robert Bradshaw, Craig Chambers, Slava Chernyak, Rafael J. Fernández-Moctezuma, Reuven Lax, Sam McVeety, Daniel Mills, Frances Perry, Eric Schmidt, and Sam Whittle. The dataflow model: A practical approach to balancing correctness, latency, and cost in massive-scale, unbounded, out-of-order data processing. *Proceedings of the VLDB Endowment*, 8:1792–1803, 2015.

[67] Facebook. Caffe2 | a new lightweight, modular, and scalable deep learning framework. https://caffe2.ai/. (Accessed on 12/04/2019).

[68] Facebook. Distributed communication package - torch.distributed — pytorch master documentation. https://pytorch.org/docs/stable/distributed.html. (Accessed on 12/04/2019).

[69] Wikipedia. Message passing interface - wikipedia. https://en.wikipedia.org/wiki/Message_Passing_Interface. (Accessed on 12/04/2019).

[70] Facebook. facebookincubator/gloo: Collective communications library with various primitives for multi-machine training. https://github.com/facebookincubator/gloo. (Accessed on 12/04/2019).

[71] Nvidia collective communications library (nccl) | nvidia developer. https://developer.nvidia.com/nccl. (Accessed on 12/04/2019).

[72] pytorch/xla: Enabling pytorch on google tpu. https://github.com/pytorch/xla. (Accessed on 12/04/2019).

[73] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International Conference on Machine Learning*, pages 201–210, 2016.

[74] Ehsan Hesamifard, Hassan Takabi, and Mehdi Ghasemi. Cryptodl: Deep neural networks over encrypted data. *arXiv preprint arXiv:1711.05189*, 2017.

[75] Qian Lou, Bo Feng, Geoffrey C Fox, and Lei Jiang. Glyph: Fast and accurately training deep neural networks on encrypted data. *arXiv preprint arXiv:1911.07101*, 2019.

[76] Facebook. Crypten: A new research tool for secure machine learning with pytorch. https://ai.facebook.com/blog/crypten-a-new-research-tool-for-secure-machine-learning-with-pytorch/. (Accessed on 12/04/2019).

[77] Facebook. facebookresearch/crypten: A framework for privacy preserving machine learning. https://github.com/facebookresearch/CrypTen. (Accessed on 12/04/2019).

[78] Facebook. pytorch/captum: Model interpretability and understanding for pytorch. https://github.com/pytorch/captum. (Accessed on 12/04/2019).

[79] Facebook. Captum · model interpretability for pytorch. https://captum.ai/. (Accessed on 12/04/2019).

[80] Facebook. facebookresearch/detectron: Fair's research platform for object detection research, implementing popular algorithms like mask r-cnn and retinanet. https://github.com/facebookresearch/Detectron. (Accessed on 12/04/2019).

[81] Facebook. pytorch/fairseq: Facebook ai research sequence-to-sequence toolkit written in python. https://github.com/pytorch/fairseq. (Accessed on 12/04/2019).

[82] Noam Brown and Tuomas Sandholm. Superhuman ai for multiplayer poker. *Science*, 365(6456):885–890, 2019.

[83] Open Compute Project and Facebook. (4) ocpsummit19 - facebook ai infrastructure- presented by facebook - youtube. https://www.youtube.com/watch?v=MYlCesArTWk&t=615s. (Accessed on 12/05/2019).

[84] Jian Zhang, Jiyan Yang, and Hector Yuen. Training with low-precision embedding tables. In *Systems for Machine Learning Workshop at NeurIPS*, volume 2018, 2018.

[85] Srinivas Narayanan and Facebook AI. @scale 2019 keynote: Ai — the next big scaling frontier – @scale. https://atscaleconference.com/videos/scale-2019-keynote-ai-the-next-big-scaling-frontier/. (Accessed on 12/05/2019).

[86] Andrej Karpathy and Tesla. (4) pytorch at tesla - andrej karpathy, tesla - youtube. https://www.youtube.com/watch?v=oBklltKXtDE. (Accessed on 12/05/2019).

[87] Meet horovod: Uber's open source distributed deep learning framework. https://eng.uber.com/horovod/. (Accessed on 12/03/2019).

[88] Open mpi: Open source high performance computing. https://www.open-mpi.org/. (Accessed on 12/05/2019).

[89] Marmaray: An open source generic data ingestion and dispersal framework and library for apache hadoop | uber engineering blog. https://eng.uber.com/marmaray-hadoop-ingestion-open-source/. (Accessed on 12/05/2019).

[90] Hudi: Uber engineering's incremental processing framework on apache hadoop | uber engineering blog. https://eng.uber.com/hoodie/. (Accessed on 12/05/2019).

[91] Reza Shiftehfar and Stepan Bedratiuk. (4) uber : Big data infrastructure and machine learning platform - youtube. https://www.youtube.com/watch?v=y3O94MnO_IU&feature=youtu.be. (Accessed on 12/05/2019).

[92] Application samples from microsoft ai lab. https://www.microsoft.com/en-us/ai/ai-lab-application-samples?activetab=pivot1:primaryr8. (Accessed on 12/05/2019).

[93] Application samples from microsoft ai lab. https://www.microsoft.com/en-us/ai/ai-lab-application-samples?activetab=pivot1:primaryr7. (Accessed on 12/05/2019).

[94] Stories from microsoft ai lab. https://www.microsoft.com/en-us/ai/ai-lab-stories?activetab=pivot1:primaryr8. (Accessed on 12/05/2019).

[95] Meredith Ringel Morris. Ai and accessibility: A discussion of ethical considerations. *Communications of the ACM*, January 2020. (preprint of a "Viewpoint" column to appear in CACM in late 2019/early 2020).

[96] Ashwin Ram and Amazon Alexa. (4) ashwin ram, conversational ai in amazon alexa at the ai conference 2017 - youtube. https://www.youtube.com/watch?v=2Bazibaz1F8&t=1214s. (Accessed on 12/05/2019).

[97] Amazon sagemaker. https://aws.amazon.com/sagemaker/. (Accessed on 12/05/2019).

[98] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[99] Duy Ho H., Raj Marri, Sirisha Rella, and Yugyung Lee. Deeplite: Real-time deep learning framework for neighborhood analysis. *Stream-ML, IEEE Big Data*, 2019.

[100] Apache Kafka. apache kafka - google search. https://www.google.com/search?q=apache+kafka&oq=apache+kafka&aqs=chrome..69i57j0l2j69i65l3.1633j0j9&sourceid=chrome&ie=UTF-8. (Accessed on 12/13/2019).

[101] Zero MQ. Zeromq. https://zeromq.org/. (Accessed on 12/13/2019).

[102] Rabbit MQ. Messaging that just works — rabbitmq. https://www.rabbitmq.com/. (Accessed on 12/13/2019).

[103] Ganesh Srinivasan. @scale 2019: Kafka @scale: Confluent's journey bringing event streaming to the cloud – @scale. https://atscaleconference.com/videos/scale-2019-kafka-scale-confluents-journey-bringing-event-streaming-to-the-cloud/. (Accessed on 12/13/2019).

[104] Bingjing Zhang, Yang Ruan, and Judy Qiu. Harp: Collective communication on hadoop. In *2015 IEEE International Conference on Cloud Engineering*, pages 228–233. IEEE, 2015.

[105] Langshi Chen, Bo Peng, Bingjing Zhang, Tony Liu, Yiming Zou, Lei Jiang, Robert Henschel, Craig Stewart, Zhang Zhang, Emily Mccallum, et al. Benchmarking harp-daal: High performance hadoop on knl clusters. In *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, pages 82–89. IEEE, 2017.

[106] Intel® data analytics acceleration library | intel® software. https://software.intel.com/en-us/daal. (Accessed on 12/14/2019).

[107] Supun Kamburugamuve, Kannan Govindarajan, Pulasthi Wickramasinghe, Vibhatha Abeykoon, and Geoffrey Fox. Twister2: Design of a big data toolkit. *Concurrency and Computation: Practice and Experience*, page e5189, 2017.

[108] Supun Kamburugamuve, Pulasthi Wickramasinghe, Kannan Govindarajan, Ahmet Uyar, Gurhan Gunduz, Vibhatha Abeykoon, and Geoffrey Fox. Twister: Net-communication library for big data processing in hpc and cloud environments. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pages 383–391. IEEE, 2018.

[109] intel-analytics/bigdl: Bigdl: Distributed deep learning library for apache spark. https://github.com/intel-analytics/BigDL. (Accessed on 12/14/2019).

[110] Weka 3 - data mining with open source machine learning software in java. https://www.cs.waikato.ac.nz/ml/weka/. (Accessed on 12/14/2019).

[111] Moa – machine learning for data streams. https://moa.cms.waikato.ac.nz/. (Accessed on 12/14/2019).

[112] Apache samoa. https://samoa.incubator.apache.org/. (Accessed on 12/14/2019).

[113] H2o architecture — h2o 3.26.0.11 documentation. http://docs.h2o.ai/h2o/latest-stable/h2o-docs/architecture.html. (Accessed on 12/14/2019).

[114] Joseph E Gonzalez, Reynold S Xin, Ankur Dave, Daniel Crankshaw, Michael J Franklin, and Ion Stoica. Graphx: Graph processing in a distributed dataflow framework. In *11th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 14)*, pages 599–613, 2014.

[115] Jure Leskovec and Rok Sosič. Snap: A general-purpose network analysis and graph-mining library. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(1):1, 2016.

[116] Eric P Xing, Qirong Ho, Wei Dai, Jin Kyu Kim, Jinliang Wei, Seunghak Lee, Xun Zheng, Pengtao Xie, Abhimanu Kumar, and Yaoliang Yu. Petuum: A new platform for distributed machine learning on big data. *IEEE Transactions on Big Data*, 1(2):49–67, 2015.

[117] Deep graph library. https://www.dgl.ai/pages/about.html. (Accessed on 12/14/2019).

[118] mesham/epython: Python for the epiphany co-processor. https://github.com/mesham/epython. (Accessed on 12/13/2019).

[119] Enric Tejedor, Yolanda Becerra, Guillem Alomar, Anna Queralt, Rosa M Badia, Jordi Torres, Toni Cortes, and Jesús Labarta. Pycompss: Parallel computational workflows in python. *The International Journal of High Performance Computing Applications*, 31(1):66–82, 2017.

[120] Javier Álvarez Cid-Fuentes, Salvi Solà, Pol Álvarez, Alfred Castro-Ginard, and Rosa M. Badia. dislib: Large Scale High Performance Machine Learning in Python. In *Proceedings of the 15th International Conference on eScience*, pages 96–105, 2019.

[121] Yadu Babuji, Anna Woodard, Zhuozhao Li, Daniel S Katz, Ben Clifford, Rohan Kumar, Lukasz Lacinski, Ryan Chard, Justin M Wozniak, Ian Foster, et al. Parsl: Pervasive parallel programming in python. In *Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing*, pages 25–36. ACM, 2019.