Deep Learning Using Big Data

Edmundo Bernardo Paul Fernandes   
Masters in Data Analysis for Business  
CCT DublinWicklow, Ireland  
sbs23034@student.cct.ie

*Abstract*— Distributed computing and neural networks hold immense potential across diverse domains for tackling complex challenges.

In domains such as finance, they facilitate real-time trading and risk assessment. In Healthcare the benefits range from improved medical image analysis to drug discovery. In manufacturing, quality control and predictive maintenance are enhanced. Autonomous vehicles rely on them for object detection and path planning. Natural language processing enables accurate language translation and sentiment analysis. E-commerce gains personalized recommendations and fraud detection. Environmental sciences benefit from climate modelling and remote sensing.

Lastly, the energy sector employs them in managing smart grids and predicting energy demand.

The objectives involve leveraging distributed resources for scalable processing and utilizing neural networks for pattern recognition and prediction. The utilized methods include data distribution, model training, and real-time analysis for real time results. The findings consistently will demonstrate improved efficiency, accuracy, and scalability in specific tasks.

In conclusion the synergy between distributed computing and neural networks revolutionizes industries by addressing domain-specific challenges, paving the way for smarter and more efficient solutions.

# **Introduction**

In times where an exponential growth in data and an increasing demand for sophisticated problem- solving, the blend of distributed computing and Neural Networks has emerged as a transformative force across various domains. Distributed computing includes the power of multiple interconnected devices or servers to process and analyse large datasets, while Neural Networks, inspired by the intricate human brain neural networks, demonstrate remarkable aptitude at pattern recognition and complex decision- making. This synergy between distributed computing and neural networks holds great potential in focusing on domain-specific challenges and revolutionize the way we approach complex tasks.

The primary objective of this research is to explore and analyse the application of distributed computing and Neural Networks within specific domains. My aim is to investigate how these technologies are leveraged to enhance efficiency, accuracy, and decision-making in various fields. Our research question is as follows: How do distributed computing and Neural Networks contribute to solving complex problems and optimizing processes within specific domains, and what are the implications of their integration for future advancements?

In the subsequent sections, I will delve into concrete examples of how these technologies are applied in distinct domains, highlighting their methodologies, findings, and overarching conclusions. Through this exploration, I aim to provide valuable insights into the transformative potential of distributed computing and Neural Networks in addressing real-world challenges.

II **Literature Review**

Overview –

This section serves as an introductory roadmap, orienting the readers to the scope, objectives, and significance of the review while setting the context for the subsequent sections.

The blending of distributed computing and neural networks technology is drastically growing and emerging as a dynamic and transformative field with greater reach across various domains.

Performance is one of the key features of parallel and distributed computing systems. A vast research effort was invested in the development of approaches for performance modelling and prediction of parallel and distributed computing systems.

The act of reading and recognizing these words, sentences and paragraph is pattern recognition.

A machine could therefore be designed that could recognize the letter ‘a’ rather easily, since it would only recognize the one pattern.

But if we want to build a machine that can hand-written characters, the problem is much more difficult due to wide variation between examples of same letter. It is difficult to find a representative character for each of the letters and set a typical character for each letter to be constructed.

For example, all the different letter ‘a’s that are going to be read belong to the same set. Similarly, all the other letters belong to their own set, giving 26 sets in all.

Each set of characters is called a class, and we now must build a machine that chooses which class on input belongs to a pattern classifier.

The machine can solve this problem because the input patterns do not have to match one of the characters in the class exactly. Instead, the machine must be able to decide that the input pattern is mor like the members of one class than any of the others.

**Distributed Computing:** With growing scale of the data volume and neural network size, we have come into an era of distributed deep learning.

A distributed system is a collection of autonomous computing elements that appears to its users as a single coherent system.

High Performance training and inference on distributed computing systems has been attracting increasing research attention in both academia and industry.

Meanwhile, diversity of existing machine learning frameworks (e.g., TensorFlow, Pytorch, and MXNet) and the explosion of deep learning hardware’s, bring more challenges for users to leverage new deep learning technologies and accelerating capability of hardware devices.

Distributed computing covers all aspects of computing and information access across multiple processing elements connected by any form of communication network, whether local or wide area in the coverage. Together with management and parallel processing principle it allows to acquire and analyse intelligence from big data making big data analytics a reality. Different aspects of distributed computing paradigm resolve different types of challenges involved in analytics of big data.

Another term that is very closely related to distributed systems is parallel systems. While both refer to scaling up the computational capability, the way they achieve them is different. In parallel computing, we use multiple processors on a single machine to perform multiple tasks simultaneously, possibly with shared memory. However, in distributed computing, we use multiple autonomous machines with no shared memory and communicating with message passing.

Distributed computing is performed on a spatially distributed system, providing the user with a non-distributed view of a distributed system to implement a distributed file system that allows the programmer to ignore the physical location of his data. It refers to the paradigm of processing and analysing data across multiple interconnected computing nodes.

The adoption of distributed systems, such as clusters, grids, and cloud infrastructure has facilitated the processing of vast datasets enabling real-time analytics and complex computations.

**Artificial Neural Networks (ANN)**:

A diagram of a cell nucleus

Description automatically generated

Biological Neural Network

Neural networks have become very popular and helpful for classification, clustering, pattern recognition and prediction in many areas.

A neural network is a network of interconnected elements. These elements were inspired from studies of biological nervous systems.

In other words, neural networks are an attempt at creating machines that work in a similar way to the human brain by building these machines using components that behave like biological neurons representing a subset of machine learning algorithms.

A diagram of a network

Description automatically generated

Artificial Neural Network

Dendrites from biological neural network represent inputs in artificial neural networks, cell nucleus represent nodes, synapse represents weights, and axon represent output.

These artificial neural networks consist of interconnected nodes, or neurons, organized in layers, allowing them to learn intricate patterns and representations from data. ANN primarily consists of three layers.

A close-up of a network

Description automatically generated

Neural Network Layers

Input Layer accepts inputs in different formats provided by the programmer.

Hidden Layer presents in between input and output layers. It performs all the calculations to find hidden features and patterns.

Output Layer, the input goes through a series of transformations using the hidden layer, which finally results in output that is conveyed using this layer. The ANN takes input and computes the weighted sum of the inputs and includes a bias, and this computation is represented in the form of a transfer function.

While neural networks excel at learning complex patterns and representations, they often demand substantial computational resources which distributed computing platforms can readily provide.

Artificial neural networks full application can be evaluated with respect to data analysis factors such as accuracy, processing speed, latency, performance, fault tolerance, volume scalability and convergence.

**Specific Domains of research:**

The literature review aims at diving into the application of distributed computing and neural networks across several domains explaining in depth how these systems are revolutionising industries such as finance, autonomous vehicles, natural language processing, retail, manufacturing, agriculture, energy, banking, management, insurance, transportation, computer security, marketing, and those challenges that cannot be solved by the computational ability of traditional procedures and conventional mathematics.

**Scope of review:**

With this research, I will provide a comprehensive analysis of existing research, methodologies, and notable findings in the application of distributed computing and neural networks withing the previously domains.

By synthesizing current knowledge, I aim at identifying gaps, trends and future directions that can guide researchers, practitioners, and decision makers in harnessing the potential of these systems effectively.

Through a systematic peer reviewed literature, publications, case studies, I aim at presenting a holistic view of the state of the art in distributed computing and neural networks within specified domains.

**State of the Art in Distributed Computing and Neural Networks**

The concept of combining computing distribution and neural networks represents a powerful approach to meet the growing demands of AI applications in several domains. It addresses challenges related to the size of data, model complexity, and the need for efficient and scalable deep learning solutions.

In this section my objective is an in-depth examination of relevant research and findings in the field by contextualizing the current state of knowledge, identify key trends, and highlight significant contributions.

I will provide a taxonomy of artificial neural networks (ANNs) and furnish the reader with knowledge of current and emerging trends in ANN applications research and area of focus for researchers. Additionally, the study presents ANN application challenges, contributions, compare performances and critiques methods. This study will cover many applications of ANN techniques in various disciplines which include computing, science, engineering, medicine, environmental, agriculture, mining, technology, climate, business, arts, and nanotechnology, etc. The study assesses ANN contributions, compare performances and critiques methods.

The range of approaches for performance modelling and prediction that I will cover ranges from high level mathematical modelling to the detailed instruction level simulation.

I will now present some of the most relevant approaches for performance modelling and prediction of parallel and distributed computing systems.

The PAL approach is based on the Performance and Architecture Laboratory (PAL) which operates at the forefront of advanced system architectures and applications. PAL investigates the issues that contribute to optimal application and computer system performance for current extreme-scale systems and for future architectures.

The approach displays the time a machine takes to execute a program as a parameterized mathematical model. The parameters express the problem size and computational and communicational capabilities of the machine. The program execution time is as follows,

A close-up of a math formula

Description automatically generated

Where Tcomp is the computation time, TComm is the communication time, and TMemCont is the time spent for memory contention within a multiprocessor node.

By expressing the performance of the whole program with a mathematical expression the structural information of the program is not preserved. For this reason, this approach may not be suitable for the model-based performance evaluation of various program designs.

Performance Evaluating Virtual Parallel Machine (PEVPM) is a performance modelling system for messaging programs. Performance modelling provides hope for an escape from the measure modify development process using its predictive powers, which can advance the focus on performance optimization to earlier in the design process.

Performance Prophet (PP) is a tool for performance modelling and prediction of parallel and distributed computing systems. This tool provides a graphical user interface, which alleviates the problem of specification and modification of performance model. It works by the user specifying graphically the performance model using the Unified Modelling Language (UML) Performance Prophet then automatically transforms the performance model from UML to C++ and evaluates it by submission.

The Performance Analysis and Characterization (PACE) is a set of tools, techniques and practices used to assess and understand the performance characteristics of computer systems, software applications, or other technological systems. This environment is crucial to identify bottlenecks, optimizing resource utilization, and ensuring that systems meet performance requirements.

Performance Oriented End-to-end Modelling Systems (POEMS). The aim of this project was to develop an environment for performance modelling of parallel computing systems. POEMS proposed a methodology for the evaluation of system model using multiple evaluation tools designed to model, optimize, and monitor the performance of complex system processes. POEMS authors suggest that each component of the system model may be evaluated by the corresponding evaluation tool. The output of a tool serves as input for the subsequent tool.

Cluster Evaluator (CLUE) is an execution driven simulator that is used for performance evaluation of message passing programs on cluster computer architectures. Parallel Virtual Machine (PVM) is a software system that supports the message passing programming paradigm. CLUE is used for performance evaluating of message passing programs that are developed based on PVM.

Some fundamental concepts of Distributed Computing are parallelism, scalability,

and fault tolerance.

Scalability is the ability of a system to handle increasing workloads without degrading its performance as the system grows. Fault tolerance is the ability of a system to continue functioning despite failures or errors in some of its components.

Horizontal scaling (Scaling Out) involves adding more machines or nodes to the distributed system. This is often the preferred approach as it can be cost effective and provides scalability. Popular technologies like load balancers and container orchestration facilitate horizontal scaling.

Vertical Scaling (Scaling up) involves increasing resources (e.g., CPU, RAM) of individual machines within the distributed system. While vertical scaling has its merits, it may have limitations in terms of cost and upper bound scalability.

Load Balancing, load balancers distribute incoming workload, requests or tasks across multiple servers or nodes to ensure no single component overloads any single server. Load balancing algorithms can be simple (e.g., round-robin) or complex (e.g., based on server health or response time).

Data partitioning and sharding can be defined as distributing data across multiple servers or databases, known as partitioning or sharding, it can both improve storage and processing scalability. Each shard is responsible for a subset of the data, reducing the load on individual resources.

Caching frequently accessed data or results will significantly reduce the load on backend systems. Distributed catching mechanisms, like Redis or Memcached, can be employed to store and retrieve data quickly.

Asynchronous Processing, offloading time-consuming or non-blocking tasks to asynchronous processing frameworks (e.g., message queues like RabbitMQ or Apache Kafka) can improve system responsiveness and scalability.

Fault Tolerance refers to the ability of a system to continue to operate without interruption when one or more of its components fail. Hardware and software redundancy methods are the known techniques of fault tolerance.

A diagram of a fault tolerance

Description automatically generated

Fault Tolerance

In distributed systems there are 3 types of problems that occur. All 3 types are problem related.

Fault is defined as a weakness or shortcoming in the system or any hardware and software component. The presence of fault can lead to error and failure.

Errors are incorrect results due to the presence of faults and failure is the final outcome where the assigned goal is not achieved.

Big Data Integration, Distributed computing platforms like Apache Hadoop can seamlessly integrate with neural networks, enabling the processing of large-scale datasets for training and inference.

Monitoring and Performance tuning, regularly monitoring and distributed system’s performance and identifying problems or resources is crucial. Performance tuning may involve optimizing code, database, queries, or network configurations.

Using distributed databases (e.g., Cassandra, Hadoop) or distributed file systems can provide scalable storage solutions that can handle massive amounts of data.

Hu, Fei et al. worked on climateSpark, a framework for large scale climate data analysis. Big data analytics presents a unique set of problems for distributed computing due to the difficulty in effectively managing the massive amounts of data involved.

Because large amounts of data are distributed across multiple computing nodes, it can be challenging to manage and process this data effectively. Which can lead to data duplication, inconsistencies, and other issues that can negatively impact the accuracy and reliability of the analytics results.

While there are challenges to using distributed computing in the field of big data analytics, there are also many potential benefits. Advantages include parallel and distributed processing of huge data sets, as it may significantly increase the speed and efficiency of the analytics process and allow businesses to acquire insights from their data in real-time.

Brewer, E.A. said in his study that Big Data has a low amount of information per byte. So, given the large volume of data, the potential for great insight is relatively high if the entire dataset can be analysed. Overall, distributed computing for big data analytics presents both challenges and opportunities.

Despite facing data management and scalability challenges, distributed computing provides many benefits, including increased efficiency, speed, and reliability. As a result, it will likely continue to be a key area of research and development in big data analytics.

Artificial Neural Networks (ANN) can be best represented as weighted directed graph, where the artificial neurons form the nodes. The association between the neurons outputs and neuron inputs can be viewed as the directed edges with weights. The success of an artificial neural network highly depends on its architecture and among many approaches, Evolutionary Computation, which is a set of global search methods inspired by biological evolution which has been proven to be an efficient approach for optimizing neural networks structures. The choice of neural network architecture depends on the specific problem, type of data, and the desired performance.

I will now review some popular types of artificial neural networks:

Feedforward Neural Networks (FNNs), also known as multilayer perceptron’s (MLPs), consisting of an input layer, one or more hidden layers and an output layer and it is commonly used for regression and classification tasks.

Feedback Neural Networks, the output returns into the network to accomplish the best evolved results internally. As per the University of Massachusetts the feedback networks feed information back to itself and are well suited to solve optimization issues. The internal system error corrections utilize feedback Artificial Neural Networks.

Convolutional Neural Networks (CNNs) is a type of Deep Learning neural architecture commonly used in computer vision. It was designed for image processing and recognition tasks. It utilizes convolutional layers to capture spatial hierarchies of features, often including pooling layers to reduce spatial dimensions and are well known for their effectiveness in computer vision which is a field of Artificial Intelligence that enables a computer to understand and interpret the image or visual data.

Recurrent Neural Networks (RNNs) is a special type of artificial neural network adapted for time series data or data that involves sequences, such as time series, text, and speech. Contains recurrent connections that allow information to persist through time. RNNs have the concept of memory that helps them store information of previous inputs to generate the next output of the sequence. Variants like Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks improve handling of long-range dependencies.

Autoencoder is an unsupervised learning algorithm in which artificial neural networking is designed in a way to perform task of data encoding plus data decoding reconstruct input. It is commonly used in dimensionality reduction and anomaly detection.

Generative Adversarial Networks (GANs) it has two parts, The generator learns to generate plausible data. The generated instances become training examples for the discriminator.

The discriminator learns to distinguish the generator’s fake data from real data. The discriminator penalizes the generator for producing implausible results. Applications include image generation and style transfer.

Transformers introduced the context of natural language processing (NLP), it utilizes attention mechanisms to capture contextual information in sequences, and it is notable for the use of large pre trained models like BERT and GPT.

Crafting neural network architectures is of paramount importance for the progress of Deep Learning. There are countless new Neural Networks architectures proposed every single day.

Deep Learning revolution was catalysed by the development of deep neural networks. Deep networks learn from data in the same way that babies experience the world, starting with fresh eyes and gradually acquiring the skills needed to navigate novel environments. Learning algorithms extract information from raw data, information can be used to create knowledge, knowledge underlies understanding, understanding underlies wisdom. AI is on a trajectory measured in decades. The machine has gained the ability to recognize and process complex patterns of related information from every dimension just as the human brain, but to achieve human like intelligence a machine requires access to large volumes of data (Bengio, 2009). A system that could operate with some of the above notions, named perceptron (Rosenblatt, 1957) was suggested in Rosenblatt (1957). This perceptron went on to be the basis for creating multi-layer learning networks which have formed the basis of what is popularly called Deep Learning (Arel et al., Bengio, 2009).

Deep learning is an algorithm simulating animal neural networks structure to discover the relationship between data distributions. The current applications of Deep Learning include computer vision (CV), natural language processing (NLT), video / speech recognition(V/SP), finance, and banking(F&B).

In the early stages of Computer vision development, the deep learning approach faces difficult challenges related to computer memory, CPU, and GPU.

Many methods for CV have been proposed, such as K-Means, Naives Bayes classifier, Decision Tree, Random Forest, Haar Classifier, Expectation-Maximization (EM), K-Nearest Neighbor (KNN), and Support Vector Machine (SVM).

Deep Learning is the use of a lot of layers of neurons to progressively extract higher level features from the data that we feed to the neural network. DL has expanded exponentially in the last decade and it has been broadly separated into ten categories in terms of algorithm and architecture: Convolutional Neural Networks(CNNs), Long Short-Term Memory Networks (LSTMs), Recurrent Neural Neural Networks (RNNs), Generative Adversarial Networks (GANs), Radial Basis Function Networks (RBNFs), Multilayer Perceptrons (MLPs), Self-Organizing Maps (SOMs), Deep Belief Networks (DBNs), Restricted Boltzmann Machines (RBMs), and Autoencoders.

A diagram of machine learning

Description automatically generated Layers of ANN

Achieving the classification task using conventional ML techniques requires several sequential steps, specifically pre-processing, feature extraction, wise feature selection, learning, and classification.

Furthermore, feature selection has a great impact on the performance of ML techniques.

Biased feature selection may lead to incorrect discrimination between classes. Conversely, DL can automate the learning of feature sets for several tasks, unlike conventional ML methods.

DL enables learning and classification to be achieved in a single shot. DL has become an incredibly popular type of ML algorithm in recent years due to the huge growth and evolution of the field of big data.  Developing a model requires historical data from the domain that is used as training data.

This data is comprised of observations or examples from the domain with input elements that describe the conditions and an output element that capture what the observation means.

A problem where the output is a quantity would be described generally as a regression predictive modelling problem. Whereas a problem where the output is a label would be described generally as a classification predictive modelling problem.

A neural network model uses the examples to learn how to map specific sets of input variables to the output variable. It must do this in such a way that this is mapping works well for the training dataset, but also works well on new examples, called the ability of the model to generalize.

Despite these remarkable advancements, sentiment analysis still faces challenges. Fine-grained sentiment analysis, emotion detection, and handling sarcasm and irony in text remain an open research question. Additionally, the domain adaptation of pre trained models and mitigation of bias in sentiment analysis models and the mitigation of bias in sentiment analysis models are areas in need of further investigation.

While decisions made by rule-based software can be traced back to the last if and else, the same can’t be said about machine learning and deep learning algorithms. This lack of transparency in deep learning is what it’s called [the “black box” problem](https://bdtechtalks.com/2018/02/09/scary-ai-blackbox/).

Deep learning algorithms shift through millions of data points to find patterns and correlations that often go unnoticed to human experts. The decision they make based on these findings often confound even the engineers who created them.

This might not be a problem when deep learning is performing a trivial task where a wrong decision will cause little or no damage. But when it’s deciding the fate of a defendant in court or the medical treatment of a patient, mistakes can have more serious repercussions.

“The transparency issue, yet unsolved, is a potential liability when using deep learning for problem domains like financial trades or medical diagnosis, in which human users might like to understand how a given system made a given decision.

**Research Methodologies**

This report outlines the methods and techniques used in this study aimed at evaluating deep learning models using big datasets.

The research objective is to explain the potential of big data to enhance the performance of deep neural networks.

For this specific project I have applied two types of social research, theoretical and practical. For the theoretical approach I used an approach called basic research, where I gathered studies performed and explored and noted in academic papers. Academic research is the progenitor of virtually all scientific ideas and principles (Neuman, 2014) because it primarily focuses on the establishment and advancement of novel ideas, principle, and theories about phenomenon (National Science Foundation, as cited in Mansfield, 1980).

For my practical research approach, I used an approach called Applied research method which is basically the conduction of an inquiry to proffer solutions to certain issues and the results are targeted at influencing decisions.

For this approach I have used tools such as Jupyter notebook, Python, Hadoop and Pyspark.

Jupyter notebook is a web based interactive environment that combines code, rich text, images, videos, animations, mathematical equations, plots, maps, interactive figures, widgets, and graphical user interfaces into a single document. Jupyter notebook is easy to share, comes with a special tool, nbconvert, which converts notebooks to other formats such as HTML and PDF. Jupyter brings a lightweight interface for kernel languages that can be wrapped in Python.

Python is a suitable language for fast learning and real-world programming.

Python is a powerful high-level programming language created by a programmer named Guido Van Rossum, it supports multiple programming paradigms including imperative, object oriented, and functional programming. Python consists of a list of various useful libraries for data processing and integrated with other languages such as java as well as existing structures.

Python has various inbuilt features that support data processing regardless of the size. These features support processing for unstructured and unconventional data.

There are other technologies that can also process data more efficiently than python. They are Hadoop and Spark.

Hadoop is the best solution for storing and processing Big Data because it stores huge files in the form of HDFS, Hadoop distributed file system without specifying any schema.

Spark is as well a great choice for processing a large amount of structured and unstructured datasets as the data is stored in clusters.

Spark will conceive to store the maximum amount of data in memory so it can spill to disk.

Hadoop can be coded in Python language. We can explore programs like MapReduce in Python.

Spark provides a Python API called PySpark, which was released to support Python with Spark.

Spark comes with an interactive python shell called PySpark shell.

This PySpark shell is responsible for the link between the python API and the spark core and initializing the spark context. PySpark can also be launched directly from the command line by giving some instructions for interactive use.

 Python’s flexibility additionally permits to instrument Python code to form ML/AI scalability possibly without requiring higher expertise of distributed system and lots of invasive code changes. Hence, ML/AI users get the advantages of cluster-wide scalability with minimal effort.

HDFS cluster is based on the Hadoop Distributed File System (HDFS). Designed for use on commodity hardware, the storage system is scalable, fault tolerant, and rack aware.

Hadoop is a framework that permits the storage of large volumes of data on node systems. The Hadoop architecture allows parallel processing of data using several components:

Hadoop HDFS to store data across slave machines, Hadoop Yarn for resource management in the Hadoop Cluster, Hadoop MapReduce to process data in a distributed fashion, Zookeeper to ensure synchronization across a cluster.

HDFS has a Master-slave architecture. The daemon called NameNode runs on the master server. It is responsible for Namespace, management and regulates file access by client.

NameNode is the master server. In a non-high availability cluster, there can only be one NameNode. In a high availability cluster, there is a possibility of two NameNodes, and if there are two NameNodes there is no need for a secondary NameNode.

DataNode daemon runs on slave nodes, and it is responsible for storing data, it also creates, deletes and replicates blocks on demand from NameNode.

A block is nothing but the smallest unit of storage on a computer system.

MapReduce is a software framework that allows to write applications for processing a large amount of data.

MapReduce runs these applications in parallel on a cluster of low-end machines. It does so in a reliable and fault tolerant manner.

MapReduce job comprises several maps tasks and reduces tasks. Each task works on a part of data. This distributes the load across the cluster. The function of Map tasks is to load, parse, transform and filter data. Each reduce task works on the sub-set of output from the map tasks. Reduce task applies grouping and aggregation to this intermediate data from the map tasks.

The input file for the MapReduce job exists on HDFS. The input format decides how to split the input file into input splits. Input split is nothing but a byte-oriented view of the chunk of the input file. This input split gets loaded by the map task. The map task runs on the node where the relevant data is present. The data need not move over the network and get processed locally.

The fundamental idea of YARN is to split up the two major responsibilities of the MapReduce - JobTracker i.e., resource management and job scheduling/monitoring, into separate daemons: a global ResourceManager and per-application ApplicationMaster (AM).

The ResourceManager and per-node slave, the NodeManager (NM), form the new, and generic, system for managing applications in a distributed manner.

The main components of MapReduce are as described below:

ResourceManageris the ultimate authority that arbitrates resources among all the applica­tions in the system. The per-application ApplicationMaster is, in effect, a framework specific entity and is tasked with negotiating resources from the ResourceManager and working with the NodeManager(s) to execute and monitor the component tasks.

NodeManager is YARN’s per-node agent and takes care of the individual compute nodes in a Hadoop cluster. This includes keeping up-to date with the ResourceManager (RM), over­seeing containers’ life-cycle management; monitoring resource usage (memory, CPU) of individual containers, tracking node-health, log’s management and auxiliary services which may be exploited by different YARN applications.

JobHistoryServer is a daemon that serves historical information about completed applications.

MapReduce Key Features

Accessibility:

Supports a wide range of languages for developers, including C++, Java, or Python, as well as high-level language through Apache Hive and Apache Pig.

Flexibility:

Process all data, regardless of type or format — whether structured, semi-structured, or unstructured. Original data remains available even after batch processing for further analytics, all in the same platform.

Reliability:

Built-in job and task trackers allow processes to fail and restart without affecting other processes or workloads. Additional scheduling allows you to prioritize processes based on needs such as SLAs.

Hadoop Scalable:

MapReduce is designed to match the massive scale of HDFS and Hadoop, so you can process unlimited amounts of data, fast, all within the same platform where it’s stored.

While MapReduce continues to be a popular batch-processing tool, Apache Spark’s flexibility and in-memory performance make it a much more powerful batch execution engine. Cloudera has been working with the community to bring the frameworks currently running on MapReduce onto Spark for faster, more robust processing.

MapReduce is designed to process unlimited amounts of data of any type that’s stored in HDFS by dividing workloads into multiple tasks across servers that are run in parallel.

**Conclusion**

This project has been extremely challenging, due to problems with memory, my Ubuntu system, keeps on freezing due to low memory capacity, losing all progress until that point.

I have faced challenges such as reading my csv file due to no DateNodes in my distributed system.

I have spent days trying to understand and changing my Hadoop code as to why could I not read the file.

Mohammed has assisted in me in reading the file through PySpark due to time restrictions.

I believe my work, lacks certain depth, which I believe I could have explored but due to time restrictions, I am not able to explore Neural Networks or Hadoop any further.

This paper has strongly explored Neural Networks, Distributed Computing, Python, Hadoop and PySpark stating its benefits while dealing with Big Data.

Having a holistic approach, we can see that the evolution of Artificial Intelligence is consistently creating systems to support and evolve where previous systems lack.

There can be no doubt that through this research we can recognise the power of the spoken systems to analyse, explore, simply and explain Big Data.

**References**

Abiodun, O.I., Jantan, A., Omolara, A.E., Dada, K.V., Mohamed, N.A. and Arshad, H. (2018). State-of-the-art in artificial neural network applications: A survey. Heliyon, 4(11), p.e00938. doi:https://doi.org/10.1016/j.heliyon.2018.e00938.

Alzubaidi, L., Zhang, J., Humaidi, A.J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M.A., Al-Amidie, M. and Farhan, L. (2021). Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. Journal of Big Data, [online] 8(1). doi:https://doi.org/10.1186/s40537-021-00444-8.

Amazon Web Services, Inc. (n.d.). What is Distributed Computing? - Distributed Computing - AWS. [online] Available at: <https://aws.amazon.com/what-is/distributed-computing/>.

Analytics Vidhya. (2021). Integration of Python with Hadoop and Spark. [online] Available at: https://www.analyticsvidhya.com/blog/2021/05/integration-of-python-with-hadoop-and-spark/.

Chai, J., Zeng, H., Li, A. and Ngai, E.W.T. (2021). Deep learning in computer vision: A critical review of emerging techniques and application scenarios. Machine Learning with Applications, p.100134. doi:https://doi.org/10.1016/j.mlwa.2021.100134

Clark, T.H. (2021). 7 Essential Python Skills for Research. [online] Medium. Available at: https://towardsdatascience.com/7-essential-python-skills-for-research-496e1888e7c2 [Accessed 27 Sep. 2023].

Ch, K. and rakant (2020). Fundamentals of Distributed Systems | Baeldung on Computer Science. [online] www.baeldung.com. Available at: <https://www.baeldung.com/cs/distributed-systems-guide>.

Faik, L. (2022). Graph Neural Networks: Merging Deep Learning With Graphs (Part I). [online] blog.dataiku.com. Available at: https://blog.dataiku.com/graph-neural-networks-merging-deep-learning-with-graphs [Accessed 10 Sep. 2023].

GeeksforGeeks. (2018). Association Rule. [online] Available at: <https://www.geeksforgeeks.org/association-rule/>

GeeksforGeeks. (2020). Cosine Similarity. [online] Available at: <https://www.geeksforgeeks.org/cosine-similarity/>.

GeeksforGeeks. (2017). Dynamic Programming - GeeksforGeeks. [online] Available at: <https://www.geeksforgeeks.org/dynamic-programming/>.

GeeksforGeeks. (2023). Distributed Computing System Models. [online] Available at: https://www.geeksforgeeks.org/distributed-computing-system-models/ [Accessed 10 Sep. 2023].

GeeksforGeeks. (2017). Introduction to Convolution Neural Network. [online] Available at: <https://www.geeksforgeeks.org/introduction-convolution-neural-network/>

Google Developers. (2019). Overview of GAN Structure  |  Generative Adversarial Networks. [online] Available at: <https://developers.google.com/machine-learning/gan/gan_structure>.

Jain, A. (2020). What is Auto-Encoder in Deep Learning? [online] Analytics Vidhya. Available at: https://medium.com/analytics-vidhya/what-is-auto-encoder-in-deep-learning-5d668f94651b [Accessed 10 Sep. 2023].

Kumaraswamy, A. (2018). 10 reasons why data scientists love Jupyter notebooks. [online] Packt Hub. Available at: https://hub.packtpub.com/10-reasons-data-scientists-love-jupyter-notebooks/.

Picton, P. (1994). What is a Neural Network? Introduction to Neural Networks, pp.1–12. doi:https://doi.org/10.1007/978-1-349-13530-1\_1.

Saxena, D., Kumar, J., Ashutosh Kumar Singh and Schmid, S. (2023). Performance Analysis of Machine Learning Centered Workload Prediction Models for Cloud. 34(4), pp.1313–1330. doi:https://doi.org/10.1109/tpds.2023.3240567.

Simplilearn.com. (n.d.). Hadoop Architecture in Big Data Explained: A Complete Guide with Its Components. [online] Available at: <https://www.simplilearn.com/tutorials/hadoop-tutorial/hadoop-architecture>.

Tomar, A. (2019). ITEM-ITEM Collaborative filtering Recommender System in Python. [online] Medium. Available at: <https://medium.com/@tomar.ankur287/item-item-collaborative-filtering-recommender-system-in-python-cf3c945fae1e>

Ünal, H.T. and Başçiftçi, F. (2021). Evolutionary design of neural network architectures: a review of three decades of research. Artificial Intelligence Review. doi:https://doi.org/10.1007/s10462-021-10049-5.

Wason, R. (2018). Deep learning: Evolution and expansion. Cognitive Systems Research, 52, pp.701–708. doi:https://doi.org/10.1016/j.cogsys.2018.08.023.

www.javatpoint.com. (n.d.). Artificial Neural Network Tutorial - Javatpoint. [online] Available at: <https://www.javatpoint.com/artificial-neural-network>.