Deep Learning using Big Data

Claire O’Brien   
Data Analytics  
College of Computing and TechnologyDublin, Ireland  
sbs24031@student.cct.ie

*Abstract*—This work critically analysed the literature regarding how Deep Learning algorithms can be combined with Big Data technologies to produce accurate and precise results. Two models were built, one to demonstrate the benefits of employing the big data framework Apache Spark using the API Pyspark and the second to show the importance of choosing the optimal parameters for an ANN.

*GitHubLink: https://github.com/claireobrien00/Semester2CA1.git*

Keywords—Deep Learning, Big Data, Backpropagation, Hadoop, Apache Spark

# Introduction

The chosen topic of this research is the use of Deep Learning (DL) with Big Data (BD). These are two prominent aspects of the field of data analytics. Combining the two technologies unlocks the potential to predict the outcome and identify patterns of large, complex datasets that were previously beyond the scope of traditional predictive data analytics.

This work’s objective is to introduce both concepts, the expanding need for them in modern data analytics, their benefits, and drawbacks, investigate how they can be used in collaboration and provide a coded example of them operating in unison.

A diagram of a complex system

Description automatically generated

*Figure 1: Architecture of a standard Deep Learning Neural Network (LeCun et al., 2015)*

Deep Learning refers to hierarchical algorithms which automatically identify patterns, using unsupervised and supervised strategies, between multiple features of complex datasets. DL algorithms comprise an input layer, multiple hidden layers of non-linear transformations depending on weights, biases, and activation functions and an output layer. Figure 1 shows a visualization of this, with each circle indicating a neuron in the neural network. The algorithms output binary/multivariate classification and regression results, which have been applied to wide-ranging applications such as image detection, speech recognition and language processing (Jan et al., 2019; Najafabadi et al., 2015; Zhang et al., 2018). Deep Neural Networks’ signature is the concept of backpropagation (Chae et al., 2018). This key characteristic creates a feed forward loop in which the weights and biases are updated to improve the result with each iteration. The numerous hidden layers within the DL algorithm are trained using calculus, namely stochastic gradient descent (LeCun et al., 2015). The aim is to find the parameters where the loss is at a minimum, which is identified by calculating the gradient descent iteratively until it converges to a fixed value (Alsheikh et al., 2016).

The second focus of this work is Big Data. Big data has become prevalent due to the abundance of data, particularly the rate, being generated.

A diagram of a management system

Description automatically generated with medium confidence

*Figure 2: Function of Big Data Frameworks (Gandomi and Haider, 2015)*

The 5 V’s (Velocity, Volume, Veracity, Variety and Value) describe the characteristics of a big data set.

It demands more robust storage and analysis systems compared to conventional data (Chen and Lin, 2014). BD frameworks must be designed to store and refine the raw data, along with the processes listed in Figure 2. There are numerous frameworks and technologies currently available in the field. Hadoop MapReduce, Apache Spark and Cassandra are just a few of these. They have differing specific features but broadly speaking they each distribute the work across multiple computers and allow processes to be run in parallel. This reduces the time and improves the reliability (Zakir and Seymour, 2015).

This work aims to address the question: how do deep learning algorithms perform using big data frameworks on large datasets and what effect does the architecture of the DL algorithms have on results? How can the architectures be fine-tuned to optimize the results?

# Literature Review

A comprehensive review was performed on the published literature. The current state of the art was studied which informed the practical sections.

Chae et al. used mandatorily reported infectious disease data collected in South Korea as the basis of their predictive model. They collected data from various sources, one of which was Twitter, and it was classified as big data. A limitation of this study was BD framework used was not specified. Two DL algorithms were applied and compared to ARIMA, a common non-stationary time series ML algorithm. A lag parameter was introduced due to the serial nature of the prediction model. This parameter determined how many days after the data was collected, that data was used in the prediction of the disease levels. This parameter was optimized for each disease. A DNN and an LSTM model, a type of RNN were set up using the keras library. An RNN builds on the feed-forward characteristic of ANNs by using the previous and current iteration values to update the weights and biases. In this way, there is internal memory built into each neuron. The optimisers, activation functions and epochs were varied with all other parameters remaining constant. The RMSE was used to compare the various models. The optimized DNN models performed 24% better, followed by the LSTM, 18% better, compared to the ARIMA baseline. They outlined the limitations of the study as the small data collection window, the lack of applicability to other regions and relatively few DL parameters taken into consideration (Chae et al., 2018).

Mohapatra et al. published work detailing the need for big data storage and processing in the healthcare sector. The researchers focused on cardiac ECG data. Abnormalities are difficult to detect with the human eye as deviation from the normal range is tiny, hence an automatic detection system would be very beneficial. A cloud-based system was proposed with Hadoop MapReduce as the data manager. In the physical layer, an HDFS was employed along with an HBase repository. The MapReduce functions are applied to the characteristic ECG signals. This proved a valuable step as the map functions took the health indicators and calculated the weights, followed by the Reduce functions filtering out unneeded data and identifying the abnormal cardiac rhythms. The MapReduce function was applied twice to maximise its benefits, as shown in Figure 3. This extracted dataset was then used to train a classification ANN – does a patient with a given set of health indicators have heart disease? The process is illustrated in Figure 4.

A diagram of a diagram

Description automatically generated

*Figure 3: The two-step MapReduce function being applied to the health indicator data*

A diagram of a model

Description automatically generated

*Figure 4: The overall process of classifying if a patient has heart disease (Mohapatra et al., 2023)*

A notable addition to this model was particle swarm optimization (PSO). This is a streamlined algorithm to find the global minimum of the loss function thus extracting the best weights in as few iterations as possible. It is applied to human or animal datasets as it is routed in behavioral patterns. The author notes the benefit of the parallel MapReduce technique for training the ANN. A simulation was run with open-source data and the ANN had an accuracy of 99% when all attributes were used. This model’s accuracy decreased exponentially as the number of attributes reduced, showing ANN’s need for complex patterns within a dataset. The CPUs scaled up well as the flow of data increased, accredited to the Hadoop framework in use. The authors note the importance of normalization, collecting data from a diverse range of patients and the nature of the model i.e. ANN are well suited to classification but not temporal problems. This study used a static dataset as opposed to streamed data. A limitation of this study is that batch data was used. This is computationally less expensive than streaming data. An adaptation could be to implement an RNN. These are better suited to temporal data and produce better results, allowing the researchers to classify patients in real-time (Mohapatra et al., 2023).

Big Data often comes in an unstructured/unlabeled form which DL algorithms extract a hierarchy from and that is used to train a shallower (less deep, simpler) model. Najafabadi outlined some general applications of DL within the BD space.

Semantic Indexing techniques can be applied to BD which finds high-level, abstract links between the data. The data is stored more efficiently using these links. When the volume of data is vast this allows faster extraction of such data compared to when it is stored in its raw data bit string form. This is widely applied in document storage databases, documents’ texts can be scanned and sorted based on topic. Google’s word2vec DL algorithm is an example of this.

An extension of this would be DL algorithms employed alongside simpler linear models when analysing BD to maximise computational efficiency.

To develop this area, the limitations must be considered. It remains unclear how much data is enough to produce strong results (Najafabadi et al., 2015). DB datasets can be in the range of petabytes (Zakir and Seymour, 2015). The current standard is to utilise the entire dataset to train the model. This results in a huge volume of weights and biases for each neuron. The computational efficiency could be hugely improved if research was done to determine how much of a dataset needs to be included in training a model whilst avoiding underfitting. Domain adaption of DL within BD is discussed, whereby the distribution between the training and test data differs. Finally choosing the best performance metric of DL algorithms in BD is ambiguous. This is due to the unsupervised characteristic of many BD sets. It is not as simple as using the amount of misclassified results or the error between the predicted and actual values. Semi-labelled data can use the labelled data results to infer the results of the unlabelled data, but as of yet unsupervised algorithms lack a robust criterion (Najafabadi et al., 2015)

Kim et al. carried out work to combine Apache Spark and the TensorFlow library. So-called Deepspark aims to integrate DL algorithms and BD frameworks. It allows DL learning to be performed across multiple clusters, parameters are automatically divided using Spark. The clusters work iteratively using stochastic gradient descent, to optimize the parameters and minimize the loss. Synchronised SGD waits for the parameter values from each node to reach the barrier, the master node gathers and averages them before backpropagating them through the network. Kim noted the limitation of this system by the slowest node, so the team investigated asynchronous GSD. This utilized master-slave node architecture with a server node monitoring the parameters globally. DeepSpark is composed of:

* Apache Spark
* Parameter exchanger
* Computing Engine

This architecture was found to require more epochs but less computing time overall to achieve the same loss as the benchmark. This demonstrated the benefits to DL of using a distributed framework (Kim et al., 2016).

Another example of Spark being used to speed up the training of NN is outlined by Moritz et al., namely SparkNet. It used Caffe on multiple parallel CPUs. They found the performance depended on the cluster size, communication frequency and the cluster’s communication overhead. They found their novel setup was faster compared to a single node (Moritz et al., 2016)

Versaci et al. propose an image classification algorithm using a Cassandra database solution. Cassandra DB is a highly scalable, distributed framework for handling large datasets in commercial and academic settings. The architecture for the reading in and pre-processing can be seen in Figure 5. It was noted that both the data loader and server used Cassandra DB. The updated parameters calculated during backpropagation are gathered from each local mini-batch and consolidated into overall global weights. A mirrored strategy was adopted, whereby the consolidated parameters were kept on each of the working servers. This currently has not been applied to TensorFlow however, it is a promising step in the right direction for combining DL and BD (Versaci and Busonera, 2021).

Singh et al. commented on the poor performance of MapReduce with iterative ML algorithms such as K-Means clustering. This is one of the reasons why MapReduce was not explored in the coded example of this work as ANNs are also iterative (Singh and Reddy, 2015). Zakir et al. mirror this sentiment and praise Apache Sparks’ ability to handle iterative algorithms better (Zakir and Seymour, 2015)

LeChun et al. commented on the likely move towards unsupervised deep learning in the coming years. Unsupervised data makes up much of the big data available so these two fields need to work together to create models using this form of data. They also make note of the applications in reinforcement learning which will likely be explored (LeCun et al., 2015)

A diagram of a data processing process

Description automatically generated

*Figure 5: The Cassandra DB workflow proposed by Versaci et al. (Versaci and Busonera, 2021)*

An additional level of complexity to be added to an algorithm is a dataset containing time series data. Lv et al. published preliminary work on collecting real-time traffic data and using it to predict the future state of traffic on the same roads. Stacked Autoencoders were used, where the output becomes the input in the next loop. The researchers noted it performed well in medium and heavy traffic but poorly in light traffic (Lv et al., 2014). DeepTFP, another traffic predictor, predicted the traffic much more accurately than ARIMA when comparing the RSME (Chen et al., 2017).

Another source of BD to consider is the data pertaining to the Internet of Things, IoT. IoT systems produce a huge amount of sensor data and streaming data that are collected at regular intervals giving the datasets a characteristic serial element. The review commented on the various DL frameworks, such as TensorFlow and Torch, that work well with IoT-style data and can incorporate Hadoop file storage and management. They note the suitability of RNNs, specifically LSTM, with serial data and how they can provide better models compared to ANNs. The authors observed the lack of technology to handle real-time data, for which they propose using the Chainer framework. Chainer is designed to be updated dynamically and is optimized in a defined-by-run manner. The noisy, unlabeled and varying formats of IoT data were also highlighted as a potential challenge for the clusters of computers used in BD. The authors made note of the lack of focus of DL algorithms on regression type models and encouraged more work to be carried out in this area (Mohammadi et al., 2018).

Mobile Big Data, data produced by mobile devices, is another dataset type which produces data with the 5 V characteristics. Alsheikh et al. used MapReduce across multiple Spark nodes to run DL algorithms. They noted the beneficial scalability of Spark. The architecture had many workers and one master Spark which read in all the workers’ outputs and combined them to deliver the result. It was concluded that the BD platform, Apache Spark, deals well with the 5 V’s of MBD as well as the DL tools identified the complex patterns within the MBD accurately. The authors noted that the data’s veracity property required subject matter expert input to validate. They trained a model to identify activity recognition errors. Their model performed the best when compared to other ML techniques (Alsheikh et al., 2016).

# Methodology

To further investigate this area data was acquired from a reputable online source, namely the UC Irvine machine learning repository. The datasets are controlled using the Creative Commons licencing agreement. This gives consent for public use provided that the site is appropriately referenced (“Legal Code - Attribution 4.0 International - Creative Commons,” n.d.) .

Multiple datasets were explored and two were chosen, one for regression, the concrete dataset (I-Cheng Yeh, 1998), and one for classification, the banking dataset (S. Moro, 2014).

## Advanced Data Analytics

The concrete dataset was composed of physical features of concrete, with the target variable being the compressive strength.

The bulk of the Neural Network analysis was performed on this dataset. EDA was performed. The features were all numeric, so encoding was not required. No missing values were present. The spread of the features was checked. They were a mixture of approximately normal and skewed distributions. Boxplots were created. Some features had outliers but the data was mostly uniform. The dependencies between the features were plotted using a pair plot.

Standard and MinMax scaling was tried. Better results were achieved with MinMax scaling.

To combine BD and DL technologies the dataset was added to user1 using Hadoop. Apache Spark was chosen as the BD platform due to its successful implementation with ANN outlined in the review. A Pyspark notebook was opened and the CSV file was loaded in using spark.read.load. An Artificial Neural Network was constructed to predict the compressive strength of the concrete.

An input layer, hidden layer and output layer were added. The parameters were tuned to find the combination which gave the lowest loss. Due to this being a regression problem no activation function was applied to the output neuron. A single output neuron was chosen as the model was producing a single numeric predicted output. The following parameters were tuned:

* Number of Neurons
* Activation Function
* Loss Function
* Number of Epochs
* Optimizer
* Batch Size

Multiple values were chosen for each hyperparameter and the value was varied while the others remained constant. The minimum loss for each was noted and the minimums for each were used in the final ANN.

## Big Data

The banking dataset was considered to demonstrate the benefit of using Pyspark on a large dataset. This had 45,000 rows and took up 4.6 MB of storage, compared to 58 KB for the 1,000 rows in the concrete dataset. This dataset’s purpose was to demonstrate how effective the Apache Spark platform is at dealing with larger datasets.

The dataset was explored, scaled and encoded using a standard pandas data frame for ease and then saved to the Hadoop user1. The CSV file was added to the notebook using the spark.read.load

A basic NN was compiled and run on a Pyspark and Jupyter notebook. The task manager was opened and the performance of the Jupyter notebook browser and the VM was compared.

A screenshot of a computer

Description automatically generated

*Figure 6: Computer Performance without Apache Spark platform*

A screenshot of a computer

Description automatically generated

*Figure 7: Computer Performance with Apache Spark platform*

# Results and Discussion

The aim of the ANN built using the concrete dataset was to predict the compressive strength of a concrete sample based on 8 input features.

An RNN was tried on this dataset, but they are better suited to time series data and so were removed from the analysis.

One parameter which was not explored in this work was dropout. Dropout is used to reduce overfitting. As this data was relatively unnoisy, indicated by the boxplots, overfitting was not a likely problem.

The combination of parameters which provided the lowest loss is shown in Table 1.

|  |  |
| --- | --- |
| Number of Neurons in Hidden Layer | 500 |
| Activation Function | Relu |
| Loss Function | Mean\_squared\_logarithmic\_error |
| Number of Epochs | 500 |
| Optimizer | Adagrad |
| Batch Size | 10 |

*Table 1: The optimal hyperparameters for the ANN*

The activation and loss functions had the greatest impact on reducing the loss. Choosing these two functions carefully is crucial for an optimally performing NN. The architecture of the final ANN can be seen in Figure 8.

Scaling the data also had a big impact. Initially, standard scaling was tried but due to the approximate normal distribution of the features min-max scaling was tried and found to perform better.

Looking at the Chrome in Figure 6 and the VM in Figure 7 the memory usage is 97.5 MB vs 41.4 MB. This highlights the improvement of a distributed computing system like Apache Spark. It’s simple to implement in Python thanks to the Pyspark API. Due to time constraints, this DL algorithm was not explored past this point. There was an issue with an exploding gradient. A clipping parameter was added to the optimizer, but the issue could not be resolved. An improvement to this section would be to perform the EDA using Pyspark. This removes the need to save the data again once it has been scaled and encoded. There were issues relating to the loss function of the bank dataset ANN.

A black and white diagram with white text

Description automatically generated

*Figure 8: Architecture of the ANN*

This may have been due to the encoding of the data. Label encoding was used in place of one hot for simplicity.

# Conclusion

This work investigated how Deep Learning Neural Networks can be used with Big Data platforms to analyse large-scale data. A thorough literature review was conducted in which the state-of-the-art research was considered, the gaps and limitations explained and the applications of these technologies in the real world.

Two datasets were explored, one regression and one classification. The datasets were visualised, scaled and encoded where necessary. Apache Spark was chosen as the Big Data framework and demonstrated using an Ubuntu Virtual Machine. The computer’s performance was compared using task manager, and it was found that when using Apache Spark, the laptop used less than half of the memory allocation compared to just running the algorithm on its CPU. This illustrated the power of Spark’s clustering framework. The regression ANN’s parameters and architecture were refined to find the optimal combination and minimize the loss. The parameters had a major impact on the ANN’s result. This highlighted the criticality of refining a model for a specific problem.

This is an exciting field of research with wide-ranging applicability, such as IoT devices, healthcare settings and mobile data.

# References

Alsheikh, M.A., Niyato, D., Lin, S., Tan, H.-P., Han, Z., 2016. Mobile Big Data Analytics Using Deep Learning and Apache Spark.

Chae, S., Kwon, S., Lee, D., 2018. Predicting Infectious Disease Using Deep Learning and Big Data. IJERPH 15, 1596. https://doi.org/10.3390/ijerph15081596

Chen, X.-W., Lin, X., 2014. Big Data Deep Learning: Challenges and Perspectives. IEEE Access 2, 514–525. https://doi.org/10.1109/ACCESS.2014.2325029

Chen, Y., Shu, L., Wang, L., 2017. Poster abstract: Traffic flow prediction with big data: A deep learning based time series model, in: 2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). Presented at the 2017 IEEE Conference on Computer Communications: Workshops (INFOCOM WKSHPS), IEEE, Atlanta, GA, pp. 1010–1011. https://doi.org/10.1109/INFCOMW.2017.8116535

Gandomi, A., Haider, M., 2015. Beyond the hype: Big data concepts, methods, and analytics. International Journal of Information Management 35, 137–144. https://doi.org/10.1016/j.ijinfomgt.2014.10.007

I-Cheng Yeh, 1998. Concrete Compressive Strength. https://doi.org/10.24432/C5PK67

Jan, B., Farman, H., Khan, M., Imran, M., Islam, I.U., Ahmad, A., Ali, S., Jeon, G., 2019. Deep learning in big data Analytics: A comparative study. Computers & Electrical Engineering 75, 275–287. https://doi.org/10.1016/j.compeleceng.2017.12.009

Kim, H., Park, J., Jang, J., Yoon, S., 2016. DeepSpark: A Spark-Based Distributed Deep Learning Framework for Commodity Clusters.

LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature 521, 436–444. https://doi.org/10.1038/nature14539

Legal Code - Attribution 4.0 International - Creative Commons [WWW Document], n.d. URL https://creativecommons.org/licenses/by/4.0/legalcode (accessed 9.23.24).

Lv, Y., Duan, Y., Kang, W., Li, Z., Wang, F.-Y., 2014. Traffic Flow Prediction With Big Data: A Deep Learning Approach. IEEE Trans. Intell. Transport. Syst. 1–9. https://doi.org/10.1109/TITS.2014.2345663

Mohammadi, M., Al-Fuqaha, A., Sorour, S., Guizani, M., 2018. Deep Learning for IoT Big Data and Streaming Analytics: A Survey.

Mohapatra, S., Sahoo, P.K., Mohapatra, S.K., 2023. Healthcare Big Data Analysis with Artificial Neural Network for Cardiac Disease Prediction. Electronics 13, 163. https://doi.org/10.3390/electronics13010163

Moritz, P., Nishihara, R., Stoica, I., Jordan, M.I., 2016. SparkNet: Training Deep Networks in Spark.

Najafabadi, M.M., Villanustre, F., Khoshgoftaar, T.M., Seliya, N., Wald, R., Muharemagic, E., 2015. Deep learning applications and challenges in big data analytics. Journal of Big Data 2, 1. https://doi.org/10.1186/s40537-014-0007-7

S. Moro, P.R., 2014. Bank Marketing. https://doi.org/10.24432/C5K306

Singh, D., Reddy, C.K., 2015. A survey on platforms for big data analytics. Journal of Big Data 2, 8. https://doi.org/10.1186/s40537-014-0008-6

Versaci, F., Busonera, G., 2021. Scaling deep learning data management with Cassandra DB, in: 2021 IEEE International Conference on Big Data (Big Data). Presented at the 2021 IEEE International Conference on Big Data (Big Data), IEEE, Orlando, FL, USA, pp. 5301–5310. https://doi.org/10.1109/BigData52589.2021.9672005

Zakir, J., Seymour, T., 2015. BIG DATA ANALYTICS. IIS. https://doi.org/10.48009/2\_iis\_2015\_81-90

Zhang, Q., Yang, L.T., Chen, Z., Li, P., 2018. A survey on deep learning for big data. Information Fusion 42, 146–157. https://doi.org/10.1016/j.inffus.2017.10.006

**.**