Storage Solutions and Data Analytics: RDBM,

Hadoop and APIs in Neural Networks Contexts

Jose Maria Rico Leal  
*Master of Science in Data Analytics*  
sba23021@student.cct.ie

*Abstract*

*This study investigates the relationship between storage solution tools such as Relational Database Management Systems (RDBMS), Hadoop, and APIs, and how they can be separately interlinked with advanced data analytics, specifically neural networks. The purpose of the study is to compare an RDBMS with Hadoop when processing a 1.6GB dataset, and then apply a Neural Network. To expand the scope, this study will also include the usage of APIs (specifically, the Keras library) to implement a Convolutional Neural Network (CNN). This study was conducted using my personal laptop to load a 1.6GB dataset into an RDBMS and Hadoop. I utilized Jupyter Notebooks to interact with these two technologies, exploring computing times, roadblocks faced, and other insights. Following this, we applied the same Neural Network to predict if certain jobs are more popular based on gender. Another aspect of the study involves utilizing an API; for this purpose, Keras and a Convolutional Neural Network (CNN) were used. One of the objectives is to evaluate the performance of the CNN model in classifying movie reviews as positive or negative based on their sentiment. The research findings indicate that HDFS is a better choice for storing and processing large datasets compared to RDBMS; however, in certain scenarios and for non-technical users, an API like Keras can speed up the preprocessing of the data to model a NN.*

Keywords: Relational Database Management System, Hadoop, HDFS, Spark, API, Keras, Neural Network (NN), Convolutional Neural Network (CNN)

# Introduction

Relational Database Management Systems have been well-established since the late 1970s; at that time, the concept of Big Data was not the same as it is today. As technology rapidly advanced, the industry needed to process large amounts of data. To address this need, an open-source framework for writing and running distributed applications, called Hadoop, entered the scene (Lam, 2011). These two technologies, RDMS and Hadoop, are great; however, the implementation of both requires a high level of technical software skill. This is where APIs offer a solution to this problem, which the industry refers to as Machine Learning as a Service (MLaaS), e.g., Azure ML or AWS ML, just to mention a few (Atakan Cetinsoy et al., 2016).

The intention of this paper is to explore all three technologies—RDBMS, Hadoop, and APIs—to determine which one is the best fit for data extraction and processing in the context of Neural Networks implementation. This consideration is crucial, given that many individuals interested in Machine Learning are not software developers, and the need for a 'plug-in' to deploy their ML models is evident.

# Topic overview

The chosen topic is storage solutions and Neural Networks, with NN being considered a type of Machine Learning (ML) process known as deep learning (Mishra and Gupta, 2017). The field of Big Data is constantly growing and encompasses a need for efficient data management and processing tools. Two well-known tools for handling and analyzing large datasets are Relational Database Management Systems (RDBMS) and Hadoop. However, the rampant advancement of Machine Learning and Neural Networks, the integration of these data management tools with advanced analytics technologies is the focus of this paper.

## Objectives

* Examine the current state of RDBMS, Hadoop, and APIs when used in modeling NN and a CNN.
* Store a 1.6GB dataset in both a RDBMS (MySQL) and Hadoop, and then retrieve the data into a Jupyter Notebook to model a neural network.
* Utilize an API (*Keras*) to model a convolutional neural network and compare its performance in conjunction with RDBMS and Hadoop.
* Discuss the rationale behind the selection of the NN and CNN model for each scenario.

## Research question

How do Relational Database Management Systems (RDBMS) and Hadoop compare in terms of efficiency and effectiveness in processing large datasets for the application of neural networks, and how can APIs, particularly the Keras library, streamline the implementation of neural network models in data analytics (Lee and Song, 2019).

# State of the art

## RDBMS

The current state of RDBMS has evolved substantially with enhancements in storage, speed, and scalability by using cloud-based solutions (Li et al., 2016). The future holds a shift for RDBMS transitioning to a NoSQL database or having a unified metamodel for both (Candel, Sevilla Ruiz and García-Molina, 2022). To understand why NoSQL is taking over RDBMS, it is crucial to talk about: Schemas, where NoSQL uses dynamic instead of static schemas; the type of data to be stored, with NoSQL databases offering advantages for hierarchical data storage due to their flexible data models and scalability, while RDBMS are not that flexible; scalability, with NoSQL depending on horizontal scalability and RDBMS on vertical scalability; and other points where NoSQL surpasses RDBMS, including data warehouse, complexity, cloud, and big data handling, and output performance (Palanisamy and Suvitha Vani, 2020).

## Hadhoop

In the era of Big Data, where we sometimes run out of storage and face difficulties on a single host due to the volume of data, Hadoop came into the scene to tackle this by offering computational capabilities over huge amounts of data (Lam, 2011). The present and future look bright for Hadoop, as some of the major Big Data companies, such as *Google, Facebook, eBay, Twitter* and *Spotify,* rely on this technology (Neeta Awasthy and Nikhila Valivarthi, 2023).

## ML APIs

Machine Learning APIs have helped developers integrating data flows into complex algorithms without requiring deep expertise. These APIs were once primarily used for basic tasks like picture and speech recognition, but they have since grown to include a variety of machine learning activities, such as predictive analytics and natural language processing (Halevy, Norvig, & Pereira, 2009). Today, they are essential to leading tech companies and cutting-edge industries, being ML accessible and customizable than in the past. Looking at the future ML APIs will remain as they have simplified model development across diverse environments (Jin et al., 2023).

## Deep Learning Neural Networks

The development of Deep Learning Neural Networks (DLNNs) traces back to the 1950s. The method has improved since the introduction of Convolutional Neural Networks (CNNs) by LeCun et al., in the late 1980s, which showed how good deep learning architectures are for image processing (LeCun, Bengio, & Hinton, 2015). In 2012 this field achieved another milestone with AlexNet success in the ImageNet challenge, showcasing DLNNs potential in image recognition (Krizhevsky, Sutskever, & Hinton, 2012). The introduction of Transformer models by Vaswani et al. in 2017 marked another significant advancement, in natural language processing (Vaswani et al., 2017). Deep learning NNs have the future guaranteed as they are extensively used to model data stored in RDBMS, HDSF and ML APIs.

## Research methodologies and key papers

The research methodology used in this paper is experimental, as can be seen in Section V. *Implementation.* The study aims to determine cause-and-effect relationships by manipulating one or more variables while controlling others.

Four papers have been crucial to unblock roadblocks faced during the elaboration of this paper:

* An Evaluation of Training Size Impact on Validation Accuracy for Optimized Convolutional Neural Networks (Barry-Straume et al., 2019). This paper validates the impact of training sizes. The concept of validation is adopted and applied to validate the NN created using HDFS and MySQL data. After adjusting the model, the NN showed promising results. However, the training accuracy and loss had not been validated. Upon introducing validation, it was revealed that the model was overfitted. Thanks to this validation, further steps were taken to mitigate the overfitting.
* Is Your Dataset Big Enough? Sample Size Requirements When Using Artificial Neural Networks for Discrete Choice Analysis (Alwosheel, van Cranenburgh, and Chorus, 2018). This paper focuses on determining the optimal sample size for training ANNs. In point *3) Overfitting Assesment and Mitigation,* after validation more data was added to the model; specifically, the number of rows increased from 2,907 to 11,628. This strategy of augmenting the dataset effectively resolved the issue of overfitting.
* Deep Learning with Python (Chollet, 2018, pp. 253-259). This book provided the example used in section F. *Keras Data to Model a Convolutional Neural Network.* It offers a great illustration of how an API like Keras can be utilized in modelling a CNN.
* Regularization of deep neural networks with spectral dropout (Khan, Hayat and Porikli, 2019). This research enhances the concept of dropout introduced by the authors of the ImageNet Challenge (Krizhevsky, Sutskever, & Hinton, 2012). This technique clearly reduces overfitting in CNNs, after testing Chollet CNN, it clearly overfitted. Dropout was introduced in phase 2 and results were satisfactory mitigating overfitting.
* Regularization of Deep Neural Networks with Spectral Dropout (Khan, Hayat, and Porikli, 2019). This research builds on the concept of dropout, originally introduced by the authors of the ImageNet Challenge (Krizhevsky, Sutskever, & Hinton, 2012). The technique significantly reduces overfitting in CNNs. After testing the Chollet CNN model, clear evidence of overfitting was observed. Dropout was introduced in phase 2, and the results were effectively satisfactory in mitigating overfitting.

# Literature review

This literature review is based on a topic paper that explores how technologies like RDBMS, HDFS, and APIs are shaping the constantly evolving field of deep learning, particularly in Neural Networks. To develop this paper, 16 articles and 2 books were reviewed, contributing to its creation. The sources have been thematically classified into two categories:

* Storage Solutions: Comparing HDFS and RDBMS.
* Machine Learning APIs, and Neural Networks.

## Storage Solutions: Comparing HDFS and RDBMS.

Chuck Lam (2011) provides a solid foundation in the basics of Hadoop, including its architecture, file system (HDFS), and core components like MapReduce. He demonstrates how Hadoop can be leveraged to process large datasets efficiently also shows Hadoop in real worlds applications for large organisations such as the New Your Times, China Mobile, StumbleUpon and IBM. This source helps justify the importance of Hadoop as a storage solution and its emergence by providing computational capabilities over large amounts of data. Hadoop then came into play to tackle the challenges posed by Big Data for large corporations, a focus of Neeta Awasthy and Nikhila Valivarthi (2023) study. The index building for Google Search involves massive datasets and performs statistical analysis in indexing through large-scale batch processing. Facebook manages two primary clusters that store approximately 12 terabytes of data each. Additionally, eBay maintains substantial clusters designed to manage the data of 180 billion active users. To this paper, the value of this study lies in how it demonstrates the real-world applications of HDFS. A key component for RDBMS is memory and its performance role it plays that is what Li et al., (2016) studied. They compared physical memories (SSD or HDD) with cloud memory (remote direct access, RDMA). This paper examines four scenarios to demonstrate that remote memory hosted in servers outperforms traditional physical memory systems. It is relevant to note that the state of the art of RDBMS has dramatically improved thanks to cloud-based solutions. Palanisamy and Suvitha Vani (2020) conducted a survey comparing the concepts of NoSQL and RDBMS, including their limitations, and also addressed the advantages and types of NoSQL databases. Their research concluded that a NoSQL database is an excellent choice when the data is structured, and the volume is not large, while NoSQL is preferable for unstructured data or for structured data with the potential for rapid growth. Essential to this paper is the way this article summarizes the points where NoSQL overtakes RDBMS. Arshad et al., (2023) compare NoSQL technologies with traditional RDBMS in the context of Big Data analytics. The paper describes NoSQL as "Not Only SQL" and categorizes these databases into key-value stores, document databases, wide-column stores, and graph databases, with Hadoop classified as a wide-column store. It outlines the evolution of Big Data from megabytes and gigabytes to terabytes and petabytes, constantly challenging the industry to develop new storage solutions to meet escalating demands. The nature of Big Data is also elucidated in terms of its volume, variety, velocity, and variability. Furthermore, the paper details the ACID properties of RDBMS—atomicity, consistency, isolation, and durability—and compares them with the CAP theorem of NoSQL, which emphasizes strong consistency, high availability, and partition tolerance. The authors conducted a survey among relevant IT companies, revealing a preference for NoSQL technologies when managing Big Data. The study concludes that applications dealing with Big Data tend to perform better in NoSQL environments. Candel, Sevilla Ruiz, and Garcia-Molina (2022) are determined to prove that a unified metamodel for NoSQL and relational databases represents the future in the field of databases, especially as NoSQL technologies have gained popularity recently. In their conclusion, the authors present a metamodel named U-Schema, where both NoSQL and RDBMS are optimized. Pertinent to this paper is an understanding of the current state of the art concerning RDBMS.

## Machine Learning APIs, and Neural Networks.

Halevy, Norvig, and Pereira (2009) argue that large volumes of data can be more valuable than complex algorithms in developing artificial intelligence systems. They maintain that large-scale data can improve language processing and that simple algorithms can outperform complex ones when coupled with massive datasets. How this study fits into the paper is significant because it demonstrates an early stage of what is now known as an API; the authors used web-based data to train their models. The evolution of ML API (Atakan Cetinsoy et al., 2016), traces their journey from the era when researchers relied on pen and paper, to the present, where IT systems deploy these ML models. It also addresses current challenges, emphasizing the treatment of data before implementing an ML model. The authors highlight solutions such as Machine Learning as a Service (e.g., Azure ML, AWS) and REST APIs (e.g., BigML), showing how these technologies are valuable because they provide evidence of large companies creating and using APIs for modelling their ML models. A big breakthrough happened when Krizhevsky, Sutskever, and Hinton (2012) explain the development of a deep convolutional neural network (CNN) that outperforms prior models in classifying high-resolution images in the ImageNet challenge. The CNN features 60 million parameters and 650,000 neurons across five convolutional and three fully connected layers. This study is one of the first to introduce the term 'dropout' as a method to prevent overfitting. Continuing with the development of deep learning LeCun, Bengio, and Hinton (2015), emphasizing the ability of deep learning to create computational models that recognize complex patterns through multiple processing layers. The authors describe how deep learning has improved areas like speech recognition, visual recognition, object detection, drug discovery, and genomics. All these advancements are thanks to the use of backpropagation algorithms. It also mentions that deep convolutional networks are great for image, video, audio processing, text processing, and speech processing. A new neural network architecture, known as the Transformer, represents an evolution from recurrent and convolutional neural networks and was introduced by Vaswani et al. (2017). The Transformer employs attention mechanisms, which enable better parallelization, reduce training times, and enhance performance on machine learning translation tasks. This paper concludes that the Transformer is a superior model to its predecessors in terms of translation models. According to Mishra and Gupta (2017), deep learning surpasses traditional ML models in its ability to perceive text and images. Neural Networks, as a crucial component of deep learning, are discussed in depth—specifically, ANNs and CNNs, which are key to this paper. It concludes that NNs are among the more popular techniques for solving deep learning problems. Chollet (2018) provides great examples of best practices for deep learning, particularly the use of Keras as an API for modelling CNNs. Chollet demonstrates how Keras streamlines data processing in a CNN structure, from processing IMDB reviews to constructing network layers. This example was instrumental to this paper, showcasing Keras robustness and ease of use in the context of neural networks. Lee and Song (2019) focus on examining parameter estimation procedures for deep neural networks, as well as the structures of CNN models, ranging from basic to advanced techniques. The authors are also determined to demonstrate the critical steps in CNNs that enhance image classification performance on the CIFAR-10 dataset using Keras. Their conclusions suggest that utilizing multiple stacks of convolutional layers along with batch normalization can lead to improved predictions. Another important finding that is relevant to this paper is that Keras, due to its popularity as a neural network API, enables individuals to quickly familiarize themselves with deep learning methodologies. Jin, Chollet, Song, and Hu (2023) present AutoKeras, an Automated Machine Learning (AutoML) library designed to simplify the application of deep learning. It offers a solution for challenges in model selection and hyperparameter tuning, thereby making deep learning accessible to those with limited technical expertise. The library provides a user-friendly interface, assisting inexperienced users in addressing machine learning problems with minimal coding required. This study represents a significant step towards democratizing deep learning technology and aligns with the promising future of ML APIs discussed in this paper. Barry-Straume, Tschannen, Engels, and Fine (2018) assess how varying training set sizes influence the validation accuracy of CNNs. The study determines the optimal data volume required to achieve maximum accuracy during model validation. Furthermore, the research indicates that larger datasets can significantly enhance the predictive capabilities of CNNs. This study is crucial to the present paper as it introduces the concept of validation. In the implementation phase, validation was conducted to assess overfitting. Initially, the neural network and the specific CNN in focus here were prone to overfitting; however, through validation, this was effectively mitigated. Alwosheel, van Cranenburgh, and Chorus (2018) examine the appropriate sample size for ANNs in discrete choice modelling. They fill a gap in empirical guidelines by establishing a rule-of-thumb based on Monte Carlo analyses of both synthetic and real data. Their research suggests that a dataset size fifty times the number of weights in the ANN is more effective than the traditionally used benchmark of a dataset only ten times the weight count. This research illuminates a section of this paper, where, in phase 2, an overfitted NN was corrected by increasing the dataset size, thus resolving the overfitting issue. Khan, Hayat, and Porikli (2019) introduce 'Spectral Dropout,' an enhancement of the dropout technique newly introduced for the ImageNet challenge in 2012 by Krizhevsky, Sutskever, and Hinton (2012). Spectral Dropout augments traditional CNNs with a decorrelation transform and tackles overfitting by mitigating weak and noisy Fourier domain coefficients of network activations. The research concludes that this method's efficacy surpasses current regularization methods and increases network training speed. This paper is crucial in addressing overfitting in CNN phase 1, highlighting that since 2012, researchers have been using and improving dropout, leading to its implementation in CNN phase 2.

In conclusion, this literature review has identified two significant gaps:

* The application of HDFS and RDBMS in NN contexts.
* The needs of non-technical users who must model NNs using datasets large enough to overwhelm conventional applications. All literature reviewed was directed towards technical users proficient in advanced programming techniques for modelling NNs.

This paper aims to address the gaps mentioned above.

# Critical evaluation

After the implementation that can be seen in the Annex, the research has found the following key findings:

## HDFS vs. MySQL dataset loading time

In this process, HDFS clearly showed its advantages over MySQL. MySQL required schema and table creation, and the data load was done via Jupyter Notebook because MySQL Workbench failed to import the data. Using the batching function, data was successfully loaded into the table, taking 7 minutes and 28 seconds. On the other hand, HDFS only required the file to be moved into the Hadoop directory. The nature of HDFS as a file system that directly handles large files without the need for schema definitions or data processing allows for faster data loading compared to MySQL. The implication is clear, HDFS is faster when storing large datasets. The research also aligns with this implication (Palanisamy and Suvitha Vani, 2020), although a limitation might be that no further research was performed to decrease MySQL loading time using Jupyter Notebook.

## HDFS vs. MySQL storage memory usage

As seen above, a file of size 1.63 GB was processed by both systems. After loading, the size in Hadoop was 1.52 GB, whereas in MySQL, it was 1.87 GB. This efficiency is attributed to HDFS being optimized for storing and processing large data sets. It stores data in blocks across a distributed cluster, whereas MySQL is a row-based storage system that uses storage engines like *InnoDB*, which can increase the size of the dataset. This implies that Hadoop is more efficient than MySQL in storing files, a finding also supported by the research of Neeta Awasthy and Nikhila Valivarthi (2023). A limitation of this study is that no further research was conducted to decrease the size of the MySQL dataset. A limitation perhaps refining data types when creating the table could help reduce the dataset size.

## HDFS vs. MySQL data extraction for modeling a NN

Both extractions were performed using Jupyter Notebooks. For HDFS, the extraction was achieved using Spark, and for MySQL, a connection was established to pull data from the database. In this process, HDFS again showed its advantages over MySQL. HDFS completed the task in 2 minutes and 51 seconds, whereas MySQL took just 6 seconds. However, this result is not in favor of MySQL because, using Spark, the entirety of the dataset, which is 14,000,000 rows, was pulled, whereas the same was not achievable with MySQL due to this error: *“OperationalError 2013 (HY000): Lost connection to MySQL server during query.”* This was because the RAM was insufficient for the process, leading to a workaround of pulling only 2,000,000 distinct rows. The implication of this point is that Spark is more efficient than MySQL when extracting large amounts of data, a finding that is also supported by the research of Arshad et al. (2023). A limitation for MySQL could be that the batch function was not used when attempting to pull the 14,000,000 rows, which means loading times could not be accurately compared between Spark and MySQL for data extraction.

## Neural Network feature adjustment

The NN architecture is composed of a first layer with 12 neurons, followed by a second layer of 8 neurons, and an output layer with a single neuron classifying the outputs as 1 for Male and 0 for Female. In the initial phase, the NN had *input\_dim=2*, indicating the model should expect two input features: job title and age*.* The results were very poor after training over 100 epochs; the model achieved 50% accuracy and exhibited a 70% loss, model running time was 7 minutes and 3 seconds. These numbers remained constant during the training period. After making some adjustments and recognizing that neural networks perform better with more features, the job title column was transformed, treating each job category as a separate feature, and using people ages as values. This transformation increased the number of input features to 639. With the new data distribution, after over 100 epochs, the accuracy increased to 99.86% and the loss decreased to 1.34%. However, without further validation, there is a possibility that the model might be overfitting. The model’s running time is 8 seconds, and by distributing the data accordingly, the model not only performs better but also requires less computing time. This demonstrates the real implication of adding more features for improved NN performance. Literature research did not yield a conclusive study on the optimal number of NN features; however, this finding helps bridge that gap. A limitation could be that this study did not identify the ideal number of features or the number that would maximize accuracy and minimize loss over epochs.

## Neural Network validation

The NN was performing well in terms of training accuracy and loss, but there was no visibility on validation. Training accuracy and validation loss were then introduced, revealing a significant divergence; over 100 epochs, the training accuracy reached 99.74%, while the validation accuracy was only 49.48%. The training loss was 1.93%, and the validation loss skyrocketed to 188.22%, clearly indicating that the model was overfitted. The introduction of validation into the NN demonstrates that the model can be fully evaluated to determine whether it is learning and generalizing well on unseen data, which, in this case, is not. The literature, specifically Barry-Straume et al. (2019), provided guidance on implementing validation. A potential limitation is that without introducing validation into an NN, the model's performance may always appear skewed, not accurately indicating whether it is fitted or overfitted.

## Neural Network overfitting mitigation

By adding more data to the model, overfitting was reversed, leading to a well-fitted model where training and validation scores converged. The number of rows increased from 2,907 to 11,628 by concatenating the data frame four times. Over 100 epochs, the training accuracy reached 99.97%, and the validation accuracy was 99.61%. The training loss was 0.083%, and the validation loss was 3.9%. The model is learning and generalizing well on unseen data. The implication of adding more data in terms of NN performance is clear: the model performs better with more data. The literature review also supports this finding (Alwosheel, van Cranenburgh, and Chorus, 2018); they suggest that a dataset size fifty times the number of weights in the ANN is more effective than the traditionally used benchmark of a dataset only ten times the weight count. A limitation might be that, while using synthetic data makes the model easier to adjust, using real-world data may require further considerations to achieve similar results.

## Keras Deep Learning API vs. HDFS and RDBMS

*Keras* is simple, flexible, and powerful (keras.io, n.d.). Being a deep learning API, it is an excellent choice for learning how to model NN without the need for data preparation and processing like with HDFS or RDBMS. However, a significant limitation could be that, depending on the domain the study aims to model, *Keras* may seem insufficient in terms of datasets available for use. The implication in terms of learning and ease of use is clear: *Keras* is the best option for novice users who do not have HDFS or RDBMS knowledge. During the research, no literature was found comparing *Keras,* HDFS, and RDBMS in neural network contexts; this study aims to address this gap. Nonetheless, an excellent *Keras* implementation example modeling a CNN was found (Chollet, 2018, pp. 253-259).

## Convolutional Neural Network overfitting mitigation

After implementation and validation the Chollet CNN model, it was clear that the model was overfitted, as validation scores were nowhere near the training scores. A set of changes were introduced, with dropout being the most impactful. A potential limitation might be presented by additional CNN architectural changes, such as regularization and a smaller batch size; these changes could enhance the effectiveness of dropout, but this factor was not measured in this study. No gaps were found in the literature regarding the concept of dropout, which is extensively covered by Krizhevsky, Sutskever, & Hinton (2012) and Khan, Hayat, and Porikli (2019). The implication of dropout in this experiment is evident: it effectively corrected the overfitting.

## Rationale behind the selection of a NN and a CNN

Many NN structures were tested and discarded during the implementation phase of the paper. After numerous trials, the focus shifted to numbers and text, as prior attempts with images proved to be more difficult to adjust than the models discussed. It became clear that numbers and text would be the inputs for modelling the NNs, leading to the task of finding the right NNs. As seen in the annex implementation section, a NN was selected to classify gender based on job and age, and a CNN was chosen to process text and classify reviews. It is fair to say that these NNs performed well during the modelling phase, which is why they were selected and, most importantly, this explains the rationale behind their selection. The limitation is evident: there are many other NN structures and data types to experiment with. However, given that one of the objectives was to model both an NN and a CNN, this limitation leaves room for future work. In regards to the CNN, Chollet (2018, pp. 253-259) was a great help, and for the NN, the examples shown by my lecturers were instrumental in shaping this example.

# Conclusions

All objectives have been thoroughly explored and discussed throughout the paper:

1. The current state of RDBMS, HDFS, and APIs was examined in the context of modeling a NN and a CNN.
2. A dataset of 1.6GB was loaded into MySQL and HDFS to model the same NN.
3. The Keras library was explored for modeling a CNN, comparing its performance with that of RDBMS and HDFS.
4. The rationale behind the selection of a NN and a CNN was detailed.

Conclusions drawn from this experiment include:

1. HDFS is more efficient than MySQL in terms of storage memory usage and processing.
2. A deep learning API like Keras can be an excellent starting point for non-technical users who wish to model a NN. The technical knowledge required for using MySQL or Hadoop signifies why Keras can be a viable alternative.
3. NNs must be validated; training scores alone are insufficient to determine if a model is overfitted. Validation of accuracy and loss should be conducted and compared with training scores.
4. A NN will generalize better to unseen data if more data is fed into the model. An effective approach to mitigate overfitting is to augment the dataset.
5. Dropout should be implemented and tested when a CNN is overfitted; in this case, it proved helpful in reducing overfitting.

# Bias, Ethics and Validity

This section examines the principles of validity, bias, and ethics within the paper, with a special focus on the dataset *(people.csv)* used for modelling a NN, as it classifies gender based on job title and age. Additionally, the classification problem itself can be problematic, as it may discriminate against certain age or job groups. This research has addressed these concerns in the following points:

## Bias

To reduce bias in this study, a synthetic dataset *(people.csv)* was selected to ensure neutrality. The dataset is randomly generated, which helps to minimize biases that could arise from an unbalanced dataset where certain demographics or job sectors differ significantly from one another. There is one direct bias related to my technical knowledge: the implementation of MySQL. As I am quite familiar with this technology, it has been used in this study.

## Ethics

At this point, two concerns arise: the dataset *(people.csv)* adhering to GDPR requirements and the conclusions drawn from the NN study. The first concern is mitigated by the use of a dummy-generated dataset, which contains no real information. Consequently, no further remarks have been made on the NN model's findings. The efforts were focused on creating a responsive model rather than drawing conclusions based on gender classification by job title and age.

## Validity

Validity is demonstrated throughout this paper by ensuring that the research methods accurately measure what they are intended to measure and support the conclusions drawn. Furthermore, this paper can be reproduced by anyone with the proper setup who follows the implementation guidelines.

# Future work

Several points have room for improvement:

## Using a real world dataset instead of a synthetic one

It would be interesting to evaluate the NN model with real world data, and compare the results obtained with the synthetic one. However this future study requires a strict adherence to GDPR, bias and ethics, these questions must be carefully reviewed before conducting this research.

## Expanding the selection of NN, CNN and different data types

This study has effectively modelled a NN and a CNN. Future steps would include exploring different NN architectures as well as processing different data types, such as images, sound, or video.

## HDFS processing tools

Spark has been used for processing the data stored in HDFS. Future steps include using Hive, HBase, Flink, among others; the list is long.

##### Annex

# Word count and Github Links

The word count, includes all sections except from the annex, also does not include headers, sub-headers and Harvard citations, total word count equals to 4,856 words.

GitHub links:

* Student repository:

https://github.com/JoseRicoCct/CA1\_Integrated\_Assesment\_MSc\_Data\_Analytics\_CCT\_Semester\_2.git

* CCT repository:

https://github.com/CCT-Dublin/adv-data-big-data-sb-ca1-JoseRicoCct.git

# Implementation

This section explains the demo work with all its components.

## Technical Setup

Hardware and software configuration of the host laptop and the VM.

* Laptop: HP 250 G8 PC.
  + Operating System (OS): Microsoft Windows 10 Pro.
  + Processor: 11th Gen Intel® Core™ i7-1165G7 @ 2.80GHz, 2803 Mhz, 4 cores, 8 logical processors.
  + RAM: 16GB.
  + Hard Disk Drive (HDD): 237GB.
* VirtualBox: Version 7.0.14.
  + OS: Ubuntu 22.04 LTS (Jammy Jellyfish) (64-bit).
  + Processor: Configured with 2 cores and 2 logical processors from the host's 11th Gen Intel® Core™ i7-1165G7 processor.
  + RAM: 4GB.
  + Memory (Disk Space): 100GB.

Essential software versions installed on VM:

* Hadoop: 3.3.6.
* Spark: 3.4.2.
* MySQL: 8.0.36.
* MySQL Workbench: 8.0.36.
* Jupyter Notebook: 6.4.8.

Software used to record the demo:

* OBS-Studio 30.1.1.

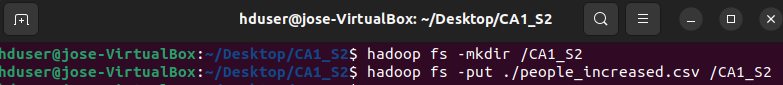
## Dataset

The dataset loaded into Hadoop and MySQL was sourced from the Datablist website (www.datablist.com, n.d.). It comprises nine columns and two hundred rows, containing personal data such as names, surnames, gender, job positions, etc. A significant consideration with this type of data is privacy; however, the data was randomly generated by the Python Faker package, as seen on the Datablist GitHub account (GitHub, 2023). This approach ensures GDPR compliance, carefully avoiding any conflict with it. Importantly, this dataset does not require a license for use, as this data is dummy generated for testing purposes.

## Data load

At first *people.csv* weighted 0.23 GB however for testing purposes it was increased to 1.6GB using *1.Increasing\_dataset\_size.ipynb* script. The dataset was duplicated seven times, resulting in *people\_increased.csv.* This choice is made with the intention of approaching Big Data. Although 1.6GB is not near what Big Data looks like nowadays (Arshad et al., 2023), it is close to concept in terms of overwhelming most conventional applications. For instance, Excel CSV grid will crash automatically when attempting to open this file. Yes, we can use Notepad ++ to how the data looks but no more than that.

### Hadoop people\_increased.csv load: First step creating a new directory and moving the dataset into it:



1. Commands to create and move a dataset into Hadoop.

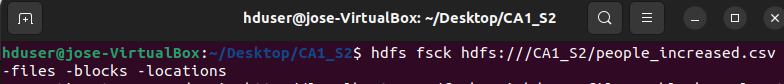
After a successful load Hadoop UI shows:

A screenshot of a computer

Description automatically generated

1. Hadoop UI, Utilities, Browse the file system, *CA1\_S2* directory.

Above figure shows a 1.52GB file that is replicated once with a block size of 128MB. To get a sense of how HDFS works we need to run the following command:



1. HDFS report health files command.

After running it, we see the file divided and stored across 13 blocks:

A computer screen shot of a program

Description automatically generated

1. Console output HDFS report health files command.

This means the HDFS has filled 12 blocks completely 128MB (134,217,728 bytes) in size, consistent with HDFS's default block size setting except the last one as is the remainder, with 20.24MB (21,234,682 bytes). This is common as the final block not using the default block size unless is a multiple of that block size.

### MySQL people\_increased.csv load: Before loading the file a schema and a table inside must be created:

A screenshot of a computer

Description automatically generated

1. MySQL workbench schema creation.

A screenshot of a computer

Description automatically generated

1. MySQL table creation.

After successful schema and table creation, data will be loaded via *2.Importing\_1.6GB\_CSV\_to\_MySQL.ipynb* script.This approach was chosen because MySQL Workbench server import failed, displaying the error : *”Error Code: 2013. Lost connection to MySQL server during query”.* MySQL workbench appears to struggle with importing large CSV files into a schema. However, the script did with a time of 7 minutes and 28 seconds. Let us examine the size of the table:

A computer screen shot of a computer code

Description automatically generated

1. MySQL table size

### HDFS vs MySQL loading process: After this implementation step, it is clear that HDFS is quicker in terms loading time. It required just two commands and took only 5 to 10 seconds for the file to be integrated in the system whereas, MySQL took 7 minutes and 28 seconds. Furthermore, MySQL required the creation of a schema and table. A second advantage of HDFS is memory comsumption; from an intial 1.63GB csv file when loaded, it was reduced to 1.52GB, whereas MySQL, upon loading, increased to 1.87GB.

## Data from HDFS Modeled with a Neural Network

Script *3.HDFS\_Data\_to\_model\_NN.ipynb* will guide the entire process outlined in this section. Once the data is loaded into HDFS, it must be extracted to implement the neural network. To facilitate this, a Python package called PySpark is used to read, process, and analyse the data stored in HDFS. The Apache Spark Web UI can be accessed at localhost:4040, where all the application stages can be viewed.

**A screenshot of a computer

Description automatically generated**

1. Apache Spark Web UI.

The running time for extracting data from HDFS and printing it into a pandas data frame is 2 minutes and 41 seconds; this will be compared with the running time for data extraction from MySQL.

The next step is to implement the NN, which comprises three distinct phases:

### Initial Model Setup: The model is a fully connected feedforward neural network, often known as Multilayer Perceptron (MLP). The architecture breaks down in:

* The first layer is a dense layer with 12 neurons, indicated by *input\_dim=2*, which means the model should expect two input features: job title and age. The *relu* activation function is used as a linear function that outputs the input directly if it is positive, and zero if it is negative. This function was chosen due to the simplicity it brings to the model and its suitability for the binary nature of the problem, classifying gender as male or female based on job title and age.
* The second layer is a dense layer with 8 neurons and *relu* activation.
* The output layer, the final layer, is a dense layer with a single neuron that uses the *sigmoid* activation function. For binary classification tasks the *sigmoid* function is very convenient, as it maps any input value to an output between zero and one.

A network of green and blue dots

Description automatically generated

1. MLP structure (alexlenail.me, n.d.).

The compilation is carried out using *binary\_crossentropy* loss function which measures the difference between predicted binary outcomes and actual binary labels. The Adam optimizer adjusts the weights and the model tracks the accuracy as a metric for evaluation.

The results after training for 100 epochs, the model achieved 50% accuracy and showed 70% loss rate with no variation whatsoever. Clearly, it is not performing well and needs adjustment. The model running time is 7 minutes and 3 seconds.

A screenshot of a graph

Description automatically generated

1. Training Accuracy and Loss Phase 1.

### Model Data Input Adjustment: Initially, we had 2 input features. Upon realizing that neural networks perform well with more features, we transposed the job title column, treating each job category as a separate feature with people age as values. After this transformation, the number of input features increased to 639. With the new data distribution, over 100 epochs, accuracy increased at 99.86% and loss decreased at 1.34%. However, without further checks, there is a possibility that the model might be overfitted. Model running time takes 8 seconds, by the distributing the data accordingly the model not only performs better it takes less computing time.

A graph of a training curve

Description automatically generated with medium confidence

1. Training Accuracy and Loss Phase 2.

### Overfitting Assesment and Mitigation: To assess overfitting, data validation must be conducted (Barry-Straume et al., 2019). To determine if overfitting is present, a comparison between training and validation accuracy should reveal a significant difference; the same applies to training and validation loss. Over 100 epochs, the training accuracy is 99.74% and validation accuracy is 49.48%. Training loss is 1.93%, and validation loss is 188.22%. These results clearly indicate an overfitted model.

A graph of a graph of a graph

Description automatically generated with medium confidence

1. Training accuracy and loss phase 3 overfitted model.

By adding more data to the model, overfitting was reversed (Alwosheel, van Cranenburgh and Chorus, 2018), leading to a well-fitted model where training and validation scores converged. The number of rows increased from 2,907 to 11,628 by concatenating the data frame four times. Over 100 epochs, the training accuracy reached 99.97%, and the validation accuracy was 99.61%. Training loss was 0.083%, and validation loss was 3.9%. The model is learning and generalizing well on unseen data.

A graph of a model training

Description automatically generated

1. Training accuracy and loss phase 3 fitted model.

## Data from MySQL Modeled with a Neural Network

In this section, data is pulled from a MySQL database, however, NN configuration remains unchanged. The script used for this demonstration is *4.MySQL\_Data\_to\_model\_NN.ipynb*. When comparing running times, HDFS Spark took 2 minutes and 41 seconds, whereas MySQL took just 6 seconds. At first glance, it appears that MySQL is outperforming Spark, but this is not actually the case. For the Spark demonstration, 14,000,000 rows were loaded into a Spark data frame variable, while in the MySQL scenario, the selection was limited to 2,000,000 rows. MySQL performance significantly deteriorates when pulling larger datasets. When attempting to fetch 14,000,000 rows, the following error was encountered:*“OperationalError 2013 (HY000): Lost connection to MySQL server during query”.* This indicates that there is not enough memory (RAM) available to handle the process.

The main difference between MySQL and HDFS Spark can be established. HDFS Spark did handle the 14,000,000 rows because it is designed for distributed computing, that means it process data parallelly across multiple nodes, in this case only one node was used, however the way HDFS stores data as blocks and the sequential data access patterns Spark uses, makes the combination of Hadoop and Spark a better choice than MySQL for large datasets. However, it is possible to extract the 14,000,000 rows using MySQL by emulating the loading process for *people\_data* table. The *chunksize* function can assist in batching the records and transferring them into a pandas data frame. While MySQL can be an excellent option for smaller datasets, in this scenario, it has been shown that HDFS and Spark surpass the performance of a traditional RDBMS.

## Keras Data to Model a Convolutional Neural Network

Unlike the two sections mentioned above, data has not been processed in this section. This is because Keras comes with built-in data preprocessing capabilities, packaging the data in a form that is almost ready for model deployment. All relevant code can be found in *5.API\_Data\_to\_model\_CNN.ipynb.* Machine learning libraries like *Keras* are quite convenient for deploying models, as they require no specialized skills compared to setting up a Hadoop or MySQL database. Moreover, *Keras* provides subsidiary libraries like *TensorBoard* and *keras.utils,* which offer visual insights to understand model performance and architecture. Another advantage of using *Keras* is its quick loading time, which enables faster experimentation.

The example explored in this section involves training a 1D CNN for the IMDB sentiment analysis task (Chollet, 2018, pp. 253-259). This example was chosen because it effectively demonstrates how a CNN can be implemented using the IMDB dataset in *Keras.*

### Model Phase 1: The model tested in this phase suffers no variation from its source (Chollet, 2018, pp. 253-259).

A diagram of a program

Description automatically generated

1. Architecture of 1D CNN.

Breaking down above figure:

* Embedding layer transforms the input data into dense vectors of fixed size, in this case 128.
* Conv1D layer performs convolution over the sequence, using 32 filters and a kernel size of 7. The intention is to recognize patterns among the word sequences.
* MaxPooling1D layer, reduces the input by taking the maximum value over a window of size 5. It helps reducing dimensionality and abstracting the features.
* A Second Conv1D layer, same function as the first one it helps to capture more complex patterns in the data.
* GlobalMaxPooling1D layer, takes the maximum value over the time dimension for each feature and reduces the output of the convolutions to a fixed size vector.
* Dense layer that outputs a single value for binary classification.

After training the model the scores indicate that it is overfitted. The validation loss increases with more epochs even when the training loss decreases. This means the model is memorizing the training data and not being able to generalise unseen data.

*A graph of different colored lines

Description automatically generated*

1. Training accuracy and loss phase 1, overfitted model.

### Model Phase 2: As the intial model was poorly performing the following changes have been put in place:

1. Regularization in the Conv1D and dense layers, that will add a penalty for weight magnitude to the loss function. This forces the model to learn smaller weights, leading to a simpler model.
2. Including dropout after the embedding layer and before the final dense layer. Dropout randomly disables neurons during training, forcing the network to learn redundant patterns, this makes a robust model (Khan, Hayat and Porikli, 2019).
3. Smaller batch size of 64, as opposed to 128 in the first model. Smaller batch sizes can lead to a regularization effect and that might help the model to generalize better.

A diagram of a flowchart

Description automatically generated

1. Architecture of 1D CNN with dropout.

CNN architecture has changed, its components can be explained as it follows:

* Embedding layer same as the first model but with a smaller output dimension 64.
* Dropout layer randomly sets a fraction of input units to 0 at each update during training to prevent overfitting.
* Conv1D layer with L2 regularization applies convolution but penalizes large weights due to the regularization.
* MaxPooling1D layer, same as in the first model, reduces dimensionality and abstracts features.
* GlobalMaxPooling1D layer, also like the first model, condenses the feature information.
* Dense layer with L2 regularization and dropout, introduces another point where regularization and dropout are applied.
* Output dense layer with sigmoid activation, outputs a probability for the binary classification task.

Overall, this CNN model is better structured than the one provided by the book (Chollet, 2018, pp. 253-259). Overfitting is mitigated through the use of regularization and dropout. As a result, the model memorizes less of the training data and is more capable of generalizing from the patterns learned during training, leading to improved validation performance. In terms of computing times, the first model took 11 minutes and 22 seconds to train, while the second model required only 4 minutes and 18 seconds

Adjusting the model, the parameter that made a significant difference was the dropout rate. Rates close to one did not address overfitting, while rates closer to 0.5 began to properly fit the model. The ideal dropout rate was found to be 0.4, determined through a trial-and-error approach.

A graph of different colored lines

Description automatically generated

1. Training accuracy and loss phase 2, less overfitted model.

##### Acknowledgment

A big thank you to my lecturers for all the support and guidance, and for creating these assignments. Honestly, they truly serve to enhance our knowledge and teach us how to find solutions for given scenarios.

##### References

Alexlenail.me. n.d. NN SVG. [online] Available at: https://alexlenail.me/NN-SVG/ [Accessed 22 March 2024].

Alwosheel, A., van Cranenburgh, S. and Chorus, C.G., 2018. Is your dataset big enough? Sample size requirements when using artificial neural networks for discrete choice analysis. Journal of Choice Modelling. https://doi.org/10.1016/j.jocm.2018.07.002

Arshad, M., Brohi, M.N., Soomro, T.R., Ghazal, T.M., Alzoubi, H.M. and Alshurideh, M., 2023. NoSQL: Future of Big Data Analytics Characteristics and Comparison with RDBMS. In: *The Effect of Information Technology on Business and Marketing Intelligence Systems*, pp.1927-1951. doi: 10.1007/978-3-031-12382-5\_106.

Barry-Straume, J., Tschannen, A., Engels, D.W. and Fine, E., 2018. An Evaluation of Training Size Impact on Validation Accuracy for Optimized Convolutional Neural Networks. *SMU Data Science Review*, 1(4), Article 12. https://scholar.smu.edu/datasciencereview/vol1/iss4/12

Candel, C.J.F., Sevilla Ruiz, D. and García-Molina, J.J., 2022. A unified metamodel for NoSQL and relational databases. *Information Systems, 104*, p.101898. <https://doi.org/10.1016/j.is.2021.101898>.

Cetinsoy, A., Martin, F.J., Ortega, J.A. and Petersen, P., 2016. The Past, Present, and Future of Machine Learning APIs. In*: PAPIs 2015 - Proceedings of the 2015 Conference on Predictive APIs and Apps*, vol. 50, JMLR: Workshop and Conference Proceedings, pp.43-49.‌

Chollet, F. (2018). *Deep Learning with Python*. Shelter Island, NY: Manning Publications. pp. 253-259.

Datablist. (n.d.). *Download Sample CSV Files for free*. [online] Available at: https://www.datablist.com/learn/csv/download-sample-csv-files [Accessed 6 March 2024].

GitHub. (2023). *Sample CSV files*. [online] Available at: https://github.com/datablist/sample-csv-files [Accessed 6 March 2024].

Halevy, A., Norvig, P., & Pereira, F. (2009). *The Unreasonable Effectiveness of Data*. IEEE Intelligent Systems, 24(2), 8-12.

Jin, H., Chollet, F., Song, Q. and Hu, X. (2023). AutoKeras: An AutoML Library for Deep Learning. *Journal of Machine Learning Research*, 24, 1-6.

keras.io. (n.d.). *Keras documentation: About Keras.* [online] Available at: <https://keras.io/about/> [Accessed 1 April 2024].

Khan, S.H., Hayat, M. and Porikli, F., 2018. Regularization of deep neural networks with spectral dropout. *Neural Networks.* https://doi.org/10.1016/j.neunet.2018.09.009

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). *ImageNet classification with deep convolutional neural networks.* Advances in Neural Information Processing Systems, 25.

Lam, C. (2011). *Hadoop in Action*. Manning Publications Co.: Stamford, CT.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). *Deep learning. Nature,* 521(7553), 436-444.

Lee, H. & Song, J. (2019). Introduction to convolutional neural network using Keras: An understanding from a statistician. *Communications for Statistical Applications and Methods,* 26(6), 591–610. https://doi.org/10.29220/csam.2019.26.6.591

Li, F., Das, S., Syamala, M. and Narasayya, V.R., 2016. Accelerating Relational Databases by Leveraging Remote Memory and RDMA. In: *Proceedings of the 2016 International Conference on Management of Data - SIGMOD ’16.* https://doi.org/10.1145/2882903.2882949.

Mishra, C. and Gupta, D.L., 2017. Deep Machine Learning and Neural Networks: An Overview. *IAES International Journal of Artificial Intelligence (IJ-AI)*, 6(2), pp.66-73. <https://doi.org/10.11591/ijai.v6.i2.pp66-73>.

Neeta Awasthy and Nikhila Valivarthi (2023). Evolution of Hadoop and Big Data Trends in Smart World. In S. Awasthi, G. Sanyal, C.M. Travieso-Gonzalez, P. Kumar Srivastava, D.K. Singh & R. Kant (Eds.), *Sustainable Computing* (pp.148-186). Cham: Springer. https://doi.org/10.1007/978-3-031-13577-4\_6

Palanisamy, S. and Suvitha Vani, P., 2020. A survey on RDBMS and NoSQL Databases: MySQL vs MongoDB. In: *2020 International Conference on Computer Communication and Informatics (ICCCI -2020)*, 22-24 January 2020, Coimbatore, India. IEEE.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). *Attention is all you need.* Advances in Neural Information Processing Systems, 30.

Delete this on final commit!!!