## Sentiment Analysis through Time Series using Big Database storage program, Cassandra

Author: Teresa Quain

Student ID:  [sba22235](mailto:sba22235@student.cct.ie)

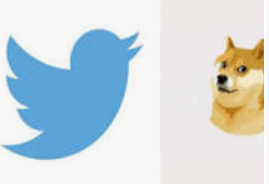
Student email: [sba22235@student.cct.ie](mailto:sba22235@student.cct.ie)

Module: Advance Data Analysis

CA2 60% Integrated Assessment, MSc in Data Analytics

Wordcount: 3000





Contents

[Table of Figures 2](#_Toc132456137)

[Table of Tables 2](#_Toc132456138)

[Abstract 2](#_Toc132456139)

[Literature review 3](#_Toc132456140)

[1. COVID-19 CT image recognition algorithm based on transformer and CNN 3](#_Toc132456141)

[2. A multi-biometric iris recognition system based on a deep learning approach 3](#_Toc132456142)

[3.Deep Feature Learning for Disease Risk Assessment Based on Convolutional Neural Network with Intra-Layer Recurrent Connection by Using Hospital Big Data 4](#_Toc132456143)

[4. Big data analytics in healthcare: promise and potential 4](#_Toc132456144)

[5. Deep learning-based remaining useful life estimation of bearings using multi-scale feature extraction 5](#_Toc132456145)

[6. Face mask recognition system using CNN model 5](#_Toc132456146)

[7. Performance Analysis of State-of-the-Art CNN Architectures for LUNA16 6](#_Toc132456147)

[8. Hyperparameter Optimization 6](#_Toc132456148)

[9. CNN Architectures and Feature Extraction Methods for EEG Imaginary Speech Recognition 7](#_Toc132456149)

[10. Impact of dataset size and convolutional neural network architecture on transfer learning for carbonate rock classification 7](#_Toc132456150)

[Research Proposal 9](#_Toc132456151)

[Introduction 9](#_Toc132456152)

[Research Questions 10](#_Toc132456153)

[Methodology 11](#_Toc132456154)

[Results 12](#_Toc132456155)

[Conclusion 19](#_Toc132456156)

[References 23](#_Toc132456157)

# Table of Figures

[Figure 1 Neural Network architecture 9](file:///E:\cct\assigments\draft1.docx#_Toc132406708)

[Figure 2 Model 1 Summary 14](file:///E:\cct\assigments\draft1.docx#_Toc132406709)

[Figure 3 Model 1 Accuracy Result after 5 epochs 15](file:///E:\cct\assigments\draft1.docx#_Toc132406710)

[Figure 4 Model 1 Accuracy Result after 20 epochs 15](file:///E:\cct\assigments\draft1.docx#_Toc132406711)

[Figure 5 Model 2 Accuracy Result after 20 epochs 17](file:///E:\cct\assigments\draft1.docx#_Toc132406712)

[Figure 6 Model 3 plot on Linux VM 17](file:///E:\cct\assigments\draft1.docx#_Toc132406713)

[Figure 7 Model 3 Summary 18](file:///E:\cct\assigments\draft1.docx#_Toc132406714)

[Figure 9 Model 3 Accuracy Results after 5 epochs 19](#_Toc132406715)

[Figure 8 Model 3 Accuracy vs Epochs (5) for cross validation and training accuracy 19](file:///E:\cct\assigments\draft1.docx#_Toc132406716)

[Figure 10 Model 3 Uncertainty vs. predicted values 19](file:///E:\cct\assigments\draft1.docx#_Toc132406717)

[Figure 11 Model 3 Accuracy vs Epochs (20) for cross validation and training accuracy 20](file:///E:\cct\assigments\draft1.docx#_Toc132406718)

[Figure 12 Model 3 Accuracy Results after 20 epochs 20](file:///E:\cct\assigments\draft1.docx#_Toc132406719)

[Figure 13 Model 3 Summary Runtime VM 21](file:///E:\cct\assigments\draft1.docx#_Toc132406720)

[Figure 14 Model 3 Summary Runtime Local 21](file:///E:\cct\assigments\draft1.docx#_Toc132406721)

[Figure 15 Model 3 Compile Runtime Local 22](file:///E:\cct\assigments\draft1.docx#_Toc132406722)

[Figure 16 Model 3 Compile Runtime VM 22](file:///E:\cct\assigments\draft1.docx#_Toc132406723)

[Figure 17 Model 3 Fit Runtime Local 23](file:///E:\cct\assigments\draft1.docx#_Toc132406724)

[Figure 18 Model 3 Fit Runtime VM 23](file:///E:\cct\assigments\draft1.docx#_Toc132406725)

# Table of Tables

[Table 1 Model Result Summary 13](#_Toc132406737)

# Abstract

Neural networks have the ability to learn from large and complex datasets, making them useful in applications such as image recognition, fraud detection, and prediction modelling. They are a subset of artificial intelligence that are designed to mimic the functioning of the human brain. These systems consist of layers of interconnected artificial neurons that process and transmit information.. Neural networks have been used in many areas, including computer vision, natural language processing, and speech and image recognition. The training process involves adjusting the weights of the connections between neurons through backpropagation, allowing the network to learn patterns and relationships in the data ie. Machine learning. NoSQL databases refers to non-SQL databases or ‘not only’ SQL databases. These databases store data in a different format to the typical SQL databases, that is relational tables. NoSQL provides flexible schemas for many types of data. It supplements the analytics tool kit rather than replaces active technologies. To rephrase, NoSQL doesn’t rely on data shaped in a certain way like SQL, so that it may be consistently retrieved. SQL differs by administering a rigid schema, clearly defined data, strong consistency and vertical scalability. Depending on the requirement of a business, the best fit between SQL and NoSQL can be decided. In distributed systems, NOSQL is often preferred due to the scalability problems of SQL, but there is other reasons also.

**Keywords**: Neural Networks, Artificial intelligence, NoSQL, Scalability, Big Data

Presentation of state of the art, including research methodologies

# Literature review

The literature review examines existing research and scholarly works on Neural Networks and Big Data Storage, providing a comprehensive overview of the current state of knowledge. It identifies key concepts, theories, methodologies, and findings, highlighting gaps and ongoing debates in the field. The review serves as a foundation for the research study and offers insights into the research context and significance.

[7] This article ‘Performance Analysis of State-of-the-Art CNN Architectures for LUNA16’ focuses on comparing the performance of three CNN architectures for accuracy, specificity, sensitivity and predictive value.

Several studies have explored the application of deep learning-based approaches to detect and diagnose lung cancer from medical images, with the aim of improving accuracy and reducing the error rate. For instance, a framework based on modified AlexNet (MAN) was proposed in [13] to detect lung cancer and pneumonia abnormalities using X-ray images, achieving 97.27% accuracy. Another study [14] presented a deep convolutional neural network called AlexNet to classify malignant and benign nodules, achieving 99% training precision and 97% validation accuracy. In [15], a deep learning algorithm based on segmentation was used for lung cancer detection, achieving high accuracy through five-fold cross-validation. In [16], a CAD system for pulmonary pure ground-glass nodules was presented using a convolutional neural network.

While these studies have shown promising results, there is still room for improvement in terms of performance and generalization to different datasets. This motivates the need for further research to investigate different optimization algorithms and architectures for lung cancer detection using deep learning. This study aims to fill this gap by comparing the performance of different CNN architectures (LeNet, AlexNet, VGG16, ResNet-50, and Inception-V1) with multiple optimizers (RMSProp, Adam, and SGD) on the LUNA16 dataset. The results show that AlexNet with SGD optimizer achieved the highest validation accuracy, sensitivity, specificity, and F1 score for CT lung cancer detection. This study provides valuable insights for researchers and medical experts on the application of deep learning for lung cancer detection, highlighting the importance of optimizing CNN architecture and using appropriate optimization algorithms.

## 8. Hyperparameter Optimization

[ machine learning approaches, springer series, implementation cost, comparison, <https://link.springer.com/chapter/10.1007/978-3-030-05318-5_1>]

[8] The article hyperparameter Optimisation is part of the Springer series on challenges in Machine Learning book by Matthias Feurer. The book provides a high level but comprehensive overview of the challenges facing HPO, hyperparameter optimization since it began such as implementation cost, configuration space, commercial restrictions. The author compares the high computation costly method vs. a multi-fidelity method that employs cheaper variants of the blackbox function to assess the quality of the hyperparameter applied settings. Various optimization methods are discussed such as Gridsearch, random search, population-based methods such as evolutionary algorithms, evolutionary strategies and genetic algorithms and Bayesian optimisation.

In summary, while many HPO techniques assume a finite set of components for machine learning pipelines or a finite maximum number of layers in neural networks, there are approaches that allow for arbitrary-sized pipelines. TPOT uses genetic programming and multi-objective optimization to trade off pipeline complexity with performance, while ML-Plan uses hierarchical planning and hierarchical task networks to create pipelines. These approaches have shown competitive performance compared to AutoML systems with fixed pipeline lengths, but more research is needed to determine if larger pipelines can provide further improvements.

## 9. CNN Architectures and Feature Extraction Methods for EEG Imaginary Speech Recognition

[imaginary speech, signal processing, Kara One database, <https://www.mdpi.com/1424-8220/22/13/4679> ]

[9] The article ‘CNN Architectures and Feature Extraction Methods for EEG Imaginary Speech Recognition Al-Jumaily et al. provides a comprehensive review of the applications of artificial intelligence (AI) in the health system focusing on the developments of this rapidly evolving field over time The authors begin with an overview of the historical context of AI in healthcare, including early developments in medical decision-making systems and image analysis. Recent advances in AI technologies, such as deep learning and natural language processing were included, which could have valuable applications in various healthcare domains, including diagnosis, treatment planning, and personalized medicine for patients.

The authors reiterate the need for continued research and development in this field, including the need for interdisciplinary collaboration and the establishment of standards and guidelines for AI applications in healthcare. Their focus was on improving communication for patients who are victims of cerebral stroke, lock-down syndrome, amyotrophic lateral sclerosis, cerebral palsy etc through interpretation of brain activity signals. Their CNN models were based on a multi classification problem with three dataset imagined EEG-speech, biased imagined spoken EEG speech and gated imagined -speech. The most successful model based on its accuracy was the Kara One phonemes. The authors state the potential benefits of AI in healthcare, including improved accuracy and efficiency for care and administration, reduced healthcare financial costs, all of which could lead to better patient care. They also discuss the challenges and limitations of implementing AI in healthcare, such as ethical and regulatory considerations, data privacy concerns, and the need for robust validation and verification of AI models by external parties.

Overall, the authors Anan-Luiza Rusnac and Ovidiu Grigore provide a comprehensive overview of the past, present, and future of AI in healthcare in relation to speech recognition through brain signals using neural networks. The authors highlight the potential benefits and challenges of implementing AI in this domain, and they provide valuable insights into the direction of future research and development in this rapidly evolving field.

## 10. Impact of dataset size and convolutional neural network architecture on transfer learning for carbonate rock classification

[ geological classification, deep learning dataset, Dunham classification Inception v-3 architecture, <https://www.sciencedirect.com/science/article/pii/S0098300422002333>]

[10] The article published in February 2023, Impact of dataset size and convolutional neural network architecture on transfer learning for carbonate rock classification was written by Harriet Dawson , Olivier Dubrule and Cedric M. John was included in Volume 171 of the Computers and Geoscience publication.

This study presents a new approach to carbonate core classification using deep learning algorithms. The aim is to improve overall interpretation accuracy, reduce subjectivity, and interpretation time. The study evaluates the performance of nine different CNN architectures on three complex geological datasets with varying amounts of data, summarized in Table 2. The results show that the Inception-v3 architecture achieved the highest overall accuracy of 92% when trained on the larger dataset. The VGG19 architecture was found to be the most suitable for smaller datasets. The study also found that the size of the dataset played a key role in the performance of the models, with smaller datasets showing a strong tendency to overfit. The study monitored training and validation losses during the deep learning network's training phase. Afterward, the network's prediction performance was evaluated using accuracy, precision, recall, and confusion matrices. Accuracy was measured as the ratio of correctly predicted classifications to the total number of classified samples.

The use of machine learning in geoscience is growing rapidly, but the lack of large, labelled training datasets presents a challenge in improving the prediction performance of CNNs for geological applications. As data availability in geosciences increases, the deep-learning approach presented in this study can be further improved with additional information generated from more recent data and the digitalisation of existing datasets from long-established sources. The framework laid out in this study could aid the future of deep learning-based carbonate classification and can be modified for the classification of cores from different lithologies and formations.

‘

Figure 1 Neural Network architecture

Research Proposal:

The proposal of this report is to perform Sentiment Analysis using Natural language techniques on a large Twitter data using Apache Spark Big Data Storage and Processing on a Linux VM. Time series analysis will be performed on the sentiment data to forecast 1 week, 1 month and 3 months ahead into the future. The results will be displayed on a dynamic, interactive dashboard.

# Introduction

Sentiment Analysis is a popular text analysing technique used for various contextual tasks such as classification, object detection, and segmentation, and with good reason. The dataset is sourced from the Twitter api archive from 2009 and contains 1,048,574 number of tweets over a 3 month period from April to May in 2009. In this research proposal, it is aimed to investigate the predictive performance performance of sentiment analysis using time series techniques on this 2009\_tweet dataset. This is a standard Twitter api dataset of 1,048,475 rows divided into 4 labelled classes below.

* id
* date
* flag
* user
* text

The data is soured from <https://www.kaggle.com>, originally downloaded from the Twitter api before free access was discontinued in February 2023.

# Research Questions

1. How does the performance of different CNN architectures on the CIFAR-10 dataset differ?
2. How does the choice of hyperparameters such as learning rate, batch size, and optimizer choice the performance of CNNs on the CIFAR-10 dataset?
3. What is the performance of different CNN architectures on the CIFAR-10 dataset using big data storage and processing?
4. Can the performance of CNNs on the CIFAR-10 dataset be improved by using big data storage and processing techniques?

# Methodology

Three experiments using various CNN architectures, such as LeNet-5 and AlexNet were conducted on the CIFAR-10 dataset. Three Models were trained on the training set and their performances were evaluated on the validation and test sets, (Model1, Model2 and Model3). The hyperparameters such as learning rate, batch size, and optimizer were adjusted to investigate their effect on the performance of the models.

Initially Apache Spark V3.2.3 was downloaded from the website [11] through a Linux VM employed during the college module Big Data Storage & Processing [12]. Anaconda3-2022.10 was also downloaded [13] and both were launched through a terminal on the virtual machine. The Spark environment variables were added to the baschrc file allowing pyspark to be launched. Hadoop was launched using the standard commands ($start-dfs.sh and $start-yarn.sh) The jupyter notebook ‘Notebook-ca1’ was started automatically with the google chrome browser.

For each model the data was initially split between the training and test dataset with 5000 training samples, and 1000 testing samples. The class vectors were converted to binary class matrices using np\_utils. The categorical variables were converted to one hot encoded vectors, where values were represented as binary vectors. The training and test datasets were normalized by dividing them by 255.The resulting y\_train\_cat and y\_test\_cat variables are matrices with the same number of rows as y\_train and y\_test, respectively, but with 10 columns are each representing a different class. These matrices were used as target variables for the neural network model.

Tuning the hyperparameters

Three models in total were tested, each with various hyperparameter settings. The details of each setting can be seen in Table 1 below.

Three types of Keras optimizers were used including ‘Adam’ and ‘SGD’, and RMS prop. Two types of activation functions were used including the nonlinear ‘ReLU’ in the hidden layers for computational performance, and ‘Softmax’ in the last output layer. The primary role of the Activation Function is to transform the summed weighted input from the node into an output value to be fed to the next hidden layer or as output.  Learning rates were adjusted between 0.01 and 0.001 in order to reassess the model in response to the estimated error for each time the model weights were adjusted. Too large, and the model can converge too quickly to a suboptimal solution, whereas a learning rate that is too small can cause the process to get stuck and hinder it from moving forward. Loss type was retained ‘categorical crossentropy’ and a standard decay rate at 1e-6, since the model is a multi-classification type with more than two labels

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Model 1** | | **Model 2** | **Model 3** | |
| **Type** | sequential | | sequential | sequential | |
| **Architecture** | LeNet-5 | | AlexNet | AlexNet | |
| **Optimizer** | Adam | | SGD | RMS prop | |
| **Activation (hidden layer)** | ‘ReLU’ | | ‘ReLU’ | ‘ReLU’ | |
| **Activation (output layer)** | softmax | | softmax | ‘ReLU’ | |
| **Batch Size** | 32 | | 32 | 128 | |
| **Learning rate (lr)** | 0.001(default) | | 0.01 | 0.001 | |
| **Loss** | categorical crossentropy | | categorical crossentropy | categorical crossentropy | |
| **Dropout Level** | 0.2 | | 0.25 | 0.2 | |
| **Num. epoch's** | 5 | 20 | 20 | 5 | 20 |
| **Final accuracy score** | 68.3% | 70.4% | 76.4% | 77.1% | 83.3% |

Table 1 Model Result Summary

The data was stored using Apache Spark, a big data storage technique, and the models trained using Convolutional neural network, which is a big data processing technique.

Expected Outcomes

It is expected to obtain a comparative analysis of different CNN architectures on the CIFAR-10 dataset using big data storage and processing techniques. The highest performing model with the best hyperparameters and big data storage and processing techniques that can improve the performance of CNNs on the CIFAR-10 dataset should be highlighted. This research can contribute to the development of more accurate and efficient CNN models for image classification tasks using big data storage and processing techniques.

# Results

The model architecture and tuning parameter are tabulated in Table 1. Two types of architecture were used LeNet-5 and AlexNet. The optimizer, type of activation, learning rate and the number of epoch’s each model was cycled for the number of epochs listed.

For Model 1 and 3, it can be clearly demonstrated that the accuracy is directly proportional to the number of epochs.

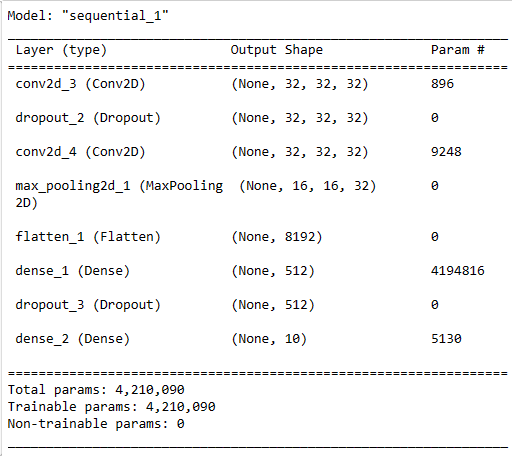


Figure 2 Model 1 Summary

Model 1 summary has image input sizes of 32x32, with 2 hidden layers. The added dimension from Keras creates the final input shape of (32,32,32). After convolving the 32x32 image for the first time, where ‘None’ is specified as the batch size, image size remains at 32,32 with a 32 filters and ‘same’ padding. Kernal constraint maxnorm was set to 3. Ouput shape continues at 32x32.Max pooling layer took the input image of size 32,32 and reduced it by half, to 16,16. This pattern was applied to both hidden Conv2D and max pooling layer. The flatten layer took the pixels along the channel and created a 1D vector flattening the input from (16,16,32) to 8192. The Dense layer is reduced from 512 in the first layer to 10 after the second layer. The final output shape of Model 1 is (None, 10), ie. 10 values per sample in the batch. The model has a total of 4,210,090 weights, all of which are trainable, and 0 untrainable weights. Model 1 used an ‘Adam’

optimizer. Rectified Linear Unit (ReLU) was used an activation for the hidden layers, and ‘Softmax’ for the output layer.



Figure 3 Model 1 Accuracy Result after 5 epochs

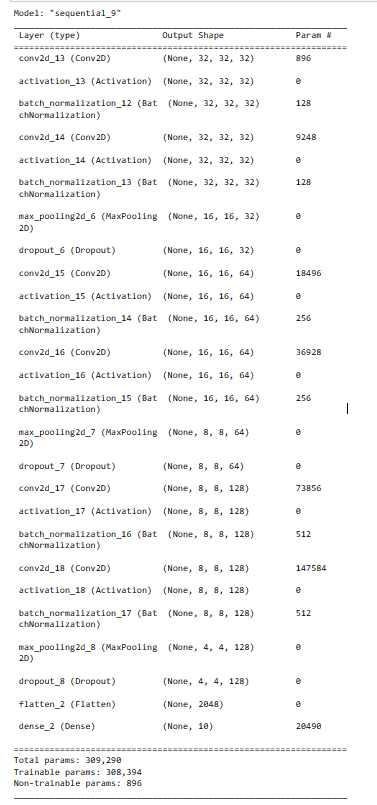
Model 1, produces a final test set accuracy of 68.26%. This increased to 70.38% after 20 epochs, indicating the learning rate is appropriate. The model weights are changing due to the high learning rate and moving away from the local minimum where accuracy would typically be highest.



Figure 4 Model 1 Accuracy Result after 20 epochs

Model 2

Model 2 has 5 hidden layers, also with input image sizes of (32,32,32). After the fifth convolving, the output image size is (8,8,128). Max pooling reduces this to (4,4,128). The Dense layer is reduced to 10, in the final layer. The output shape of Model 2 is identical to Model 1, (0,10). The model has a total of 308,395 trainable weights, and 896 untrainable weights. Model 2, similar to Model 1 used a Rectified Linear Unit (ReLU) as the activation for the hidden layer, and ‘Softmax’ for the output layer. Stochastic gradient descent, (SGD) was used as the optimizer.



After 5 epochs Model 2 produced an accuracy of 76.44%

Figure 5 Model 2 Accuracy Result after 20 epochs

Model 3

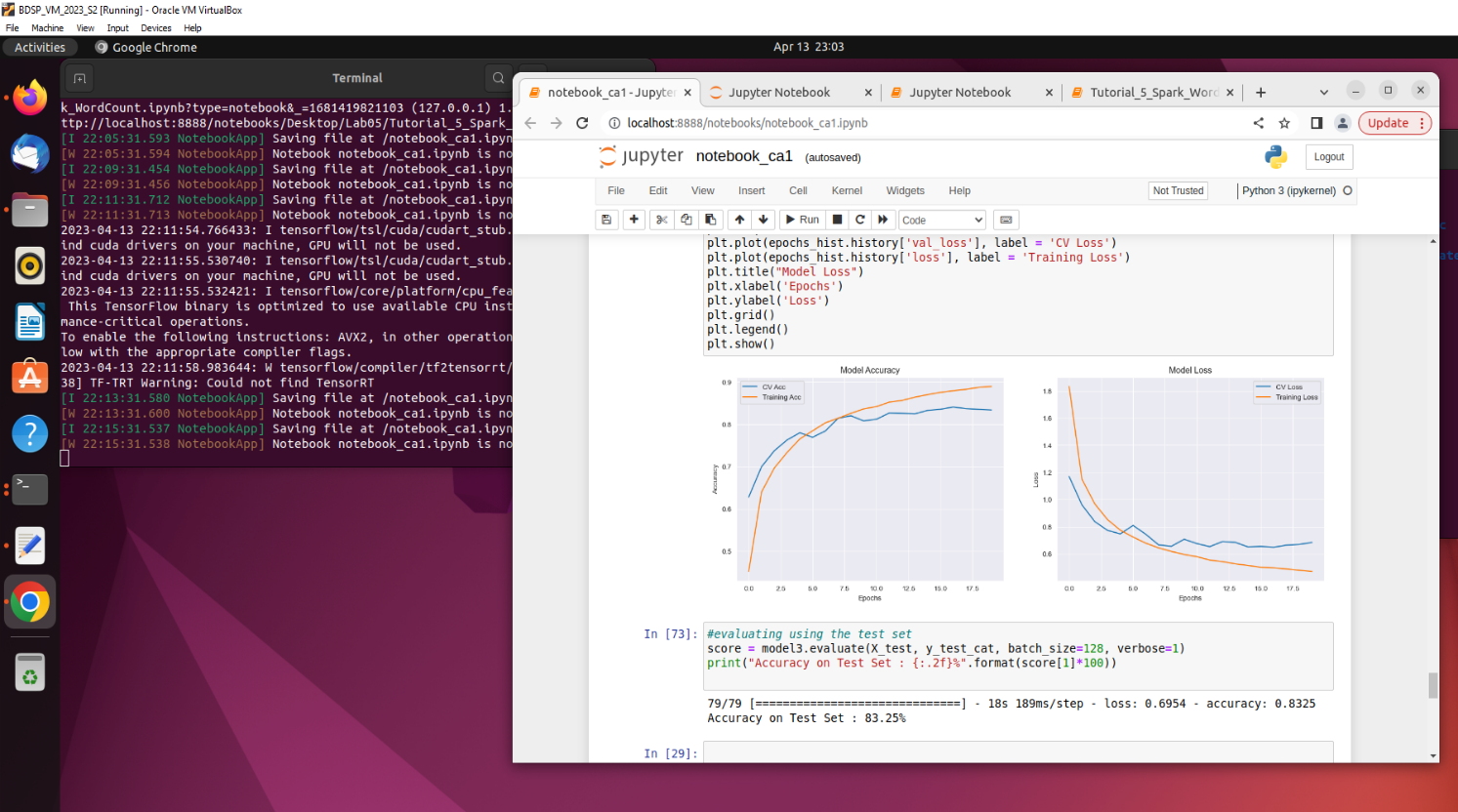
Model 3 has 5 hidden layers, with input images sizes of (32,32,32). At the final fifth convolving, the output image size is (8,8,128), similar to Model 2. The model has a today of 308,394 trainable weights, and 896 untrainable weights. ReLU was used as the activation for both the hidden and output layer. RMSprop algorithm was used at the optimizer with a learning rate of 0.001 and a decay of 1e-6.

Figure 6 Model 3 plot on Linux VM

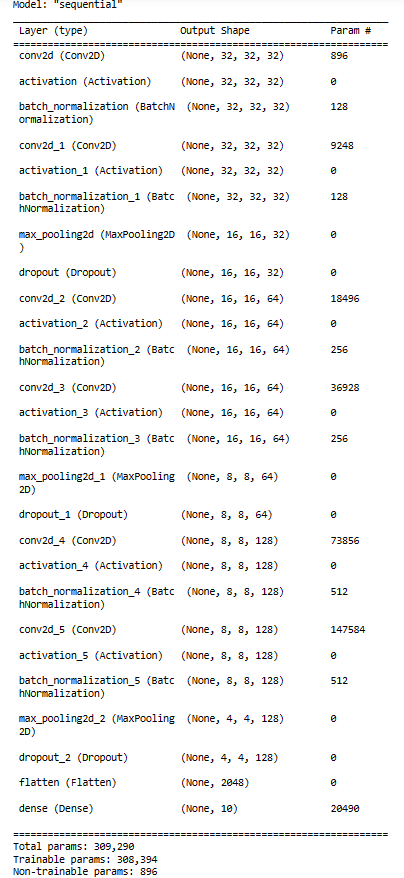


Figure 7 Model 3 Summary

Plotting the Accuracy Curve and the Loss Curve gives a snapshot of the training process and the direction in which this network learns. Losses were logged after every epoch (5 in this case, due to processing constraints) for the test data set, X\_test and y\_test set. The test set accuracy is 77.1% after 5 epochs with a batch size of 128.

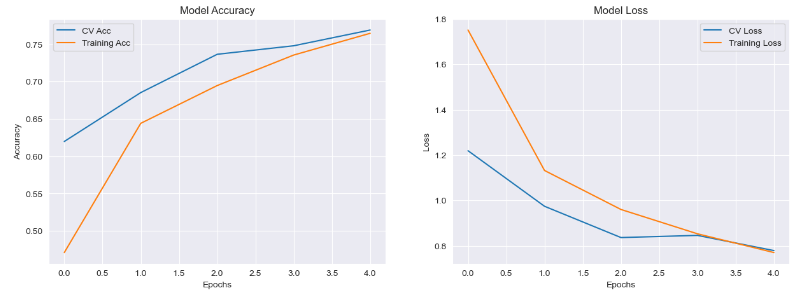
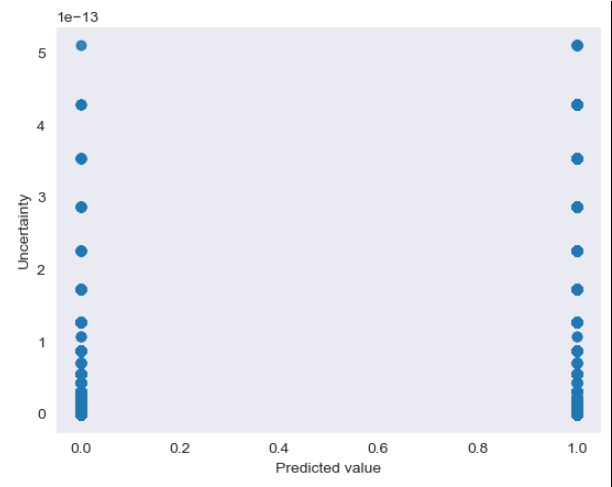
Figure 9 Model 3 Accuracy Results after 5 epochs

Figure 8 Model 3 Accuracy vs Epochs (5) for cross validation and training accuracy

Figure 10 Model 3 Uncertainty vs. predicted values



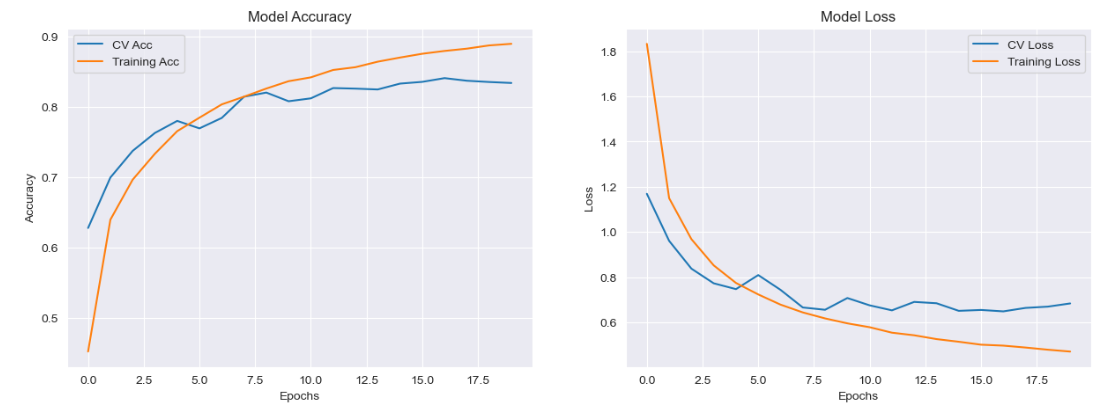
Plotting the uncertainty against 50 predicted y values produces the scatter plot in figure 10, did not provide a clear picture of whether the model is becoming more certain with further training during the epochs. Ideally we would like to see uncertainty decreasing with training. This could be due to the limited number of epochs executed (5).

Figure 11 Model 3 Accuracy vs Epochs (20) for cross validation and training accuracy

After 20 epochs, this accuracy increased to 83.25%.

Figure 12 Model 3 Accuracy Results after 20 epochs

# Conclusion

In this research proposal, it was aimed to investigate several research questions outlined at the beginning of the report. The first was how the performance of CNN architectures differ on the performance of CIFAR-10 dataset. Referring to table 1, the most successful architecture was AlexNet in Model3 and Model2 who reached accuracies of 83.3% and 76.4% respectfully. Question 2,the choice of hyperparameters would appear to have a significant effect on the performance of the CNN. The lower learning rate of 0.001 was successful in Model 3 with ‘Adam’ type optimizer and a batch size of 128. The batch size was reduced to 32 for Model1, but the learning rate and optimizer type were identical suggesting

the combination of certain parameters is important, not just one hyperparameter.

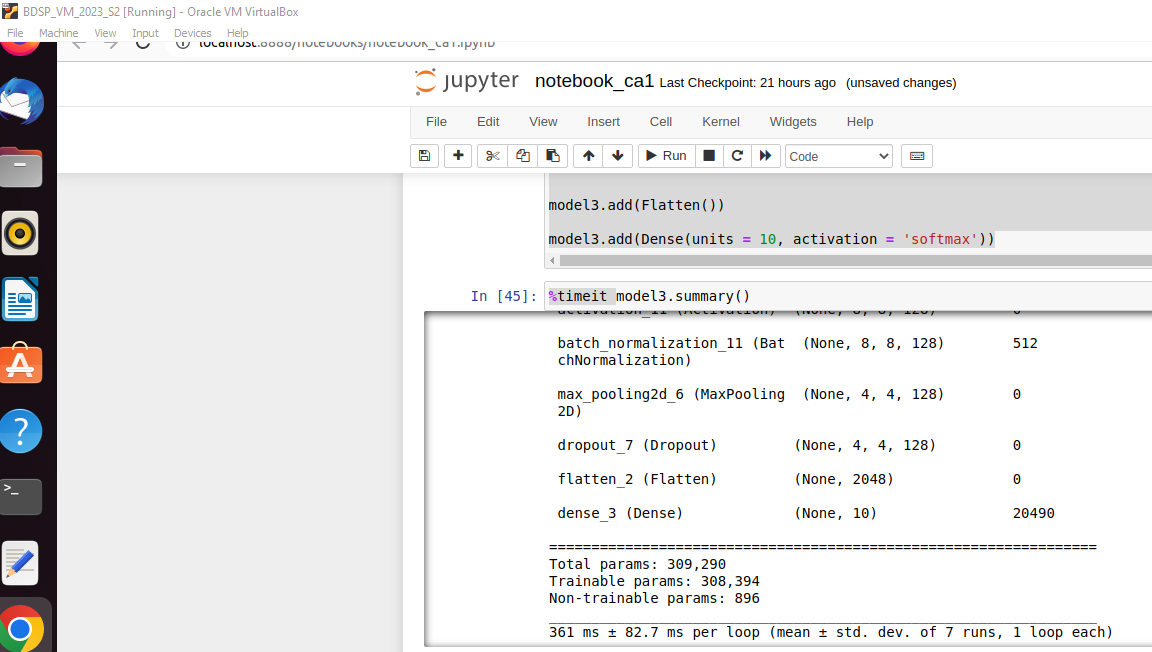
Question 3 was how the performance of different CNN architectures differ using big data storage and processing techniques. In order to evaluate this question the magic python command ‘timeit’ was employed to measure code running time between a juptyer notebook running on a local pc and compare the same code running through Apache Spark on the linux VM.Looking at Model 3 on the local machine in figure 14, the model summary takes 125ms to run/loop

Figure 13 Model 3 Summary Runtime VM

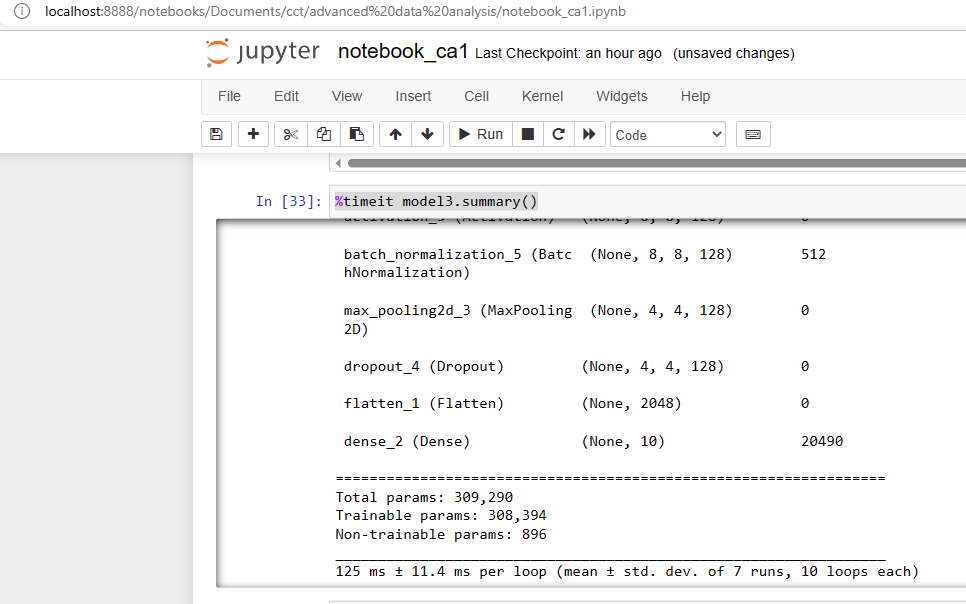


Figure 14 Model 3 Summary Runtime Local

While on the VM, it can be seen that the same section of code takes 361ms (figure 13) to run per loop. These timing are heavily based on the pc capacity and hard drive availability at the time of execution.

Similarly, compiling time for Model3 is 5.17ms on the local in figure 15, while it take 361ms to run on the VM. To improve this sharding would be one viable option, where Tensorflow would take every nth element of the dataset. Another option would be, that when fitting is performed Tensorflow could be instructed to use just 1/nth of the total batches for training. This would reduce training and testing time with less hardware capabilities [13]. Using big data storage generally allows data training to be done more effectively and quickly, since CNN’s have a hard time identifying images with different orientations and positions in image recognition.

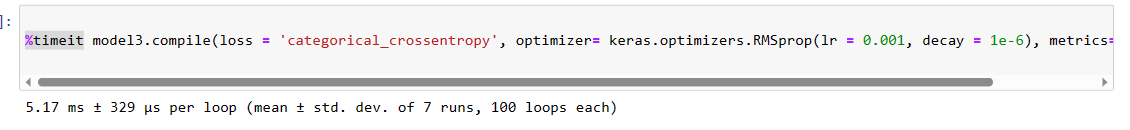


Figure 15 Model 3 Compile Runtime Local

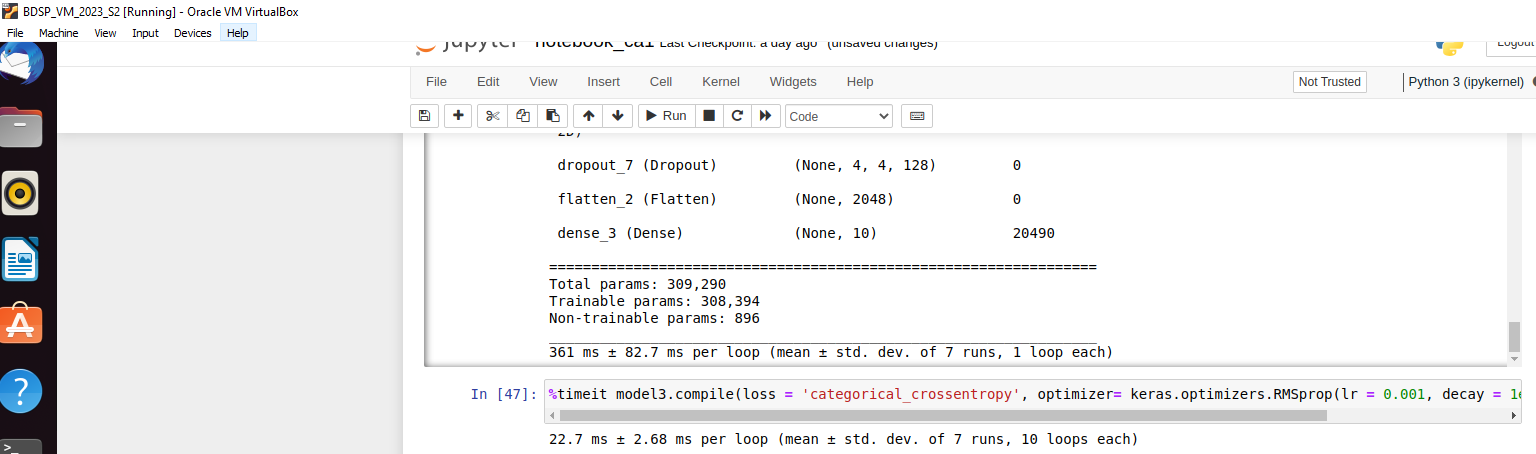


Figure 16 Model 3 Compile Runtime VM

Fitting Model3, with a reduced figure of 1 epoch took 9min, 48 seconds on the local machine in figure 17, and 10min, 17 seconds on the virtual machine in figure 18.

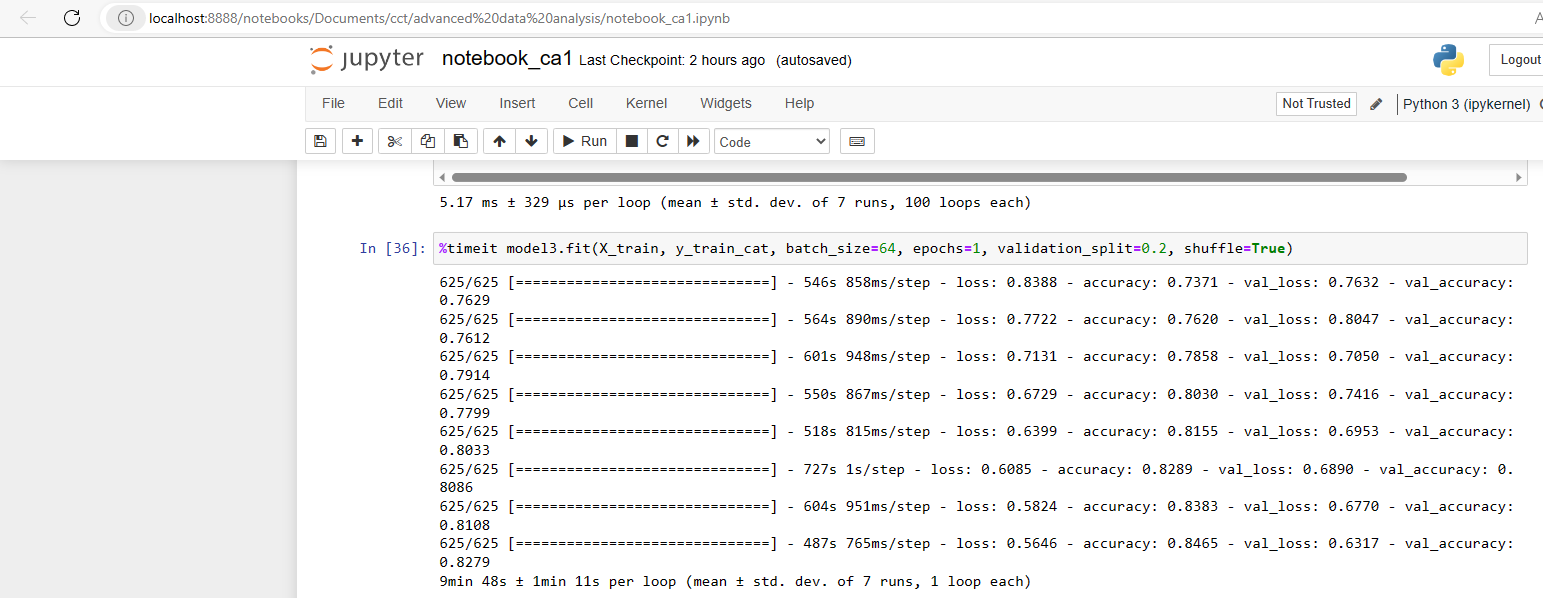


Figure 17 Model 3 Fit Runtime Local

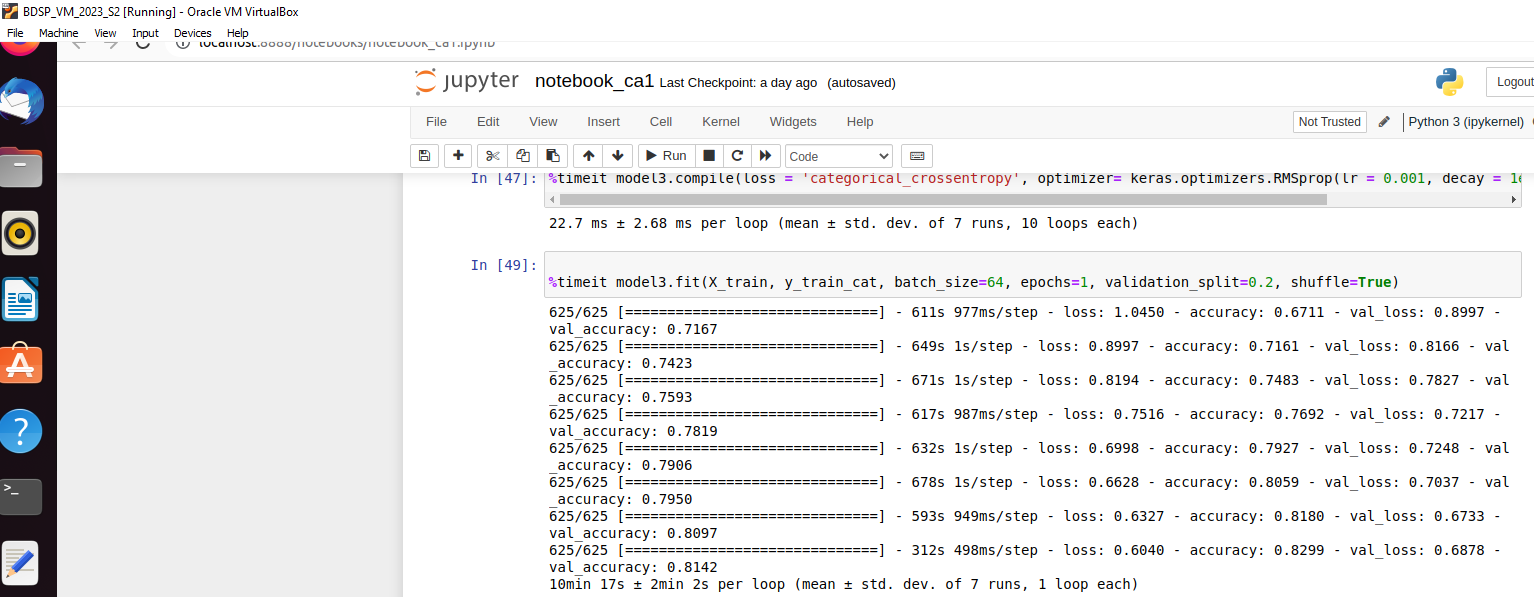


Figure 18 Model 3 Fit Runtime VM

By conducting experiments using various CNN architectures and hyperparameters, the aim was to identify the best models and techniques for achieving high accuracy on the dataset while utilizing big data storage and processing. Model 3 with 20 epochs, was identified as performing the best with a model accuracy of 83.3% after 20 epochs. This research can contribute to the development of more accurate and efficient CNN models for image classification tasks in the big data area by its use of a standard dataset for comparative purposes.

# References

1. Adhikari, T. (2023). *Designing a Convolutional Neural Network for Image Recognition: A Comparative Study of Different Architectures and Training Techniques*. [online] papers.ssrn.com. Available at: <https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4366645> [Accessed 15 Apr. 2023].
2. Agarwal, B. (2022). *How to Reduce the Training Time of Your Neural Network from Hours to Minutes*. [online] Medium. Available at: <https://towardsdatascience.com/how-to-reduce-the-training-time-of-your-neural-network-from-hours-to-minutes-fe7533a3eec5>.
3. Al-Waisy, A.S., Qahwaji, R., Ipson, S., Al-Fahdawi, S. and Nagem, T.A.M. (2017). A multi-biometric iris recognition system based on a deep learning approach. *Pattern Analysis and Applications*, 21(3), pp.783–802. doi:<https://doi.org/10.1007/s10044-017-0656-1>.
4. Dawson, H.L., Dubrule, O. and John, C.M. (2023). Impact of dataset size and convolutional neural network architecture on transfer learning for carbonate rock classification. *Computers & Geosciences*, [online] 171, p.105284. doi:<https://doi.org/10.1016/j.cageo.2022.105284>.
5. Fan, X., Feng, X., Dong, Y. and Hou, H. (2022). COVID-19 CT image recognition algorithm based on transformer and CNN. *Displays*, [online] 72, p.102150. doi:<https://doi.org/10.1016/j.displa.2022.102150>.
6. Feurer, M. and Hutter, F. (2019). Hyperparameter Optimization. *Automated Machine Learning*, pp.3–33. doi:<https://doi.org/10.1007/978-3-030-05318-5_1>.
7. Jan, B., Farman, H., Khan, M., Imran, M., Islam, I.U., Ahmad, A., Ali, S. and Jeon, G. (2019). Deep learning in big data Analytics: A comparative study. *Computers & Electrical Engineering*, 75, pp.275–287. doi:<https://doi.org/10.1016/j.compeleceng.2017.12.009>.
8. Jason Brownlee (2019). *Understand the Impact of Learning Rate on Neural Network Performance*. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/>.
9. Jose, G.V. (2019). *Useful Plots to Diagnose your Neural Network*. [online] Medium. Available at: <https://towardsdatascience.com/useful-plots-to-diagnose-your-neural-network-521907fa2f45>.
10. Kaur, G., Sinha, R., Tiwari, P.K., Yadav, S.K., Pandey, P., Raj, R., Vashisth, A. and Rakhra, M. (2022). Face mask recognition system using CNN model. *Neuroscience Informatics*, [online] 2(3), p.100035. doi:<https://doi.org/10.1016/j.neuri.2021.100035>
11. <https://downloads.apache.org/spark/spark-3.2.3/spark-3.2.3-bin-hadoop3.2.tgz>
12. Big Data Storage & Processing, Msc Data Analysis, Dr. Muhammad Iqbal, CCT College
13. Li, X., Zhang, W. and Ding, Q. (2019). Deep learning-based remaining useful life estimation of bearings using multi-scale feature extraction. *Reliability Engineering & System Safety*, 182, pp.208–218. doi:<https://doi.org/10.1016/j.ress.2018.11.011>.
14. Najafabadi, M.M., Villanustre, F., Khoshgoftaar, T.M., Seliya, N., Wald, R. and Muharemagic, E. (2015). Deep learning applications and challenges in big data analytics. *Journal of Big Data*, 2(1). doi:<https://doi.org/10.1186/s40537-014-0007-7>.
15. Naseer, I., Akram, S., Masood, T., Jaffar, A., Khan, M.A. and Mosavi, A. (2022). Performance Analysis of State-of-the-Art CNN Architectures for LUNA16. *Sensors*, [online] 22(12), p.4426. doi:<https://doi.org/10.3390/s22124426>.
16. Rusnac, A.-L. and Grigore, O. (2022). CNN Architectures and Feature Extraction Methods for EEG Imaginary Speech Recognition. *Sensors*, 22(13), p.4679. doi:<https://doi.org/10.3390/s22134679>.
17. Usama, M., Ahmad, B., Wan, J., Hossain, M.S., Alhamid, M.F. and Hossain, M.A. (2018). Deep Feature Learning for Disease Risk Assessment Based on Convolutional Neural Network With Intra-Layer Recurrent Connection by Using Hospital Big Data. *IEEE Access*, 6, pp.67927–67939. doi:<https://doi.org/10.1109/access.2018.2879158>.
18. Uzila, A. (2022). *5 Popular CNN Architectures Clearly Explained and Visualized*. [online] Medium. Available at: <https://towardsdatascience.com/5-most-well-known-cnn-architectures-visualized-af76f1f0065e>.