*Evaluation of Convultional Neural Networks for Image Classification using the Fashion MNIST Dataset*

*Abstract*—Convolutional Neural Networks are one of the leading deep learning methods for image classification today. This paper reviews deep learning with a focus on CNN models and literature review of CNN model and fashion MNIST datasets. The research focuses on implementing five CNN models on the fashion MNIST dataset with an effort to improve accuracy and loss scores. The best performing model for accuracy was CNN\_Model\_Four with a 91.49% accuracy rate. The best performing model in terms of loss was CNN\_Model\_One at 28.34%. This research attempted the implementation of Apache Spark processing, however was unsuccessful.

Keywords—Neural Network, CNN, Apache, Big Data, Hadoop

# Introduction

Machine learning (ML) is a form of artificial intelligence that utilizes algorithms and statistical models to train a computer. Deep learning is a subset of ML which applies neural networks to explore patterns and derive relationships within complex datasets. It closely resembles how the human brain learns and is utilized in tools we use every day such as speech recognition and natural language processing. Neural network algorithms are based on the human brain and how it learns. to machine learning algorithms but contain many more layers which allow for the analysis of complex datasets.(Umer, 2022; Taye, 2023)

Image classification is one of the many challenges that deep learning tools have been employed for particularly within medical imaging to identify diseases. Another industry is retail, where neural networks are utilized to categorize products in online stores in order to increase sales and provide personalized profiles for customers. For example, ASOS a large online retailer could have almost 5000 new products per week in over 100 categories. By applying artificial intelligence technologies, the retailers can present each customer products that they are more likely to click on and buy based on previous orders and wish list likes.(Cardoso, Daolio and Vargas, 2018)

The main types of neural networks are Artificial Neural Networks (ANN), Convolution Neural Networks (CNN), Recurrent Neural Networks (RNN), Perceptron, Long Short-Term Memory Networks and Radial Basis Functional Neural Networks. (Pai, 2023)

The ANN mimics a human brain’s neural structure. It is comprised of three layers, input, hidden and output, which are interconnected through nodes. ANNs are suitable for numeric and structured data and can be used for image analysis or forecasting. The main disadvantage of an ANN is the transparency of the network operation and the inability for it to assess sequential information.(Pai, 2023; Singh, 2023)

In order to overcome the limitation of ANNs for assessing sequential information, RNNs and CNNs were developed. RNNs has an architecture similar to an ANN however it contains a recurrent connection on the hidden layer that allows for sequential information to be processed. RNNs are commonly used for time series, text and audio data. CNNs are arguably the most popular neural networks in literature today, especially with image and video analysis. These networks can process sequential information unlike the ANNs. (Pai, 2023)

Perceptron is single layer neural network that act as binary classifiers. The model works with linearly separable data which is also its main limitation. Each input is connected to an output and a weight is applied to each input based on the significance. (Pai, 2023)

LSTM networks are a class of RNNs with the ability to process sequential information but to also retain information through the addition of a memory cell with three gats, input forget and output. The application of LSTMs includes speech recognition, natural language processing and time series. (Pai, 2023)

Radial Basis Functional (RBF) neural networks are a class of ANNs which are used for pattern recognition, function approximation and time series. They contain three layers, input hidden and output. The main issues with RBFs are around the basis functions for selection and quantity and in addition are likely to overfit when trained. (Pai, 2023)

The aim of this research paper is to compare variations of a convolutional neural network model due to their popularity in image classification and to assess the changes in accuracy and loss on the fashion-MNIST data.

## Convolutional Neural Networks

Convolutional Neural Networks, a feed forward neural network, is the main method used for image classification. CNNs mimic the way the human eyes and brain works, by taking small subsections of an image and analyzing them. There are three main layers that occur with a CNN, the convolution layer, the pooling layer and the fully connected layer. The convolution layer essentially extracts the features of the image. A filter size and stride size are defined and used to divide up the image into a number or pixels. Next is the pooling layer which takes the output of the convolution layer and reduces the dimensionality again where the average or max values from the results are taken. Then the fully connected layer, where the neurons in each layer are taken and connected to the subsequent layers. This layer then learns how to classify particular objects (Lang, 2021) (Li *et al.*, 2022) (Taye, 2023)

An activation function forms part of the neural network that determines what information is communicated to the next neuron. Activation functions common within neural networks are linear, sigmoid, tanh, ReLu and swish. Li *et al.*, 2022 evaluated the activation functions on the fashion MNIST data and concluded that the linear activation performs the worst as expected with a multi-layer model and that ReLU performed the best overall with relation to accuracy, training time and stability. ReLU is the most commonly used activation function as due to time and resource savings, simpler gradient definitions and sparser representation. (Li *et al.*, 2022) (Taye, 2023)

Loss functions are a measurement that calculates the distance between the predicted and actual value. Models are optimized to reduce the value of the loss function. For classification, cross entropy is the most commonly used which takes the predicted probability and the output value and the distance between them is used to calculate the penalty value. In CNNs, it is incorporated in a SoftMax layer. Other loss functions that have been introduced to account for the cross entropy disadvantages include contrastive loss, triplet loss, center loss and the large margin SoftMax.(Li *et al.*, 2022)

Optimizers are functions that are used to reduce loss functions in neural networks. Gradient descent optimizers work to convert the model to a set of parameters that minimizes the loss on training. Examples of optimizers include stochastic gradient descent, Adam, and RMSprop. Each optimizer performs to its own strength and weakens and should be decided based on data distribution, computing cost and accuracy. (Li *et al.*, 2022)

Hyperparameter tuning is a crucial part of any model’s development and is no different for CNNs in order to achieve the best performance. For CNNs the main hyperparameters to be tuned include, learning rate, epoch, min-batch size, number of layers and kernels and size of kernels. (Li *et al.*, 2022)

## Big Data Storage and Processing

Two concerns with deep learning architectures are the high computational cost and memory requirement which is limitations of deep CNN models mainly due to the high multitude of multiplication that occur at the convolution operation level. Due to this, an important step is the selection of a suitable big data storage and processing solution. (Khan *et al.*, 2020)

Nowadays, the quantity of data being generated and consumed is continuously increasing which demands technologies that can store more data and also process data faster. One of the most common ways that data has been stored is with relational databases. In relational databases, the data is structured based on a pre-defined type and format and can be accessed easily and quickly. While these databases have benefits such as predictability, easy user interaction, and easy retrieval, they have high set-up costs, difficult to scale and difficult to incorporate unstructured data. Examples of relational databases utilized today are MySQL and Oracle.(Jatana *et al.*, 2012; Ergüzen and ünver, 2018)

In order to overcome the challenges of relational databases, non-relational systems were established. Non-relational systems don’t utilize SQL and are often referred to as NoSQL databases. These were designed to offer high performance, availability, and scalability. However, these benefits are at the cost of losing ACID (atomic, consistent, isolated, durable) attributes that come with the relational databases. Instead, they contain the BASE (basic availability, soft state, eventual consistency) attributes. (Tudorica and Bucur, 2011)

Distributed file systems (DFSs) are a type of non-relational system that is distributed across multiple servers or locations. The main attributes of a DFSs are transparency, fault tolerance and scalability. (Depardon *et al.*, 2013) A commonly used solution for big data storage is Hadoop, an open-source software framework that is comprised of two layers, the storage layer which is called Hadoop Distributed File System (HDFS) and second layer which is processing layer called MapReduce. Hadoop is scalable, fault tolerant, cost-effective and supports unstructured data. The main limitation is the slow processing speed. (Dwivedi and Dubey, 2014; Ghazi and Gangodkar, 2015)

HDFS is targeted towards batch processing. The architecture behind HDFS is a Name Node, where the metadata is managed and Data Node, where the data is stored. The Name Node is broken down into blocks of data that get distributed to multiple Data Nodes, and often blocks are replicated across nodes as system backup. MapReduce is a programming model which writes to applications and is capable of parallel processing. There are two phases, the map phase whose input arises from the HDFS and the reduce phase whose input is the map phase output. In addition, MapReduce utilized Job Tracker daemon and TaskTracker daemon similar to the master/slave architecture. (Dwivedi and Dubey, 2014; Ghazi and Gangodkar, 2015)

Apache spark is a data processing platform that implements a hybrid framework that can support batch and real-time processing. Apache Sparks architecture is made up of driver program, cluster manage and slave nodes. The driver program is the master node and entry point for the application. The cluster manager is responsible for resourcing, splitting the jobs into the slave node clusters which they execute. Resilient Distributed Datasets (RDDs) are a core structure to Apache Spark which supports in-memory processing providing a fault tolerant framework. As a result of using RDDs, Apache Spark is a hundred times faster than Hadoop. Additional benefits of Apache Spark include use with multiple programming language, and real time processing. (Shaikh *et al.*, 2019)

Apache pig is another platform that can be used with Hadoop. It is suitable for large dataset and processes the MapReduce steps. For execution there are two environments, a local for smaller datasets and a distributed for the Hadoop cluster. Pigs’ architecture is a pig Latin script interpreter that sits above the map reduce and HDFS systems. The pig Latin script interpreter can be run through either a grunt shell or pig server, which then feeds into a parser, optimizer, compiler, and execution engine. By the end the script is converted into MapReduce jobs and is forward processed. Benefits of Pig include being able to process all data structures, easy to use and supports optimization. (Arora *et al.*, 2019)

Apache hive was established as a result of the growing platform, Facebook, and its need to process big data. Hive is a warehouse and extract transform load tool with a SQL interface that is built on top of Hadoop. Hive language is hive query language (HQL) and is similar to the well-established SQL. For execution, hive can utilize MapReduce or spark engines. Hive is structured with three layers, the hive client, hive services and Hadoop. Hive is beneficial in that it is an easy tool to summarize and analyze data and supports ad-hoc queries. In terms of Hives limitations, there is a high amount of latency, doesn’t support online transaction processing and is not suitable for real-time data. (Arora *et al.*, 2019)

Apache Cassandra is a decentralized distributed database that supports the processing of structured big data with no downtime. The data is structured in a Key Space which contains column families that are a collection of rows of data. Cassandras architecture is comprised of many nodes in cyclical nature. This structure gives Cassandra scalability, elasticity, availability, and replication. One of the main appeals of Cassandra is the zero-downtime due each node being replicated. (Wahid and Kashyap, no date)

# Related Works

Kadam *et al.* (2020), proposed five different CNN architectures for image classification. The architectures were varied through activation methods, dropout, learning rate, batch size and layers. A testing accuracy of 99.55% was obtained for the MNIST dataset and 93.56% for the Fashion MINST dataset.(Kadam, Adamuthe and Patil, 2020)

Kayed *et al.* (2020) proposed a CNN model with a LeNet-5 architecture that obtained an accuracy over 98%. The architecture contains five layers which are a combination of convolutional layers with 5x5 filters and pooling layers with 2x2 and a stride of 2.

Kayed *et al.* (2020) demonstrated image classification with machine learning and deep learning models, where the SVC reached a test accuracy of 89.70% and the deep learning CNN models reached 98.80%. (Kayed, Anter and Mohamed, 2020)

Sharma *et al.* (2018) assessed the performance of the most popular CNN models, Alex Nets, GoogLeNet and ResNet50 across various image data sets for object detection in real world scenes. The objective was to assess the accuracy and prediction consistency of each CNN. It was concludes that the higher number of layers were favorable. (Sharma, Jain and Mishra, 2018)

Nocentini *et al.* (2022) proposed four different CNN models for image classification using the Fashion MNIST dataset. The models were varied and tuned with respect to batch size, kernel size, number of filters and fully connected layers. They obtained an accuracy of 94.09% with their MCNN15 model. (Nocentini *et al.*, 2022)

Henrique *et al.* (2021) analyzed CNN models for the fashion MNIST data that varied in the number of layers and dropout values. It was concluded that the lower dropout rates and increased layers gave the best accuracy value of 99.1%. (Henrique *et al.*, 2021a)

Hedjazi *et al* (2018) compared Hadoop, Spark and Storm frameworks for distributed image processing. Through their research, they discovered that when comparing Hadoop and Spark that for non-iterative tasks Hadoop performs better, but for iterative tasks Spark performed better. Spark and Storm were then compared and it was concluded that Storm had better latency but overall both performed well for the tasks. (Hedjazi *et al.*, 2018)

Sunil *et al.* (2022), proposed a comparative study of evaluating CNN models on the fashion MNIST and MNIST datasets with and without Apache Spark. The models retrieved an accuracy of 88.65% and 98.61% for the datasets respectively. When the performance of the CPU was measured with and without spark, it was observed there was about 13.3% and 3.81% difference in performance, concluding that it was more beneficial to execute the models in Apache Spark to reduce the computational load. (Sunil and Sivagamasundari, 2022)

# Methodology

## Data Set and Processing

The fashion MNIST dataset comprises of 10 classes and 70,000 images of 28 x 28 size. This dataset was created to replace the original MNIST dataset to provide more challenge when benchmarking machine learning algorithms. For this research, the fashion MNIST dataset was imported using the TensorFlow built in keras datasets. (Science, 2021)

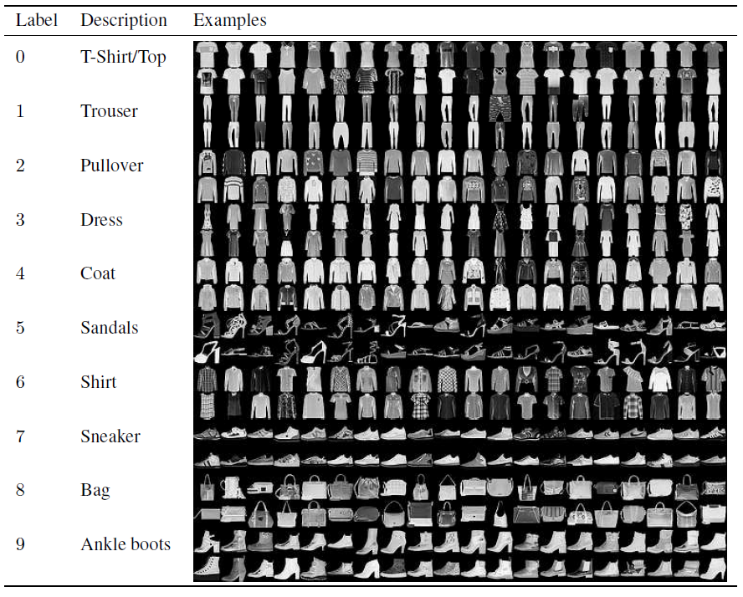


Figure 1: Sample of the Fashion MNIST Data (Henrique et al., 2021b)

## CNN Models

For this evaluation five different CNN models were created to classify the fashion MNIST data.

The first model, CNN\_Model\_One, was created as a baseline model. Three convolutional layers were included with increasing filters of 32, 64, and 128 on each layer. The size of the convolutional filters was set to (3,3) which is common and suitable for smaller image sizes like the fashion MNIST data. The padding is set to ‘same’ to ensure that the dimensions of the input and output are the same. The activation function chosen was ‘ReLu’ due to the benefits discussed above. For the model optimizer, ‘adam’ was chosen due to its efficiency, robustness and adaptive learning rate mechanism. The epoch chosen was set to 10, which is how much the data is inputted for training. (Li *et al.*, 2022)

The second model, CNN\_Model\_One-Epoch Increase, is the same as CNN\_Model\_One except the epoch value was increased to 30 to determine the effect. It can be an important value to determine as too low can result in underfitting and too high will result in overfitting. (Li *et al.*, 2022)

The third model, CNN\_Model\_Two, was built the same as CNN\_Model\_One but with five extra fully connected layers. By increasing the layers in the model it may allow the network to capture more features, potentially improving the accuracy and loss values. (Pylypenko, 2019)

The fourth model, CNN\_Model\_Three, was created as CNN\_Model\_Two however a regularization technique, dropout, was applied after each layer. With each dropout layer, the model randomly ‘drops’’ a fraction of neurons to during training so that it is not reliant on one particular section. The goal of adding the dropout would be to reduce or prevent overfitting. (Bhobe, 2019)

The fifth and final model, CNN\_Model\_Four, was created similarly to CNN\_Model\_Two with batch normalization added after each convolutional layer prior to pooling and a dropout included at the fully connected layer. Batch normalization allows each of the layers to work slightly more independently. Batch normalization was introduced to increase learning rate while also reducing overfitting. (Dwivedi, 2020)

## Evaluation Metrics

To evaluate the performance of each model, ‘sparse categorical accuracy’ and ‘sparse categorical loss’ was chosen. The sparse categorical method was chosen as this is foremost a classification problem and the labels provided were integers. The model was trained with the designed training dataset and a subset of 33% was used for validation in order to detect and visualize overfitting. Once the model was trained, it was evaluated against the testing dataset to calculate the accuracy and loss values.

Loss functions are a key component of machine learning for assessment. The loss function looks at how the neural network algorithm is actually modelling your data. For the function it looks at the predicted and the actual values and calculates the error. It is an important measurement when optimizing the model in training. There are many different types of loss functions, and these are chosen based on the specific problem statement. For classification, there is binary or categorical cross entropy. For the categorical cross entropy, this is utilized with the SoftMax activation function which converts the output predictions to probabilities so that it can be compared to the true labels. When optimizing a neural network, the aim is to reduce the value of the loss function. (Shankar, 2023)

## Apache Spark

One of the objectives of this research was to implement a CNN model in Apache Spark. Spark was chosen due to its ability for distributed data processing.

Pyspark was initially installed and set-up within the Oracle Virtual Machine to support memory requirements. While the installation completed, the execution of the code in Jupyter notebook in the VM proved challenging due to issues with Java when initializing the pyspark session. After extensive troubleshooting and no success, it was decided to set-up pyspark locally.

When using Apache Spark for deep learning tasks, the code within the Jupyter notebook needs to be altered. The process starts by importing the libraries and images. The spark application is initialized by a spark session which is the entry point. For spark, the images are converted to a spark dataframe to transform it to structured data format. A pipeline is then created to take the data and pre-process it by extracting, transforming and feature selection. The model can then be built and defined with Kera’s and tensorflow. The model then needs to be run on Spark to take advantage of the distributed engine. A library that can do this is Elephas. The deep learning pipeline is than ran to train, test and evaluate the model.(Vasquez, 2018; Violante, 2019)

# Results And Discussion

For each model the loss and accuracy result along with training time is detailed in Table 1. The aim of this research was assessing the changes in accuracy and loss based on the changes.

Table 1: Loss and Accuracy Results from CNN Models developed for this research.

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Test Accuracy (%) | Test Loss (%) | Training Time (secs) |
| CNN\_Model\_One | 91.24 | 28.34 | 606.90 |
| CNN\_Model\_One-Epoch Increase | 90.94 | 83.57 | 1752.41 |
| CNN\_Model\_Two | 91.04 | 32.64 | 607.89 |
| CNN\_Model\_Three | 89.94 | 29.54 | 656.97 |
| CNN\_Model\_Four | 91.49 | 32.50 | 973.83 |

The CNN\_Model\_One which is considered the baseline model gave a good accuracy and loss without any parameter tuning. From the training and validation results seen in Figure 2, it is clear there is slight overfitting as the validation accuracy sits below the training.

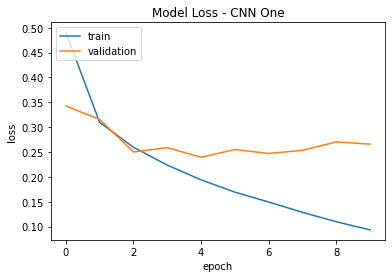
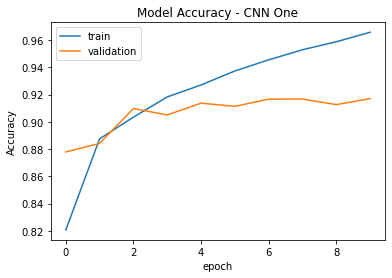


Figure 2: Model Accuracy and Loss for CNN\_Model\_One

When the epochs were increased to 30 the accuracy decreased, and the loss significantly increased. Therefore, for this classification problem a higher epoch is not suitable. In addition, the training time was also doubled, so the model became less efficient overall. For future work, it would be interesting to determine the effect of reducing the epoch value.

A graph showing the growth of a train

Description automatically generated

Figure 3: Sparse Categorical Loss for the CNN\_Model\_One-Epoch Increase

For the third model, CNN\_Model\_Two, the increase in the fully connected layers did not give the desired effect of improving the accuracy and loss values. In fact, they both were slightly worse than the first model, with values of 91.04% for accuracy and 32.64% for loss. From Figure 4, it can be seen that the model was slightly less overfitted than the first model, however this benefit is not a strong enough to account for the change in testing accuracy and loss values.

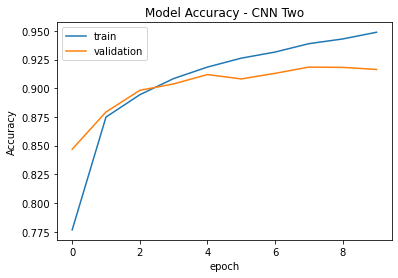
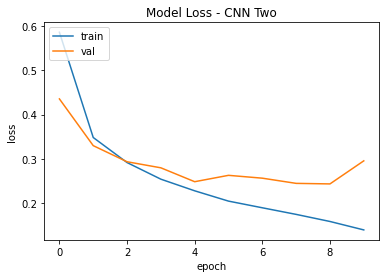


Figure 4: Model Accuracy and Loss for CNN\_Model\_Two

The fourth model, CNN\_Model\_Three, had the addition of dropouts at each layer. When compared to CNN\_Model\_Two, the accuracy and loss values worsened again with this parameter change. The aim of adding the dropout levels was to decrease overfitting. As per Figure 5, there is no visual improvement in the overfitting when compared to CNN\_Model\_Two. As the accuracy and loss worsened, the addition of these dropout layers was not beneficial. For future work, the number of dropout layers could be modified to determine the effects and also the size of the dropouts.

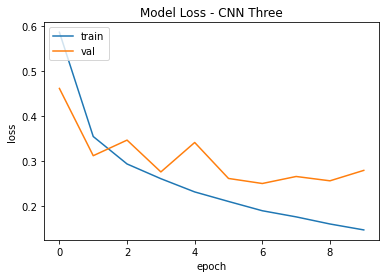
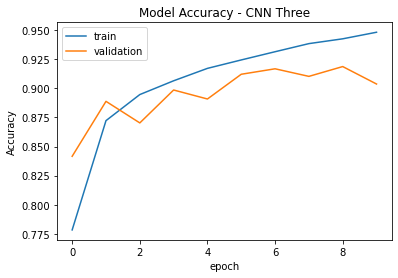


Figure 5: Model Accuracy and Loss for CNN\_Model\_Three

The fifth and final model, CNN\_Model\_Four, had minimal improvement in the accuracy and loss results. This model included batch normalization and a dropout layer. AS seen in Figure 6, it is obvious that the addition of the batch normalization and dropout did not help overfitting.

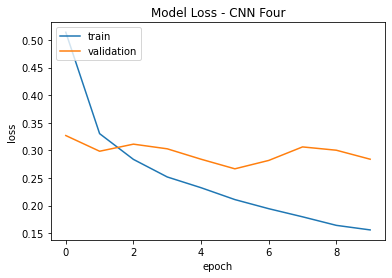
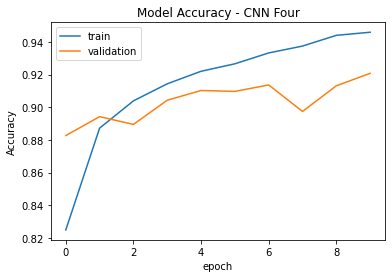


Figure 6: Figure 6: Model Accuracy and Loss for CNN\_Model\_Four

The initial proposal for this research was to conduct the CNN model through Apache Spark. One of the main limitations for CNNs is the high computational costs. With the use of a distributed processing platform, like Apache spark, this would in theory reduce this limitation. However, for this research it was not executed.

As discussed previously, overfitting is a common issue observed for CNN models and is continually being researched to overcome this limitation. Overfitting can be attributed to the lack of training data available which is imbalanced against the real time data. While overfitting has been researched extensively, and solutions such as data augmentation has been developed, it is still a limitation that requires a solid solution. (Alzubaidi *et al.*, 2021)

# Conclusion

#### The aim of this paper was to evaluate the convolutional neural networks for classifying images using the fashion MNIST dataset.

Five different models were developed, trained and tested. The models varied in epochs, number of layers, batch normalization and dropout values. The best performing model for accuracy was CNN\_Model\_Four with a 91.49% accuracy rate. This model incorporated the batch normalization and dropout features. The worst performing model for accuracy was CNN\_Model\_Three at 89.94% which contained dropouts at each layer. While these are the best and worst performing models, the variance is 1.55%, which suggests the parameter tuning that was performed was not optimal.

In terms of loss, the worst model was CNN\_Model\_One-epoch increase at 83.57%. The increased epoch level was not suitable for this dataset, and it is advised for future work that this should remain low for the fashion MNIST dataset which is a relatively small dataset. The variance between the losses for the other four models was 4.3%, similarly to the accuracy there was no major optimization. The best performing model in terms of loss was CNN\_Model\_One at 28.34%.

In addition to loss and accuracy, it is important that the models are not overfitting on trained data. CNN\_Model\_Two had the least amount of overfitting however it should be noted similarly to the accuracy and loss values, the difference between the models was minimal.

##### References

Alzubaidi, L. *et al.* (2021) *Review of deep learning: concepts, CNN architectures, challenges, applications, future directions*, *Journal of Big Data*. Springer International Publishing. doi: 10.1186/s40537-021-00444-8.

Arora, S. *et al.* (2019) ‘An Overview of Apache Pig and Apache Hive’, *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 5(2), pp. 432–436. doi: 10.32628/cseit195250.

Bhobe, M. (2019) *Classifying Fashion with a Keras CNN (achieving 94% accuracy) — Part 2*, *Medium2*.

Cardoso, Â., Daolio, F. and Vargas, S. (2018) ‘Product Characterisation towards Personalisation’, pp. 80–89. doi: 10.1145/3219819.3219888.

Depardon, B. *et al.* (2013) ‘Analysis of Six Distributed File Systems To cite this version : Analysis of Six Distributed File Systems’, *Research Report*, p. 44. Available at: https://hal.inria.fr/hal-00789086.

Dwivedi, K. and Dubey, S. K. (2014) ‘Analytical review on Hadoop Distributed file system’, *Proceedings of the 5th International Conference on Confluence 2014: The Next Generation Information Technology Summit*, pp. 174–181. doi: 10.1109/CONFLUENCE.2014.6949336.

Dwivedi, R. (2020) *Everything You Should Know About Dropouts And BatchNormalization In CNN*, *Analytics India Mag*. Available at: https://analyticsindiamag.com/everything-you-should-know-about-dropouts-and-batchnormalization-in-cnn/.

Ergüzen, A. and ünver, M. (2018) ‘Developing a file system structure to solve healthy big data storage and archiving problems using a distributed file system’, *Applied Sciences (Switzerland)*, 8(6), pp. 1–20. doi: 10.3390/app8060913.

Ghazi, M. R. and Gangodkar, D. (2015) ‘Hadoop, mapreduce and HDFS: A developers perspective’, *Procedia Computer Science*. Elsevier Masson SAS, 48(C), pp. 45–50. doi: 10.1016/j.procs.2015.04.108.

Hedjazi, M. A. *et al.* (2018) ‘A Comparison of Hadoop , Spark and Storm for the Task of Large Scale Image Classification’, in *2018 26th Signal Processing and Communications Applications Conference (SIU)*. IEEE, pp. 1–4.

Henrique, A. S. *et al.* (2021a) ‘Classifying Garments from Fashion-MNIST Dataset Through CNNs’, *Advances in Science, Technology and Engineering Systems Journal*, 6(1), pp. 989–994. doi: 10.25046/aj0601109.

Henrique, A. S. *et al.* (2021b) ‘Classifying Garments from Fashion-MNIST Dataset Through CNNs’, *Advances in Science, Technology and Engineering Systems Journal*, 6(1), pp. 989–994. doi: 10.25046/aj0601109.

Jatana, N. *et al.* (2012) ‘A Survey and Comparison of Relational and Non-Relational Database’, *International Journal of Engineering Research & Technology (IJERT)*, 1(6), pp. 1–5.

Kadam, S. S., Adamuthe, A. C. and Patil, A. B. (2020) ‘CNN Model for Image Classification on MNIST and Fashion-MNIST Dataset’, *Journal of scientific research*, 64(02), pp. 374–384. doi: 10.37398/jsr.2020.640251.

Kayed, M., Anter, A. and Mohamed, H. (2020) ‘Classification of Garments from Fashion MNIST Dataset Using CNN LeNet-5 Architecture’, *Proceedings of 2020 International Conference on Innovative Trends in Communication and Computer Engineering, ITCE 2020*, pp. 238–243. doi: 10.1109/ITCE48509.2020.9047776.

Khan, A. *et al.* (2020) *A survey of the recent architectures of deep convolutional neural networks*, *Artificial Intelligence Review*. Springer Netherlands. doi: 10.1007/s10462-020-09825-6.

Lang, N. (2021) *Breaking down Convolutional Neural Networks: Understanding the Magic behind Image Recognition*, *Towards Data Science*.

Li, Z. *et al.* (2022) ‘A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects’, *IEEE Transactions on Neural Networks and Learning Systems*. IEEE, 33(12), pp. 6999–7019. doi: 10.1109/TNNLS.2021.3084827.

Nocentini, O. *et al.* (2022) ‘Image Classification Using Multiple Convolutional Neural Networks on the Fashion-MNIST Dataset’. doi: 10.3390/s22239544.

Pai, A. (2023) *Analyzing Types of Neural Networks in Deep Learning*, *Analytics Vidhya*.

Pylypenko, I. (2019) *Exploring Neural Networks with fashion MNIST*, *Medium*. Available at: https://medium.com/@ipylypenko/exploring-neural-networks-with-fashion-mnist-b0a8214b7b7b.

Science, M. D. (2021) *How to Import and Plot the Fashion MNIST Dataset Using Tensorflow*, *Medium*. Available at: https://medium.com/the-data-science-publication/how-to-import-and-display-the-fashion-mnist-dataset-using-tensorflow-e72522f684d0.

Shaikh, E. *et al.* (2019) ‘Apache Spark: A Big Data Processing Engine’, in *2019 2nd IEEE Middle East and North Africa COMMunications Conference, MENACOMM 2019*.

Shankar (2023) *Understanding Loss Function in Deep Learning*, *Analytics Vidhya*. Available at: https://www.analyticsvidhya.com/blog/2022/06/understanding-loss-function-in-deep-learning/.

Sharma, N., Jain, V. and Mishra, A. (2018) ‘An Analysis Of Convolutional Neural Networks For Image Classification’, *Procedia Computer Science*, 132, pp. 377–384. doi: 10.1016/j.procs.2018.05.198.

Singh, G. (2023) ‘Introduction to Artificial Neural Networks’, *Analytics Vidhya*.

Sunil, K. and Sivagamasundari, G. (2022) ‘Improvement in Performance of Image Classification based on Apache Spark’, *2022 2nd Asian Conference on Innovation in Technology, ASIANCON 2022*. IEEE, pp. 1–6. doi: 10.1109/ASIANCON55314.2022.9909293.

Taye, M. M. (2023) ‘Theoretical Understanding of Convolutional Neural Network : Concepts, Architectures, Applications, Future Directions’, *computation*, 11(52).

Tudorica, B. G. and Bucur, C. (2011) ‘A comparison between several NoSQL databases with comments and notes’, *Proceedings - RoEduNet IEEE International Conference*. IEEE, pp. 1–5. doi: 10.1109/RoEduNet.2011.5993686.

Umer, A. (2022) *Deep Learning Essentials*, *Medium*.

Vasquez, F. (2018) *Deep Learning With Apache Spark — Part 2*, *Towards Data Science*. Available at: https://towardsdatascience.com/deep-learning-with-apache-spark-part-2-2a2938a36d35.

Violante, A. (2019) *Distributed Deep Learning Pipelines with PySpark and Keras*, *IBM*. Available at: https://community.ibm.com/community/user/ai-datascience/blogs/andre-violante1/2019/06/21/distributed-deep-learning-pipelines-with-pyspark-a.

Wahid, A. and Kashyap, K. (no date) *Cassandra — A Distributed Database System : An Overview*. Springer Singapore. doi: 10.1007/978-981-13-1951-8.