Shoe Brand Image Classification with Deep Neural Networks

Arooshi Taneja
Department of Computer Science
The University of Texas at Dallas
Richardson, Texas, USA
axt210000@utdallas.edu

Pooja Ramanlal Bhaiya
Department of Computer Science
The University of Texas at Dallas
Richardson, Texas, USA
pxr210008@utdallas.edu

Neetesh Kumar Dadwariya
Department of Computer Science
The University of Texas at Dallas
Richardson, Texas, USA
nkd200001@utdallas.edu

Vishakha Singh
Department of Computer Science
The University of Texas at Dallas
Richardson, Texas, USA
vxs200068@utdallas.edu

***Abstract*— Logos are an important part of branding, and companies invest significant resources to design and promote their logos. Therefore, it is essential to develop accurate and efficient methods for logo recognition and classification. This paper proposes an implementation of deep learning neural networks using backpropagation and gradient descent algorithm to efficiently classify the shoe brand images. After conducting multiple experiments, the proposed algorithm achieved a recognition accuracy of 93.75% for identifying various outdoor scenes.**

***Keywords*— Convolutional Neural Network (CNN), Image Classification, Spark, Deep Learning, Transfer Learning, Data Augmentation**

I. INTRODUCTION

The amount of data generated every day is growing at an unprecedented rate. With the advent of digital technology, more and more data is being created, stored, and shared every second. This data comes from a variety of sources, including social media, e-commerce, healthcare, scientific research, and many more.

Big Data and image classification have become increasingly important in the era of digital information. As the volume of data generated every day continues to grow exponentially, it becomes increasingly challenging to extract meaningful insights from it. Image classification is one of the most important applications of Big Data and is widely used in a variety of domains, such as healthcare, security, e-commerce, and marketing.

One specific application of image classification is logo recognition. This is particularly challenging due to the large number of logos, variations in appearance, and the need for real-time processing.

Shoe brand logo classification also plays a critical role in marketing and advertising. Shoe brands invest significant resources in promoting their logos as a means of differentiation and building brand identity. By classifying shoe brand logos, advertisers and marketers can target specific audiences based on their brand preferences and create more effective campaigns.

Logo classification enables customers to find the shoes quickly and easily they are looking for based on the brand they prefer. Brand recognition and loyalty play a crucial role in shaping consumer behavior, especially in the footwear industry.

For decades, image classification has been a significant issue in computer vision. While classifying and understanding images is a simple task for humans, it can be very expensive for computers. Big Data technologies have provided a solution to this challenge by enabling the processing of large datasets using distributed computing architectures. Deep learning algorithms have shown significant success in logo recognition. In image classification using CNN, the model learns both pixel-level and neighbor information through convolution. Convolution aggregates neighboring pixels by multiplying them and summing them up to create features that help classify the image into a class.

Deep convolutional neural networks (CNNs) have completely changed the way that large-scale image detection and classification are done in recent years. Almost all the high-performing algorithms and architectures used today for picture categorization and recognition use deep CNNs in some fashion. [1][2] CNNs are particularly useful for logo classification because they can automatically learn and extract relevant features from the images, without requiring explicit feature engineering.

While increasing the depth of a neural network can enhance its performance, this comes at the cost of additional time and computing power. To mitigate these challenges, transfer learning-based deep learning has emerged as a cost-effective approach. By leveraging the existing model's parameters to a new model the parameters of a pre-trained model to a new model, the learning efficiency of the latter can be expedited and optimized, particularly if the data or tasks are relevant. This can significantly reduce the cost of training while also improving the accuracy of the model. [5]

Logo classification typically involves distinguishing between different logos based on their visual appearance. CNNs are well-suited for this task because they can capture the various visual cues that distinguish between different logos, such as the color, shape, texture, and composition of the logo.[1]

One advantage of using CNNs for logo classification is that they can handle variations in lighting, scale, and orientation, which can be important when dealing with real-world logo images. Additionally, the deep architectures of CNNs allow them to capture increasingly abstract features as the network goes deeper, which can be useful for recognizing subtle differences between similar logos.

II. BACKGROUND WORK

The deep learning revolution is driven by many forms of neural networks, which power various applications such as image classification. ANN (Artificial Neural Network) and DNN (Deep Neural Network) are both types of neural networks with significant differences in architecture, learning, data handling, performance, and hardware requirements. ANN typically consists of a single layer or a few layers of neurons and is suitable for handling small to medium-sized datasets. In contrast, DNN has multiple layers of neurons, making it better suited for handling large and complex datasets. DNN also uses more sophisticated learning algorithms and can achieve higher accuracy and performance than ANN, especially in complex tasks such as image recognition and speech recognition to name a few.

In the perspective of machine learning, transfer learning refers to the practice of leveraging a pre-existing model that has been trained on a similar task as a starting point for a new related task. Compared to DNN, transfer learning has several advantages in big data applications.

Two significant advantages are faster training and improved accuracy. It reduces the time and computational resources and, also addresses the problem of limited data availability by allowing the new model to utilize the vast dataset used to train the pre-trained model. This can lead to better performance and accuracy, especially when the new task has a limited amount of data.

There has been a significant amount of prior research on image classification with CNN. In the field of image classification pertaining to images of shoes, [6] Khosla, Neal focuses on using convolutional neural networks (CNNs) for resolving issues with shoe picture categorization and retrieval. For classification and retrieval tasks, the authors experimented with several network designs using a dataset of more than 30,000 pairs of shoes. They discovered that even a simple three hidden layer network could classify shoes with accuracy levels exceeding 90%. For retrieval, they used transfer learning with the VGGNet architecture and achieved 75.6% precision by computing Euclidean distances between feature vectors. Overall, the authors demonstrated that CNNs can classify and compare shoes with high accuracy.

For automated logo identification in real-world photos, K. Paleček [7], introduces a deep learning system based on FasterR-CNN. An empirical study is done on different design and architectural options, and the system's performance is compared to that of other models like Mask R-CNN or RetinaNet. The authors demonstrate that these decisions have a greater impact than algorithmic changes or data augmentation. The system is tested against a variety of well-known datasets, and by refining the training process. The Red Bull logos in online media and photographs are then searched for using the algorithm.

Even though R-CNN scored highly on PASCAL2010 [8], the method's training requirements call for a sizable, meticulously annotated dataset, which is challenging for shoe logos given their diverse visual characteristics.

Over-fitting in model training may result from insufficient training data. The data augmentation approach is typically used to address over-fitting issues that arise during the training of neural network models. It makes good use of the training pixels and keeps the missing area's normalizing impact.

While attempts have been made to enable parameter coordination in clustered training environments, with frameworks such as Theano aiming to leverage distributed GPU clusters in the near future, there is still room for exploring the potential of utilizing existing batch processing distributed frameworks for training deep neural networks. Over the past decade, distributed data processing frameworks like Hadoop or Spark have been widely adopted and successful, with the MapReduce paradigm and its associated processing algorithms and support tools becoming integral to modern data science and analytics. Given the prevalence of Apache Spark implementations in the industry, it is plausible that it could be a promising platform for large-scale deep neural network training, provided that the framework is effectively leveraged.

The proposed system implements a Deep Neural Network using backpropagation and gradient descent algorithm to efficiently classify the shoe brand images and implements Inception-v3 model and further compares the two. The section below elaborates on the methodology implementation.

III. DATASET

The Nike Adidas Shoes for Image Classification dataset is a collection of 12,500 images of Nike and Adidas shoes. The images are labeled with 10 different classes, which include Nike Air Max 90, Nike Air Max 97, Nike Air Max 270, Nike Air VaporMax, Adidas NMD R1, Adidas Ultra Boost, Adidas Yeezy Boost 350, Adidas Superstar, Adidas Stan Smith, and Adidas Gazelle. The dataset contains a balanced distribution of images across the different classes, with 1250 images per class.

The images in the dataset are in JPEG format and have a resolution of 800x600 pixels. The dataset is split into training and validation sets, with 10,000 images for training and 2,500 images for validation. The training set contains 1,000 images per class, while the validation set contains 250 images per class.

Fig 1. Sample Images

IV. IMPLEMENTED ALGORITHM

A. Data Preprocessing

Data preprocessing is a crucial step which involves processing the input data in a certain way that can be fed to our mathematical models. There are various processing steps that are involved for converting the input data as per the model requirements. The operations involved as follows:

RGB to grayscale- We used an RGB image as our input, which is a tensor of shape (height, width, 3) that represents the image's height, width, and three different color channels. The grayscale intensity values are then calculated using a weighted average applied to the three-color channels. Grayscale photos need less memory and processing power than RGB images, hence converting an image from RGB to grayscale can help lower the computational expense of a model.

Rescaling- The pixel values of a picture are commonly represented in image data as numbers in the [0, 255] range, with 0 denoting black and 255 denoting white pixels. To standardize the input values and enhance the stability and convergence of the model during training, it is usual practice to rescale the pixel values to fall within the range [0, 1].

Image Resizing- While maintaining the image's original aspect ratio, we reduced the image's height and width to 150 pixels each. When preprocessing data for computer vision applications, especially when using convolutional neural networks (CNNs), it is common practice to use this size.

The computational complexity of the model can be decreased by downsizing the photos, which will speed up training and evaluation. As the model is compelled to concentrate on the most crucial aspects of the image, it can also aid in reducing overfitting.

Flattening - Flattening of image data array is a process of converting a two-dimensional (2D) or three-dimensional (3D) array of pixel values representing an image into a one-dimensional (1D) array.

Flattening of image data array was done after resizing the images to a fixed size, and before passing them through a machine learning model. This allowed the model to treat each pixel as a separate feature, which is fed into the input layer of the model.

Data Augmentation

By generating modifications of existing photos, we employed the image augmentation technique in computer vision and deep learning to expand the training dataset [12]. By giving the model additional instances to learn from and by making it more resilient to changes in illumination, orientation, and other elements that can affect image quality, this can enhance the performance of machine learning models.

There were many different techniques that we used for image augmentation, each of which can be applied individually or in combination with others. Some common techniques include:

- Flipping: To produce a mirror image, flip the image either horizontally or vertically. As a result, the model may be able to distinguish objects that are orientated differently.[11]
- Rotation: This entails turning the image a specific number of times, like 90 or 180 degrees [10]. This can imitate the effect of the camera being tilted or inclined and assist the model in learning to distinguish objects that are oriented in different directions.
- Cropping: This involves cropping the image to a smaller size, which can help the model learn to recognize objects that are partially obscured or located in different regions of the image.[11]
- Scaling: This entails scaling the image up or down, which can assist the model in learning to recognize things at various scales and distances.[9]

B. Neural Networks

Neural Networks are a popular machine learning method used to predict outcomes based on input data. They can be thought of as mathematical models that learn to map inputs to outputs by adjusting their parameters based on a set of training data.[13]

Layers of connected nodes, often known as "neurons," make up neural networks, which use mathematics to process the input data. Each neuron receives information from the neurons in the layer below, changes that information in a non-linear manner, and then delivers its output to the neurons in the layer above. This procedure is repeated until the neural network's prediction, which is the output of the last layer, is produced. The network alters its parameters, including its weights and biases, throughout training in order to minimize the difference between its predictions and the actual values in the training data. To finish this process, a gradient descent optimization technique is employed.

Neural networks can be trained to do a wide range of tasks, including speech recognition, picture recognition, forecasting, and natural language processing [13]. They are highly suited for tasks where conventional programming approaches would be challenging or impossible due to their capacity to automatically understand complicated patterns and relationships in data. This has led to them becoming more and more popular.

Feed Forward Neural Network

The output of a feed-forward neural network, also referred to as a single-layer perceptron, is created by passing inputs via a single layer of neurons [15]. As this kind of network can learn to differentiate between various input classes based on their features, it is frequently used for classification tasks.

Generally, feed-forward neural networks, including MLPs and single-layer perceptron, are efficient machine learning models that may be applied to several tasks, including speech and image recognition, natural language processing, and predictive modeling.[17].

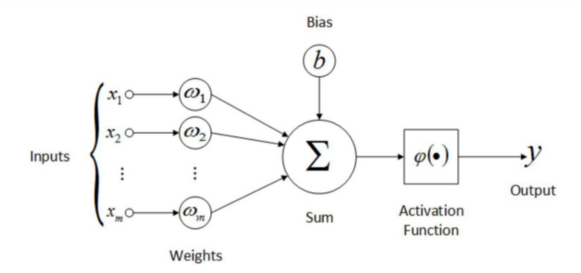

Fig 2. Perceptron Architecture

Layers

Input Layer:
The number of neurons in the Input Layer corresponds to the number of inputs that it receives. In context to our project, the image size is 150 X 150 and therefore contains 22,5000 neurons in the input layer.

Hidden Layer:
The number of neurons in the Hidden Layer is not fixed and is considered as the hyper-parameters. The number of neurons present in the hidden layer depends on the type of task and its perplexity. The number of neurons in the hidden layer our varied from 32 to 128 in context to the implementation.

Output Layer:
The number of neurons in the Output Layer is determined by the number of outputs that are required. In our case the number of neurons in the output layer is 2 since the objective is to classify the image in two different classes.

Activation Functions

Activation functions take an input signal from a neuron and apply a mathematical function to it, producing an output signal. Then a layer of neurons receives this output signal as an input signal. Forward propagation is the process of sending a neuron's output signal to a higher layer.

Because they move a neuron's output to the next neuron's input, activation functions can be thought of as transfer functions. In this approach, activation functions are fundamental to how neural networks function.

1. Sigmoid function: This function takes an input and maps it to a value between 0 and 1. It has an S-shaped curve, which makes it useful for applications where outputs need to be interpreted as probabilities. However, the main disadvantage of the sigmoid function is that it can cause the problem of "vanishing gradients" during backpropagation, which can make it difficult for the network to learn effectively.

2. Tanh function: Tthe tanh function converts an input value to a number between -1 and 1. The tanh function can learn more quickly than the sigmoid function due to its steeper gradient towards the origin. The tanh function can encounter the same issue with vanishing gradients as the sigmoid function.

3. ReLU function: A common option for activation functions in deep learning is the ReLU (rectified linear unit) function. The input signal is subjected to the function f(x) = max(0,x), which will return the input if it is positive and 0 otherwise. ReLU functions are an excellent option for deep neural network training since they are computationally efficient and do not suffer from the issue of vanishing gradients. The ReLU function, however, is not appropriate for applications where negative inputs are significant.

In summary, activation functions in neural networks are used to prepare a neuron's output signal for the network's subsequent layer and to provide the model some non-linearity. Three frequently used activation functions—the sigmoid, tanh, and ReLU—each having unique benefits and drawbacks.

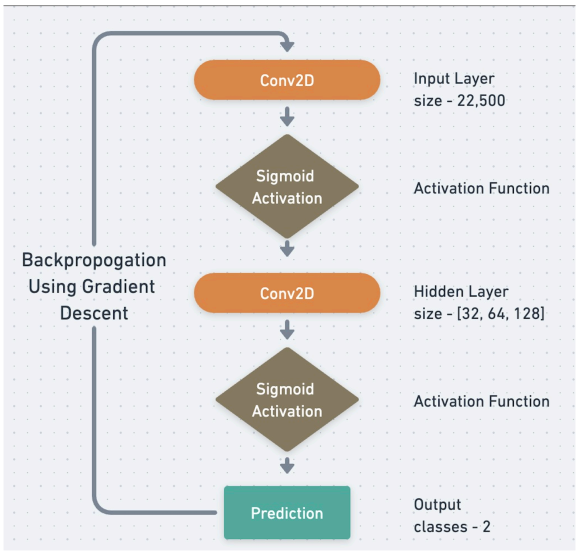

Fig 3. Implemented Neural Architecture

Gradient Descent and Cost Function

A machine learning model's objective is to produce precise predictions based on unobserved data. This is accomplished by minimizing the function(cost), which calculates the discrepancy between the outputs that are anticipated and those that are produced for a certain set of inputs. The model's parameters, W and B, are commonly defined as a function of the cost function.
We employ the gradient descent approach to get the ideal values for these parameters. By calculating the cost function's derivatives with respect to W and B, this technique iteratively updates their values.

The derivative shows how quickly the cost function changes in relation to the parameter, and its sign shows

which way the parameter should be changed to minimize the cost function.
We wish to travel in the opposite way to decrease the cost function if the derivative is positive. Like this, we want to progress in a positive manner if the derivative is negative. The learning rate, which is represented by alpha, determines the size of the update. The magnitude of the step taken in the gradient's direction is determined by the learning Rate, whose value must be carefully selected to avoid divergence.
If the learning rate is too high and the updates are too large, the algorithm might not reach the best result. The algorithm may converge too slowly and require numerous iterations if the learning rate is too low, though.
Realistically, the learning rate should begin at 0.01 but can be adjusted based on how the cost function responds to training. Different gradient descent methods, including batch gradient descent, stochastic gradient descent, and mini-batch gradient descent, can be used to calculate the gradient. The quantity of information utilized to calculate the gradient for each iteration varies between these types.
Overall, gradient descent is a fundamental machine learning optimization approach that enables us to determine the best values for a model's parameters by iteratively updating them in the direction of the cost function's gradient. The algorithm's performance can be considerably impacted by the choice of learning rate, an essential hyperparameter.

$$SSE(y_n) = E = 1/2 \sum_{n=1}^{N} (y_n - y*_n)^2$$

$$MSE(y_n) = E = 1/n \sum_{n=1}^{N} (y_n - y*_n)^2$$

N is the dataset size
y is the predicted output
y is the real output*

Equation 1: Cost Functions

$$W \neg\ W - a \frac{\delta E}{\delta W}$$

Equation 2: Gradient calculation for weights

$$B \neg\ B - a \frac{\delta E}{\delta B}$$

Equation 3: Gradient calculation for Bias

Backward Propagation: Backward Propagation: The goal of the backward propagation is to update new weights and bias parameters that, when combined with gradient descent, reduce the error.

$$\begin{cases} w_{ij}^{(1)} = w_{ij}^{(1)} - \alpha \dfrac{\partial E}{\partial w_{ij}^{(1)}} \\ b_j^{(1)} = b_j^{(1)} - \alpha \dfrac{\partial E}{\partial b_j^{(1)}} \end{cases}$$

$$\begin{cases} w_{jk}^{(2)} = w_{jk}^{(2)} - \alpha \dfrac{\partial E}{\partial w_{jk}^{(2)}} \\ b_k^{(2)} = b_k^{(2)} - \alpha \dfrac{\partial E}{\partial b_k^{(2)}} \end{cases}$$

Equation 4: Gradient update calculation for weights and Bias

$$\frac{\partial E}{\partial b_k^{(2)}} = (y_k - y*_k) \times f'(\hat{y_k})$$

$$\frac{\partial E}{\partial w_{jk}^{(2)}} = \frac{\partial E}{\partial b_k^{(2)}} \times h_j$$

$$\frac{\partial E}{\partial b_j^{(1)}} = f'(\hat{h_j}) \sum_{k=1}^{k} \frac{\partial E}{\partial b_k^{(2)}} \times w_{jk}^{(2)}$$

$$\frac{\partial E}{\partial w_{ij}^{(1)}} = \frac{\partial E}{\partial b_j^{(1)}} \times x_i$$

Equation 6: Derivative calculation for error with respect to weights and Bias

In the above equation the x_i represents the flatten input image of shape (1, 22500), y^* represents the one hot label of shape (1,2) and ĥ represents the forward propagation from input layer to hidden layer before the application of activation function. h represents the forward propagation from Input layer to hidden layer after the application of sigmoid activation. $\hat{y}$ represents the forward propagation from hidden layer to output layer before the application of activation function. y represents the forward propagation from hidden layer to output layer after the application of sigmoid activation. E is described as the error function for measuring the error which is between the true label and the predicted label.

C. *Transfer Learning*

Transfer Learning is a paradigm where we use a pre-trained model, generally pre-trained on a huge corpus, to get re-trained upon the dataset at hand, and learn the features from previous experience. This method is highly used, as the legacy level accepted models are used to train on the dataset at hand, and the training is extremely fast as compared to building a model from scratch. This also gives, the model the habituated facility to learn from the present dataset and also use the pre-trained knowledge to generate results that prove to be good in the recommended metrics.

Our image classification system is designed to be both efficient and scalable, making it well-suited for large-scale datasets. Spark's distributed computing capabilities allow us to process data in parallel, which greatly reduces the overall processing time. This is especially important for image classification tasks, as deep learning models can be very computationally intensive. To further improve the accuracy of our system, we also use Spark's deep learning library.

V. EXPERIMENTS AND RESULTS

Our hyperparameter tuning involves finding the optimal values for these hyperparameters to make the performance of the model better. We have tuned upon the following hyperparameters -

- **Hidden Layer** - Neuronal layers that process input data and produce predictions for the output make up neural networks in most cases. A hyperparameter that can impact a neural network's performance is the number of hidden layers in the network. We observed the best results when number of hidden layers are set to 128.
- **Learning Rate**- A hyperparameter called learning rate determines how rapidly the model changes its weights during training. The model may overshoot the ideal weights due to a high learning Rate, which might cause instability and subpar performance. Best model performance was obtained when learning rate was set to 0.2
- **Iterations** - The number of iterations, also known as epochs, is a hyperparameter that controls how many times the training dataset for the model is used to train the model. Most optimal number of iterations were found to be 30.

Exp No	Hyperparameters	Results
1	Number of Iterations =10 Learning Rate = 0.1 Hidden Layers = 32	Training Loss = 0.2638 Training Accuracy = 0.5978 Testing Accuracy = 0.77767
2	Number of Iterations=20 Learning Rate= 0.2 Hidden Layers=64	Training Loss = 0.2340 Training Accuracy = 0.7391 Testing Accuracy= 0.6709
3	Number of Iterations=30 Learning Rate= 0.1 Hidden Layers=64	Training Loss = 0.2700 Training Accuracy = 0.6565 Testing Accuracy = 0.7600
4	Number of Iterations=20 Learning Rate= 0.3 Hidden Layers=64	Training Loss = 0.2367 Training Accuracy = 0.7456 Testing Accuracy= 0.6581
5	Number of Iterations=10 Learning Rate= 0.3 Hidden Layers=128	Training Loss = 0.2416 Training Accuracy = 0.7586 Testing Accuracy = 0.6625
6	Number of Iterations=30 Learning Rate= 0.2 Hidden Layers=128	Training Loss = 0. 221958 Training Accuracy = 0.822174 Testing Accuracy = 0.803569

Table 1. Hyperparameters tuning results

The best preforming combination of hyperparameters are - *Max Number of Iterations: 30, Learning Rate: 0.2, Hidden Layers: 128*

With the model results as –
Training Accuracy: 0.822174, Test Accuracy: 0.803569
Training Loss: 0.221958, Testing Loss: 0.541667

We plot the accuracy learning plots for various hyperparameters.

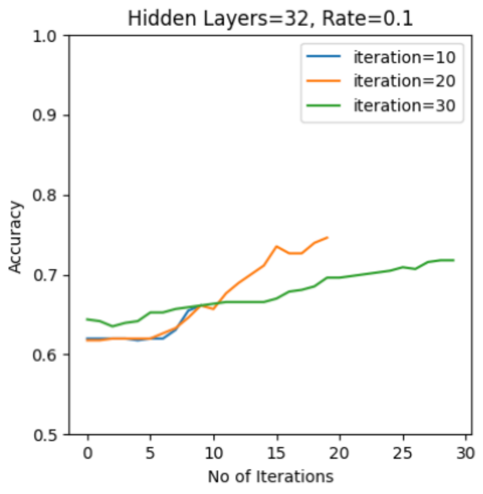

Fig 4. Accuracy vs Iterations Curve {Hidden Layers = 32, Learning Rate = 0.1}

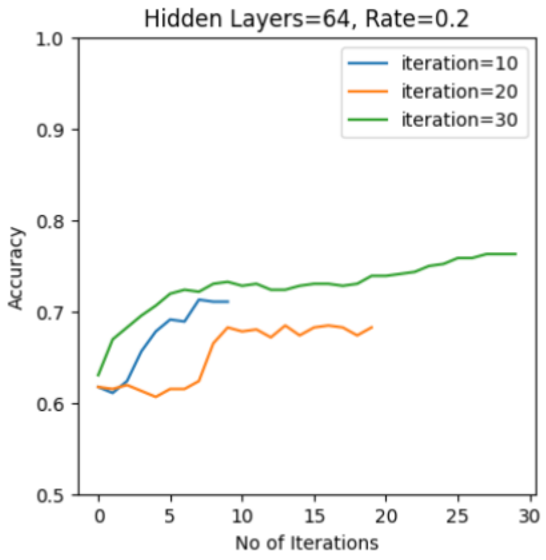

Fig 5. Accuracy vs Iterations Curve {Hidden Layers = 64, Learning Rate = 0.2}

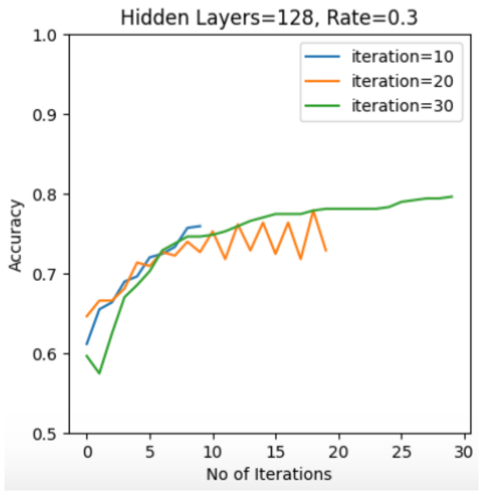

Fig 6. Accuracy vs Iterations Curve {Hidden Layers = 128, Learning Rate = 0.3}

Loss plots for various hyperparameters are also plotted.

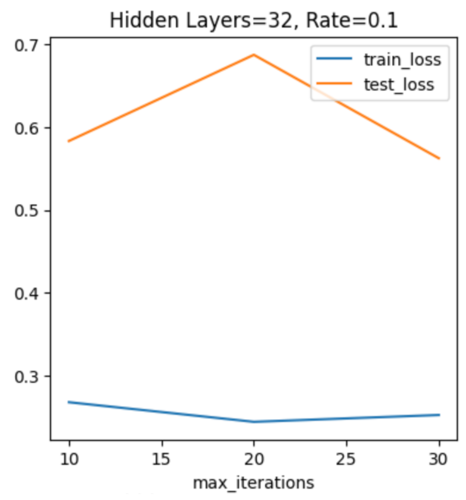

Fig 7. Loss vs Iterations Curve {Hidden Layers = 32, Learning Rate = 0.1}

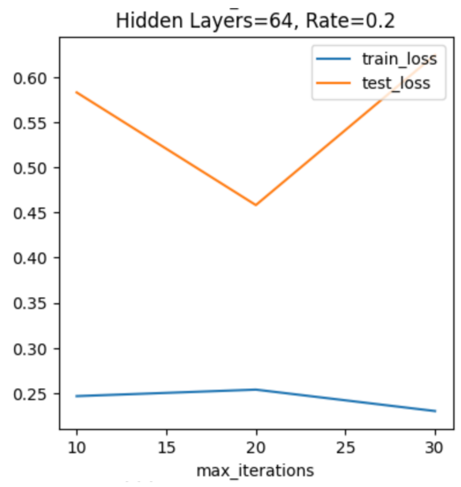

Fig 8. Loss vs Iterations Curve {Hidden Layers = 64, Learning Rate = 0.2}

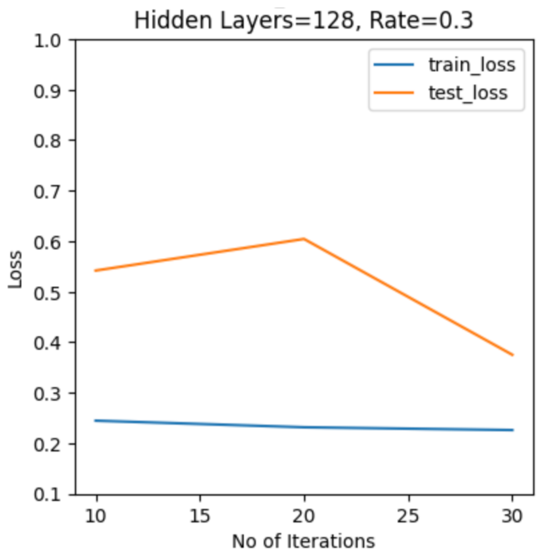

Fig 9. Loss vs Iterations Curve {Hidden Layers = 128, Learning Rate = 0.3}

We have also evaluated a transfer learning approach using 'incpetionV3' as the base model. The results obtained using transfer learning model are as follow –

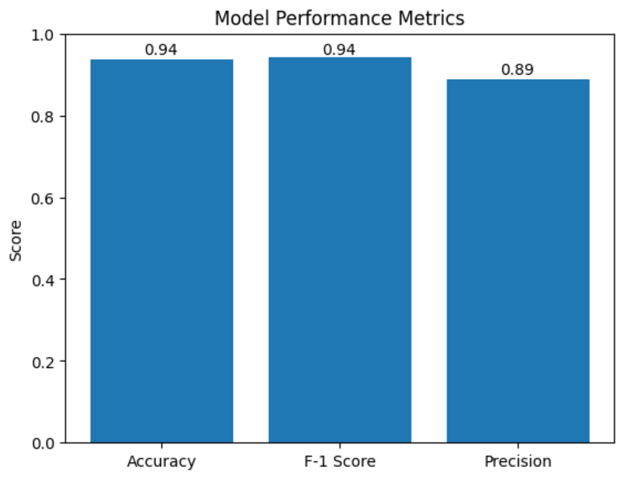

Fig 10. Model Performance Metrices for Transfer Learning Based Model

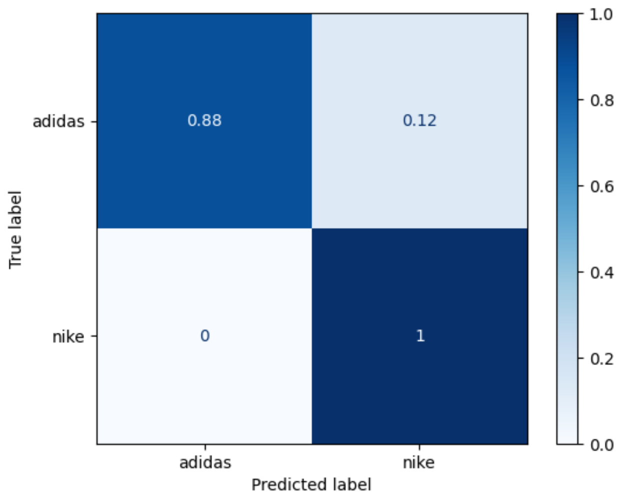

Fig 11. Normalized Confusion Metrix for Transfer Learning Based Model

From the transfer learning-based model, we achieved the accuracy of 0.9375, which is greater than 2-layer Neural Network Model.

VI. CONCLUSION

Image classification can be a time-consuming process, especially when dealing with large datasets. Big data clustered environments can distribute the workload across multiple nodes, allowing for faster processing times.

We observed two techniques for Image Classification in Big Data environment. 1) Image Classification using 2 layer Neural Network written in PySpark and NumPy 2) Image Classification using SparkDL library with the help of 'Transfer Learning'.

We can observe that transfer learning-based model gives better performances. Pre-trained models are typically trained on large and diverse datasets, which makes them better at generalizing to new datasets. Pre-trained models have already learned useful features and patterns in images that are relevant to many different tasks.

Furthermore, We observed that Image Augmentation has shown to increase accuracy by 1% as by generating new images with different variations, image augmentation can help the machine learning model to better generalize and recognize objects in new, unseen images.

VII. FUTURE WORK

For further developments we can explore the application of data parallelism techniques and optimize DNN models for distributed computing which can be essential for training large-scale models with high accuracy. Furthermore, developing strategies for model compression and optimization for efficient inference on Spark clusters can be crucial for deploying DNN models in real-world applications. Other potential research areas include investigating the use of transfer and few-shot learning, developing techniques for active learning, multi-modal classification, and adversarial robustness in the context of DNN models in Spark.

REFERENCES

[1] T. Guo, J. Dong, H. Li and Y. Gao, "Simple convolutional neural network on image classification," 2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA), Beijing, China, 2017, pp. 721-724, doi: 10.1109/ICBDA.2017.8078730.

[2] P. L. Suárez, A. D. Sappa and B. X. Vintimilla, "Cross-spectral image patch similarity using convolutional neural network," 2017 IEEE International Workshop of Electronics, Control, Measurement, Signals and their Application to Mechatronics (ECMSM), Donostia, Spain, 2017, pp. 1-5, doi: 10.1109/ECMSM.2017.7945888.

[3] Plested, Jo, and Tom Gedeon. "Deep transfer learning for image classification: a survey." arXiv preprint arXiv:2205.09904 (2022).

[4] V. Tiwari, C. Pandey, A. Dwivedi and V. Yadav, "Image Classification Using Deep Neural Network," 2020 2nd International Conference on Advances in Computing, Communication Control and Networking (ICACCCN), Greater Noida, India, 2020, pp. 730-733, doi: 10.1109/ICACCCN51052.2020.9362804.

[5] C. Wang et al., "Pulmonary Image Classification Based on Inception-v3 Transfer Learning Model," in IEEE Access, vol. 7, pp. 146533-146541, 2019, doi: 10.1109/ACCESS.2019.2946000.

[6] Khosla, Neal and Stanford. "Building Image-Based Shoe Search Using Convolutional Neural Networks." (2015).

[7] K. Paleček, "Deep Learning for Logo Detection," 2019 42nd International Conference on Telecommunications and Signal Processing (TSP), Budapest, Hungary, 2019, pp. 609-612, doi: 10.1109/TSP.2019.8769038.

[8] Everingham, Mark, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. "The pascal visual object classes (voc) challenge." International journal of computer vision 88 (2010): 303-338.

[9] Simonyan, K., and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[10] Ciregan, D., Meier, U., and Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. arXiv preprint arXiv:1202.2745.

[11] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

[12] Shorten, C., and Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. Journal of Big Data, 6(1), 60.

[13] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444. https://doi.org/10.1038/nature14539

[14] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. Nature, 323(6088), 533–536. https://doi.org/10.1038/323533a0

[15] Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review, 65(6), 386–408. https://doi.org/10.1037/h0042519

[16] Bishop, C. M. (1995). Neural networks for pattern recognition. Oxford University Press.