CNN for Image Classification

Vineeth Marikuntematha Ravisharadhya
University of Adelaide
Adelaide, SA 5000
Email id: a1896845@adelaide.edu.au

Abstract

It's the modern world with never ending fashion analogies of human beings , in one or the other way clothes has been the primary need in this modern civilized world. The clothing brands are competitive enough to enhance their websites through adding features that suggest and help find a choice of clothing. The objective of this paper is to implement multiclass classification for problem solving using the Fashion-MNIST dataset. This paper has more of a representation of Convolution Neural Network (CNN) and have been implemented through altering layers using different architecture like CNN , Visual Geometry Group (VGG) and LeNet-5 CNN. The experiment mainly uses CNN's Adam optimizer to evaluate the performance. The listed models show impressive outcomes of the Fashion-MNIST dataset and has a test accuracy of %91.44.

1. Introduction

Deep learning has proved to be the best practice for most of the researches as it uses data as network and classify into useful information for instance using image classification. Image classification is prominent challenge when it comes to the field of computer vision and it has grabbed interest in the young researchers who are keen to explore the characterization of image and its classification. It makes obvious that the CNN plays a major role when its about image classification as the dataset includes a large amount of image data that has been flooded with help of search engines and social network services. CNN's include several layers, including convolutional , non-linearity , pooling and fully connected[1]. The significance of pooling layer has key role where it lowers the feature records. The best part is that CNN's use fully connected layer from one single neuron layer to the preceding layer of neurons and takes input data in a matrix format then flatten it and then transfers it to output , then this process effectively reduces the amount of data needed for training the model.

Furthermore, considering the CNN model , it will enables to train using smaller datasets which also results in reduced requirement for parameter learning and CNN's employ shared weights to accelerate the processing. Convolutional neural networks share similarities with traditional artificial neural networks. Both types of networks consist of neurons with associated weights and bias values. Each neuron receives input data and computes the result by multiplying the input values and applying an activation function. The entire network still functions as a differentiable rating function, mapping raw image pixels from the input on one side to scores for each category on the other side[1,2]. Performing the experiment on the Fasion_MNIST dataset allows a great opportunity to work on the existing basic CNN and achieve an improvised result with minimal error rates from different architectures of CNN when compared.

Furthermore , this paper contains the section starting with method description where the methods used during the process have been explained , experimental analysis explains about results obtained during the process, conclusion and reference.

2. Description of the method

The architecture of Convolutional Neural Networks (CNNs) have a resemblance towards a traditional Artificial Neural Networks (ANNs), but in both cases their networks are built by organized neurons, which function as optimizing units, also they receive inputs while applying linear functions to generate scalar products combined with non-linear activation functions[1,2].

Furthermore, Convolutional Neural Networks (CNNs) are layers which runs on the convolution operations to the data taken as input. It also involves filters also known as kernels for the input data to extract the patterns and features .

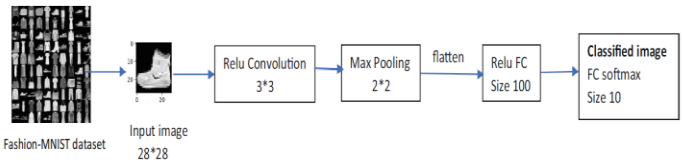

Fig.1. CNN architecture with a single layer for image classification

In this section , the methods for analysis and development for comparison of different architectures its layers and

functions of CNN are explained before the experimentation.

Sequential Model
The sequential model uses a stack of layers which are linear and also it's the simplest method to build neural network where we assign a set of layers through which the data flows sequentially in each layer. This model is frequently used in neural networks where layers are organized one after the other in particular order.

Convolutional Layers
Convolutional layers are the core building blocks of CNNs. These layers are effective in scanning and extracting patterns and features , where they have the input by using learnable filters also called kernels as these help in capturing the similar information , also the size of the kernel is obtained by the type of features that are captured.

Max Pooling Layer
Max pooling layers tend to get employed post convolutional layers as they serve the spatial dimensions of feature maps having preserved the key features. Max pooling layer performs by selecting the maximum value within a small region of the data by downscaling the feature maps. The most common choices are 2x2 adjacent cells that don't overlap, or 3x3 cells, separated from each other by a step of 2 pixels

ReLU(Rectified Linear Unit) Activation
Rectified Linear Unit (ReLU) is an activation function used after convolution or fully connected layers. It is basically used for introduction of non linear which also allows it to learn complex functional mappings between the inputs .the different activation functions are sigmoid , tanh ,RelUetc.

Second Convolutional Layer
The "Second Convolutional Layer" is a crucial part of a multi-layered Convolutional Neural Network (CNN). It comes after the initial convolutional layer and is responsible for identifying more intricate and abstract features in the data. Like the first layer, it uses filters to analyze the information from the previous layer. However, the filters in this layer are designed to recognize higher-level patterns and structures, allowing the network to understand more complex visual information by building upon the features it has already extracted.

Dropout Layer
Dropout layer is a regularization technique where it is enhanced for the generalization ability of neural network by randomly deactivating a set of neurons while the training process and also help prevent overfitting. Lastly the models are aligned into a single network.

Flatten Layer
The Flatten layer acts as a connector between the feature extraction layers and the decision-making layers, making sure that the network can effectively use the features it has learned. It is also respomsible for reshaping multi-dimensional data like one-dimensional vector.

Fully Connected Layers
Fully connected layers are also called as densely connected, which are essential building blocks in neural networks. They contain a significant number of parameters that are adjusted during training to improve the network's performance. In classification tasks, the outputs of fully connected layers are used to make predictions, with the class having the highest score being chosen as the predicted class. These layers are versatile and can be integrated into various types of neural network architectures for making important decisions

Softmax Activation
Softmax activation typically serves a specific pupose in neural networks and it transforms the raw numerical output of a network into a set of probabilities where each class is assigned a probability score.The transformation ensures that the sum of all probabilities equals 1, making it suitable for problems where an input can belong to one of several classes. It's commonly used in multi-class classification scenarios to generate a probability distribution, simplifying the process of class prediction. Along with the Cross-Entropy loss function, Softmax helps train the network by quantifying the difference between predicted probabilities and actual class labels.

Average Pooling Layer
Average pooling layers helps in reducing spatial dimensions of feature maps while capturing representative information. Unlike max pooling, average pooling computes the average value within a pool, making it more manageable and less susceptible to overfitting. Average pooling is often used in combination with max pooling to strike a balance between selecting the maximum and calculating the average values within regions. It is a valuable technique for summarizing and retaining important details in images or feature maps.

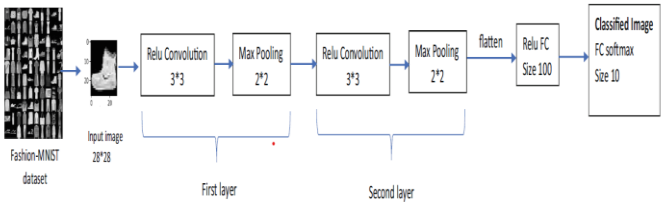

Fig2.1 Neural network with multiple convolutional layers.

Now in this paper I have used three architectures mainly,

Basic Convolutional Neural Networks (CNN's)

A basic CNN or Convolutional Neural Network , is a fundamental architecture ideal for operations like image analysis where it learns features automatically from the data enhancing it for useful image classification. The CNN used in this paper comprises several layers designed to process and analyze images It starts with two convolutional layers, each tasked with identifying patterns and features in the input images. These layers are followed by max-pooling layers that reduce the dimensions of the extracted features, keeping the most important ones. To prevent overfitting, dropout layers randomly deactivate some of the network's

neurons during training. A Flatten layer reshapes the data for further processing by fully connected layers. Two of these layers are present, the first with 128 neurons and the second with 10. The final layer uses SoftMax activation to compute class probabilities, determining the predicted class for multi-class classification tasks.

Visual Geometry Group(VGG)

Visual Geometry Groups are known for their deep structure and is often used for image classification and a series of layers work in harmony to process input images effectively. . The Convolutional Layers discrete patterns and features, while MaxPooling Layers condense feature maps, preserving critical information. Additional Convolutional Layers are added to increase network depth. The Flatten Layer restructures data for further processing, and Fully Connected Layers make high-level decisions and deliver class predictions. To combat overfitting, a Dropout Layer randomly deactivates neurons during training. The architecture culminates with a SoftMax activation layer for multi-class classification, striving to develop a deep network capable of learning intricate image representations.

3. Experimental Analysis

The Experimentation on of the Convolutional Neural Networks (CNN's) was done in python using jupyter lab and google colab. In this paper we have used three architectures of CNN , namely Basic Convolutional Neural Network(CNN),Visual Geometry Group(VGG) and LeNet-5.

Before that I have used various libraries like numpy for numerical operations, pandas for data structures, seaborn for data visualization, matplotlib for animated and interactive visualization , tensorflow for building and training deep learning models , tensorflow's keras for high level interface for building and training deep learning models.

Firstly during the process of model building , the fashion-MNIST data was loaded using the pandas and have been displayed.The later part I have did the data visualization in greyscale by loading the image data into two arrays by creating a 5 X5 grid which shows 25 subplots for displaying images as shown in figure 3

Fig 3: Data visualization in greyscale

Now the architectures for CNN model building I have used these to achieve the accuracy and get the desired output. I have did a three architecture CNN which comprise of Basic Convolutional Neural Networks (CNN's) , Visual Geometry Group(VGG) , LeNet-5 . Moving on to one by one results or outcome of each model architecture of CNN are as follows:

Basic Convolutional Neural Networks (CNN's)

A small overview of the architecture used in the process as follows , used sequential model using tensorflow's keras and have used 2D convolution layer with 32 filters with each 3x3 kernel, and have set the strides parameter as 2 by using ReLu activation, also the layers such as Dropout, Maxpooling2d , Flatten() and Dense. Furthermore, in the process summary of the model have been used, after the process we use plot_model for visual representation of a neural network model as shown in fig 4.

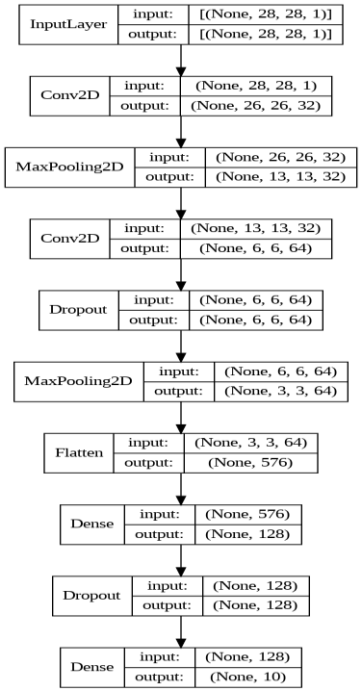

Fig 4: Representation of a neural network model (CNN)

As per the process we have defined batch_size to calculate the training samples and have run the compile command using the optimizer Adam and accuracy metrics, after that we have included used epoch_info where we have set the number of training epoch , also validation split have been kept at 0.15 i.e 15% portion of training data is used.

Table1. Represents the various accuracies on basic CNN on train dataset

Number of iterations	Accuracy	Validation Accuracy	Val_loss
5	0.7494	0.7738	0.6469
10	0.8037	0.8349	0.5080
15	0.8268	0.8564	0.4441
20	0.8421	0.8621	0.4182

Then further we test accuracy for the test data, the accuracy are shown in the table

Table 2: Represents the test accuracy on Basic CNN

Metrics of Basic CNN	
Loss	Accuracy
0.4121	0.8683

The fig 5 and fig6 depicts the accuracy in the train and validation data increase significantly and there is no over-fitting of the model and the accuracy have been mentioned in above table 2.

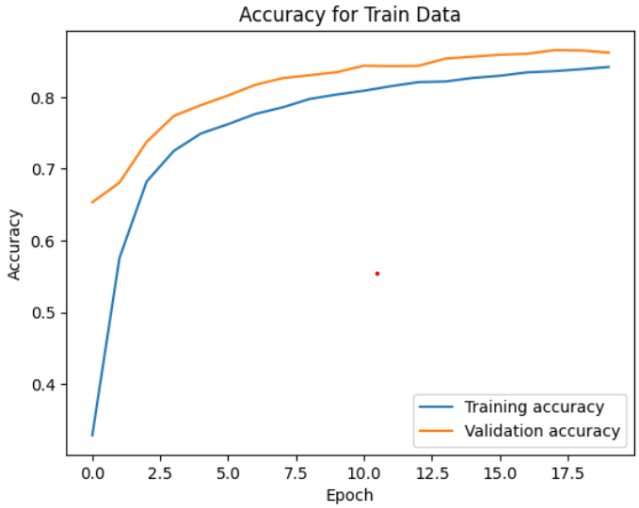

Fig 5: Represents accuracy of train data on Basic CNN model

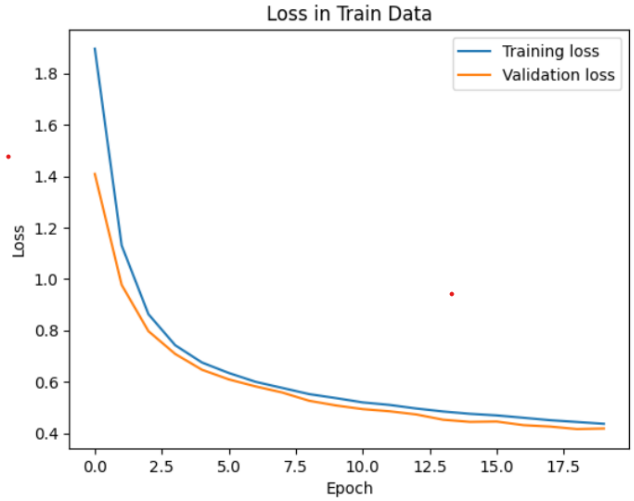

Fig 6: Represents loss in train data on Basic CNN model

Furthermore, we use the predict command to visualize a confusion matrix for basic CNN model prediction. The resulting heat map provides a visual representation of the basic CNN model seeing the results it helps to identify patterns of true image and incorrect classification, the values are represented in the Table 3 and fig7.

Table 3 : Represents accuracy of train data on Basic CNN model

Types of objects	Precision	Recall	F1-score	support
T-shirt	0.80	0.80	0.82	1000
Trouser	0.98	0.97	0.98	1000
Pullover	0.88	0.67	0.76	1000
Dress	0.90	0.87	0.89	1000
Coat	0.71	0.86	0.78	1000
Sandal	0.96	0.97	0.96	1000
Shirt	0.64	0.70	0.67	1000
Sneaker	0.94	0.92	0.93	1000
Bag	0.98	0.96	0.97	1000
Ankle boot	0.93	0.95	0.95	1000
Accuracy			0.87	1000
Macro Avg	0.87	0.87	0.87	1000
Weighted avg	0.87	0.87	0.87	1000

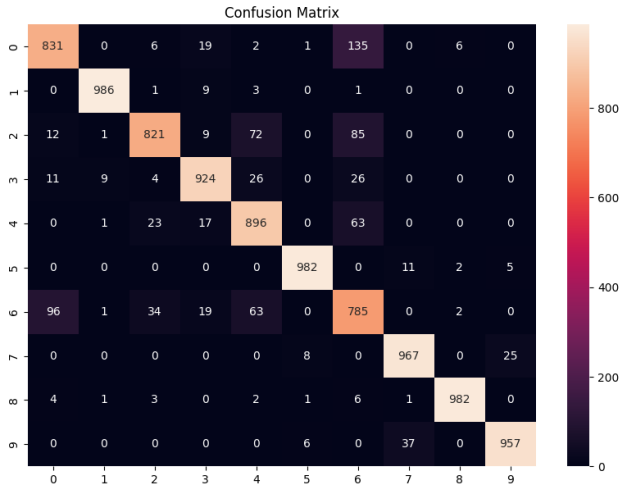

Fig 7: Represents confusion matrix on Basic CNN model

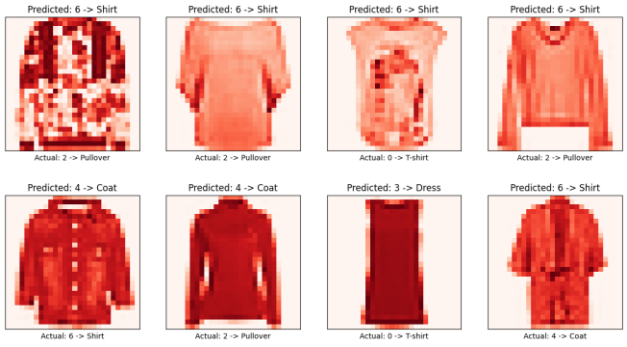

Fig 8 : Represents incorrectly classified images on Basic CNN model

Lastly, the fig 8 presents the subset of incorrectly classified images allowing to inspect and compare the predicted and actual class label of each image.

Visual Geometry Group(VGG)

A small overview of the architecture used in the process as follows , used sequential model using tensorflow's keras and have used 2D convolution layer with 32 filters with each 3x3 kernel, and have set the strides parameter as 2 by using ReLu activation, also the layers such as Dropout, Maxpooling2d , Flatten(),SoftMax and Dense layers and activation functions.

Here in this process I have used VGG instead of VGG-16 as VGG-16 uses 224x224 pixelate so the architecture doesn't fit for our dataset.

Furthermore, in the process summary of the model have been used, after the process we use plot_model for visual representation of a neural network model as shown in fig 9.

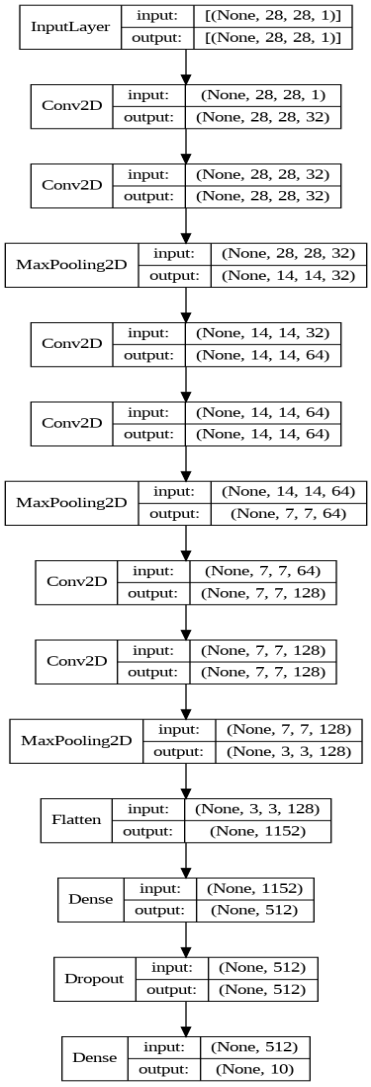

Fig 9: Representation of a neural network model VGG model

As used earlier process continues same ,as we have defined batch_size to calculate the training samples and have run the compile command using the optimizer Adam and accuracy metrics, after that we have included used epoch_info where we have set the number of training epoch , also validation split have been kept at 0.15 i.e 15% portion of training data is used.

Table 4. Represents the various accuracies on VGG model on train dataset

Number of iterations	Accuracy	Validation Accuracy	Val_loss
5	0.8336	0.8528	0.4082
10	0.8837	0.8894	0.3049
15	0.9059	0.9037	0.2705
20	0.9168	0.9058	0.2574

Then further we test accuracy for the test data, the accuracy are shown in the table 5.

Table 5: Represents the test accuracy on VGG model

Loss	Accuracy
0.23519	0.91449

The fig 10,fig 11 depicts the accuracy and loss in the train and validation data increase significantly and there is no over-fitting of the model and the accuracy have been mentioned in above table

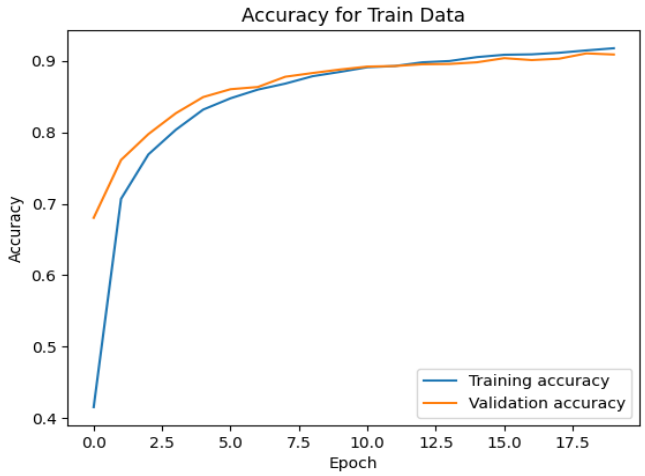

Fig 10 : Represents accuracy of train data on VGG model

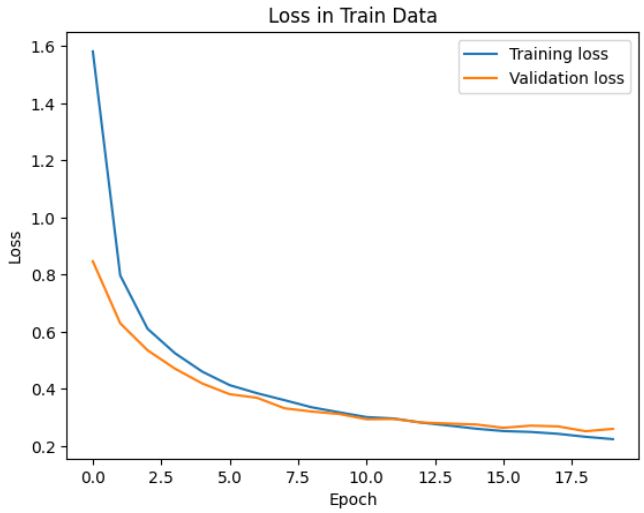

Fig 11 : Represents accuracy of train data on VGG model

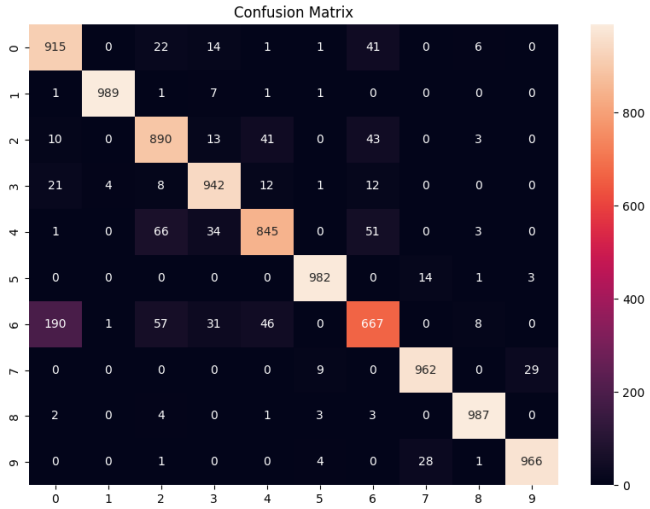

Fig 12: Represents confusion matrix on VGG model

Table 6 Represents accuracy of train data on VGG model

Types of objects	Precision	Recall	F1-score	support
T-shirt	0.80	0.92	0.86	1000
Trouser	0.99	0.99	0.99	1000
Pullover	0.85	0.89	0.87	1000
Dress	0.90	0.94	0.92	1000
Coat	0.89	0.84	0.87	1000
Sandal	0.98	0.98	0.98	1000
Shirt	0.82	0.67	0.73	1000
Sneaker	0.96	0.96	0.96	1000
Bag	0.98	0.99	0.98	1000
Ankle boot	0.97	0.97	0.97	1000
Accuracy			0.91	1000
Macro Avg	0.91	0.91	0.91	1000
Weighted avg	0.91	0.91	0.91	1000

Furthermore, we use the predict command to visualize a confusion matrix for VGG model prediction. The resulting heat map provides a visual representation of the VGG model seeing the results it helps to identify patterns of true image and incorrect classification, the values are represented in the fig 12 and table 6.

Table provides a summary of the performance metrics of CNN classification model.

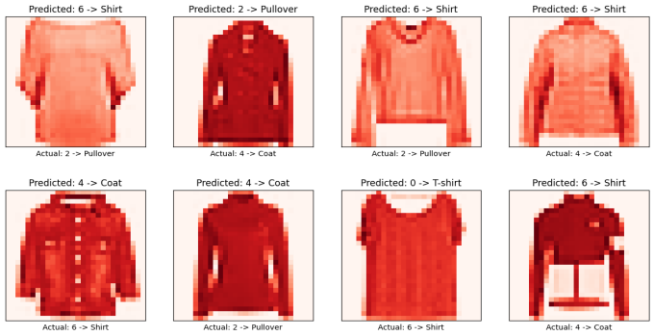

Fig 8 : Represents incorrectly classified images on VGG model

Lastly, the fig 8 presents the subset of incorrectly classified images allowing to inspect and compare the predicted and actual class label of each image , the majority of the incorrect classified images have been corrected through VGG based architecture CNN model.

Comparison between two architecture based CNN's accuracy:

Table 7: Represents accuracy in comparison of basic CNN and VGG model

Number of iterations	Basic CNN Accuracy	VGG based CNN Accuracy
5	0.7494	0.8336
10	0.8037	0.8837
15	0.8268	0.9059
20	0.8421	0.9168
Test Accuracy	0.8683	0.91449

The above table 7 clearly shows that VGG based architecture shows the best accuracy results and has the lower incorrectly classified images when compared to the basic CNN model process.

Code

The code has run successfully giving a significant improvisation between the two models namely the basic CNN model and VGG model .the code has been uploaded for the reference of the tutor by the link: https://github.com/vin66788/Deeplearninga1896845aa.git

4. Conclusion

The proposed experiment sought to predict and analyze the incorrectly classified images using the models both the basic CNN and VGG. However , the VGG performs better than the basic CNN model , where the basic CNN model gives less accuracy and more incorrectly classified images, even though there were alterations made in the number of iterations the model was not as successful as VGG giving about 91.68% accuracy of train data and 91.44% of the test data. The basic CNN method can be further improved by tuning the parameters like the activation function. But , requires more time than the VGG model as there will be changes made in the basic CNN model.

Reference

1. Janjua, J & Patankar, A n.d., 'Framework of CNN Architecture for Fashion Image Classification', in Applied Computational Technologies, Springer Nature Singapore, Singapore, pp. 96–103**.**
2. Meshkini, K, Platos, J & Ghassemain, H 2020, 'An Analysis of Convolutional Neural Network for Fashion Images Classification (Fashion-MNIST)', in Advances in Intelligent Systems and Computing, vol. 1156, Springer International Publishing, Cham, pp. 85–95.
3. Vijayaraj, A, Vasanth Raj, PT, Jebakumar, R, Gururama Senthilvel, P, Kumar, N, Suresh Kumar, R & Dhanagopal, R 2022, 'Deep Learning Image Classification for Fashion Design', Wireless Communications and Mobile Computing, vol. 2022, pp. 1–13.