Volume 11, Issue 3, July 2024 ISSN: 2348-8565 (Online)

IJMERT

International Journal of Modern Engineering and Research Technology

Website: http://www.ijmert.org Email: editor.ijmert@gmail.com

Enhanced CNN Architecture for Image Classification on the Fashion-MNIST Dataset

Shivani Awasthi
Research Scholar M.Tech.
Computer Science and Engineering
Takshshila Institute of Engineering and Technology
Jabalpur (M.P.), India
Email: shivaniprot29@gmail.com

Swati Soni
Assistant Professor
Department of Computer Science and Engineering
Takshshila Institute of Engineering and Technology
Jabalpur (M.P.), India
Email: swatisoni@takshshila.org

ABSTRACT

Fashion plays a vital role in our everyday lives, impacting everyone significantly. In this study, a CNN was used to train images of various fashion styles, successfully predicting them. Deep learning, particularly CNNs, is widely used across many industries for its accurate responses in real-world scenarios. Apparel producers use CNNs to solve issues related to clothing recognition, search, and suggestions in online stores. This study trained CNN-based deep learning architectures using the Fashion-MNIST dataset to discriminate between fashion images. The work employs convolutional layers, filter sizes, and fully connected layers for classification, utilizing activation functions, optimizers, learning rates, dropout rates, and batch sizes. The results showed that the accuracy of predictions is influenced by the choice of optimizer, activation function, and dropout rate.

The major differences between the Basic CNN and Advanced CNN architectures for the Fashion-MNIST dataset highlight the distinct design choices and their impact on performance. The Basic CNN consists of 6 layers, including one convolutional layer with 64 filters, one max pooling layer, a flatten layer, and two dense layers with 128 and 10 units, respectively. This simpler architecture lacks regularization techniques and has a total of 1,386,506 trainable parameters. On the other hand, the Advanced CNN is more complex with 13 layers, incorporating three convolutional layers (with 32, 64, and 128 filters), three batch normalization layers, three max pooling layers, two dropout layers (each with a dropout rate of 0.5), and three dense layers with 256, 128, and 10 units. This advanced architecture includes regularization techniques like Batch Normalization and Dropout, resulting in 422,474 trainable parameters and 448 non-trainable parameters.

Performance-wise, the Basic CNN achieved a training accuracy of 100% but had a validation accuracy of approximately 91.7%, with training and validation losses of 0.0005 and 0.6145, respectively. This indicates a high degree of overfitting, as evidenced by the perfect training accuracy that did not translate to a proportionately high validation accuracy. In contrast, the Advanced CNN demonstrated a training accuracy of about 98.89% and a higher validation accuracy of around 93.3%. Its training and validation losses were 0.0371 and 0.2087, respectively, showcasing better generalization and controlled overfitting due to the regularization techniques employed. The advanced architecture's enhancements, including additional convolutional layers and the use of Batch Normalization and Dropout, led to superior performance, with more *consistent and higher validation accuracy throughout the training process compared to the Basic CNN.*

Keywords:—*Image Classification, Convolutional Neural Network, Fashion-MNIST, Batch Normalization, Dropout.*

I. INTRODUCTION

Fashion has always been quite important, in our daily lives. It also significantly affects the lives of all people. In this study, photos of various fashion styles were trained using CNN, and the results were correctly predicted. Deep learning has seen significant use recently in numerous other areas. When used in real-world circumstances, a CNN is a deep neural network that produces the most accurate results. Clothing manufacturers have used CNN to handle a number of problems with clothing detection, search, and recommendation on their online storefronts. Using a collection of pictures from the Fashion-MNIST dataset, 2 CNN-based deep learning architectures are trained to distinguish between different shots. The CNN model is utilised to evaluate its conclusions using the Fashion-MNIST datasets.

A. Digital Image Processing

Binary Image - A binary image is one that consists of pixels that can have one of exactly two colors, usually black and white. Binary images are also called bi-level or two-level, Pixelart made of two colours is often referred to as 1-Bit or 1bit. This means that each pixel is stored as a single bit—i.e., a 0 Black or 1 White.

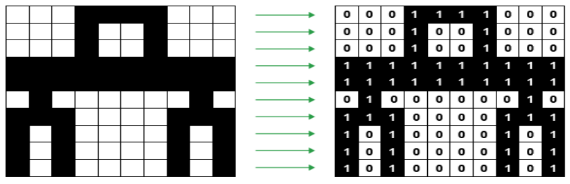


Figure 1: Inverse Binary Image

Gray Scale Image: A grayscale (or graylevel) image is simply one in which the only colors are shades of gray.

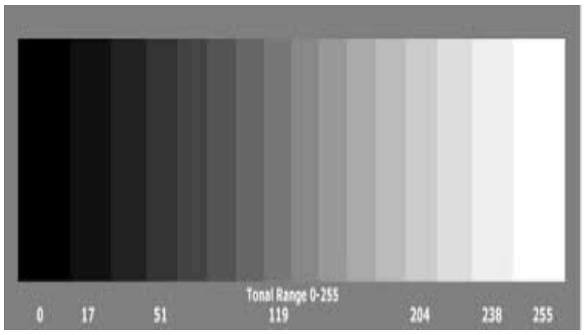


Figure 2: Gray Scale Values

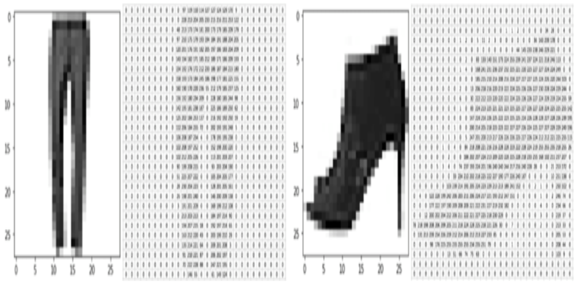


Figure 3: Inverted Gray Scale Image

Color image: Color images are represented using the RGB color model, where each pixel is composed of three-color channels: red, green, and blue. Each color channel typically requires 8 bits of information to represent 256 levels of intensity (0-255). Therefore, RGB color images commonly have 24 bits per pixel (8 bits per channel × 3 channels), but higher color depths such as 32 bits per pixel (where an additional alpha channel for transparency is included) are also used.

ANN is not appropriate for picture datasets since they require the conversion of 2-dimensional images into 1-dimensional vectors, which makes image classification issues more challenging when using ANN. As a result, there are exponentially more trainable parameters. It requires processing and storage power to increase trainable parameters. In other words, it would be expensive and time consuming.

B. CNN (Convolutional Neural Networks)

A Convolution Neural Network or CNN is a type of artificial neural network, which is widely used for image/object recognition and classification for image processing and recognition applications, a Convolution Neural Network (CNN) is a sort of deep learning technique that works well.

It is made up of multiple layers, including convolution layers, pooling layers, and fully connected layers. In order to teach CNNs to identify patterns and features linked to certain objects or classes, a sizable dataset of labelled images is used.

Sequence of CNNLayersInput =>Convolution =>Relu =>Fully Connected =>Softmax.

Key components of a Convolutional Neural Network include:

Convolutional Layers: These layers apply convolutional operations to input images, using filters (also known as kernels) to detect features such as edges, textures, and more complex patterns. Convolutional operations help preserve the spatial relationships between pixels.

The essential part of a CNN is its convolutional layers, which are where elements like edges, textures, and forms are extracted from the input image by applying filters.

After the convolutional layer's output is processed through pooling layers, the feature maps are down-sampled to lower the spatial dimensions while keeping the most crucial data. One or more fully connected layers are then applied to the output of the pooling layers in order to classify or forecast the image.

Image Kernels or Filters: Kernal in a CNN is a small matrix of weights that slides over the input data (such as an image), performs element-wise multiplication with the part of the input it is currently on, and then sums up all the results into a single output pixel.An image kernel is a small matrix used to apply effects like the ones you might find in Photoshop or Gimp, such as blurring, sharpening, outlining or embossing. They're also used in machine learning for 'feature extraction', a technique for determining the most important portions of an image. In this context the process is referred to more generally as "convolution".

The matrix on the left contains numbers, between 0 and 255, which each correspond to the brightness of one pixel in a picture of a face.

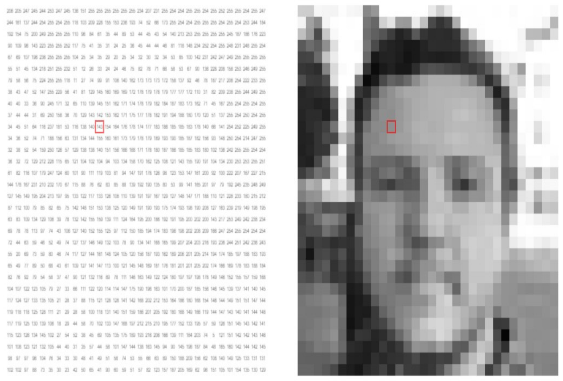


Figure 5: Gray Scale Image Matrix

for each 3x3 block of pixels in the image on the left, we multiply each pixel by the corresponding entry of the kernel and then take the sum. That sum becomes a new pixel in the image on the right.

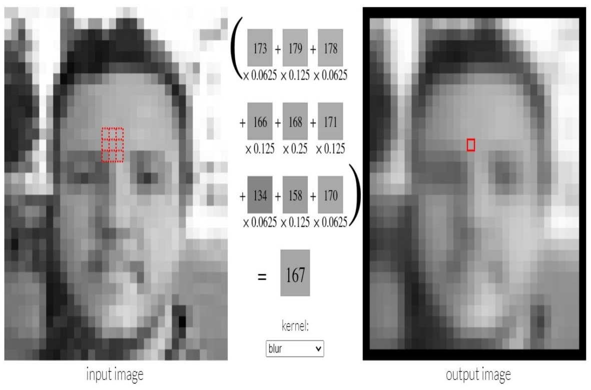


Figure 6: Image – Kernel Product Output

Stride: Stride determines how many squares or pixels our filters skip when they move across the image, from left to right and from top to bottom.

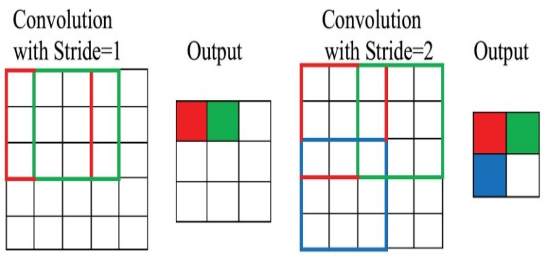


Figure 7.Convolution with Stride

Padding Layer: In convolutional neural networks, the term "padding" refers to the number of pixels that the CNN kernel adds to an image during processing. Every additional pixel added to a CNN with padding set to 0 will have a value of none. The creation of a convolutional neural network requires padding. A small, filtered image will result from shrinking the original and using a neural network with hundreds of layers.

Feature Map:A feature map is the output of a convolution layer representing specific features in the input image or feature map.During the forward pass of a CNN, the input image is convolved with one or more filters to produce multiple feature maps. Each feature map corresponds to a specific filter and represents the response of that filter to the input image.

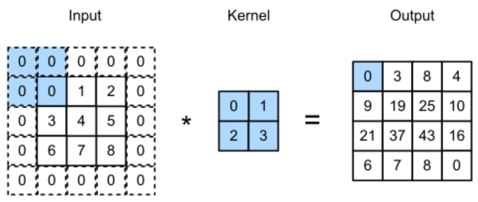


Figure 8: Convoluted Feature Mapin CNN

By pooling layers down sampling the input's spatial dimensions, the number of network parameters and computational complexity are decreased. A typical pooling operation called "max pooling" chooses the largest value among a set of adjacent pixels. Sliding a two-dimensional filter over each feature map channel and summarising the features that fall inside the filter's coverage area constitute the pooling operation.

Types of Pooling Layers: Max Pooling, Min Pooling, Average Pooling

Max Polling: The process of pooling that chooses the maximum element from the area of the feature map that the filter covers is called max pooling. As a result, the feature map that results from the max-pooling layer would include the most noticeable features from the prior feature map.

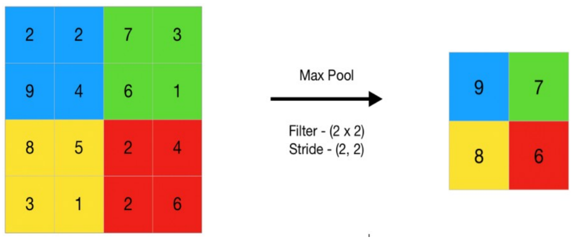


Figure 9: Max Polling

Average Polling: By using average pooling, the average of the items in the feature map area that the filter covers are calculated. Therefore, average pooling provides the average of the features present in a patch, whereas max pooling provides the most noticeable feature in a specific patch of the feature map.

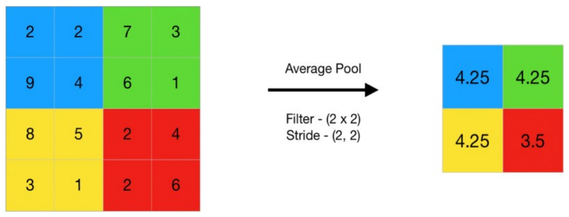


Figure 10: Average Polling

Dimensionality reduction (Pooling Layer Advantages): The primary benefit of pooling layers is their assistance in lowering the feature maps' spatial dimensions. Information loss (Pooling Layer Drawbacks): One of the primary drawbacks of pooling layers is their tendency to remove some information from the input feature maps, which may be crucial for the task of classification or regression at the end.

Fully Connected or Dense Layer: These layers are responsible for making predictions based on the high-level features learned by the previous layers. They connect every neuron in one layer to every neuron in the next layer. The dense layer is a simple Layer of neurons in which each neuron receives input from all the neurons of the previous layer, thus called as dense. The dense layer is used to classify images based on output from convolution layers.

Activation Functions: Rectified Linear Unit (ReLU) and other non-linear activation functions add non-linearity to the model, enabling it to discover more intricate links in the data. A mathematical formula that is applied to each neuron's output in a neural network is called an activation function. Its goal is to provide non-linearity to the model so that the network can learn and depict more intricate correlations and patterns in the data. A neural network would be reduced to a linear model without activation functions, unable to perform tasks like image recognition, natural language processing, or any other issue requiring the capture of complex structures in the input data.

ReLU Activation Function: Non-linear activation functions, such as Rectified Linear Unit (ReLU), introduce non-linearity to the model, allowing it to learn more complex relationships in the data.

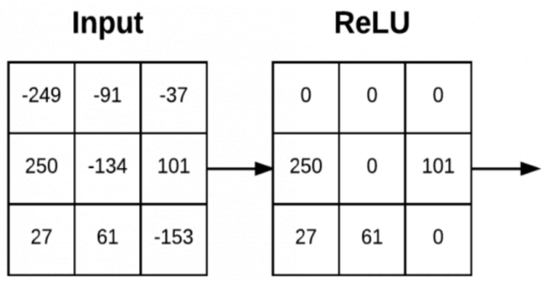


Figure 11: ReLU Activation Function

Softmax Activation Function: SoftMax is an activation function which converts the inputs and output of the last layer of your neural network into a discrete probability distribution over the target classes. Commonly used in the output layer for multi-class classification problems.

Batch Normalization: Batch Normalization is a technique used in neural networks, including Convolutional Neural Networks (CNNs), to normalize the activations of a previous layer at each batch. It helps in reducing internal covariate shift, which refers to the change in the distribution of network activations due to parameter updates during training.

In batch normalization, two parameters are learned and applied to each feature map:

1. Scale (γ): This parameter scales the normalized value.
2. Shift (β): This parameter shifts the normalized value.

These parameters are learned during training and allow the model to adaptively adjust the normalized values to better suit the task at hand.

For a CNN with a batch normalization layer, the number of parameters per feature map is 2 (γ + β), as there are two learnable parameters for each feature map.

Dropouts Layer: The dropout layer is a mask that preserves all other neurons unaltered while nullifying particular neuron's contributions to the following layer.

Dropout layers do not affect the number of parameters in a neural network. Dropout is a technique used only during the training phase to improve generalization and prevent overfitting, but it does not change the architecture of the network itself, nor does it alter the number of weights and biases.

dropout layers in our network architecture affects how the network learns by temporarily ignoring certain neurons during training. However, dropout layers do not change the number of parameters (weights and biases) in the network. The parameters are determined by the dense (fully connected) layers and other learnable layers in the network.

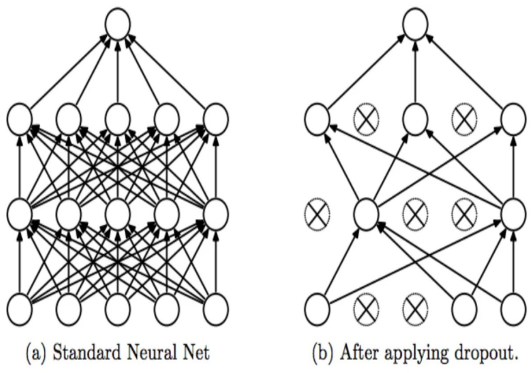


Figure 12: Dropouts Leyer

II. Literature Review

Vijayarajet al. [1, 2022], In this research, convolutional neural networks (CNN) were used to train images of different fashion styles, which were attempted to be predicted with a high success rate. Deep learning has been widely applied in a variety of fields recently. A CNN is a deep neural network that delivers the most accurate answers when tackling real-world situations. Apparel manufacturers have employed CNN to tackle various difficulties on their e-commerce sites, including clothing recognition, search, and suggestion. A set of photos from the Fashion-MNIST dataset is used to train a series of CNN-based deep learning architectures to distinguish between photographs.

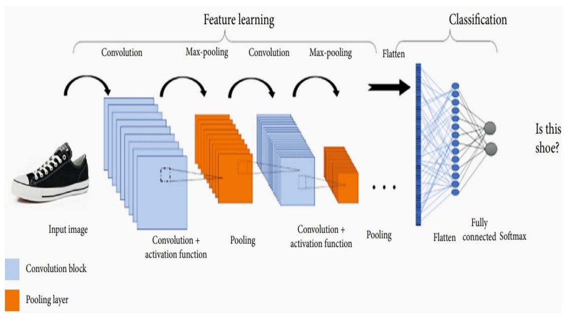
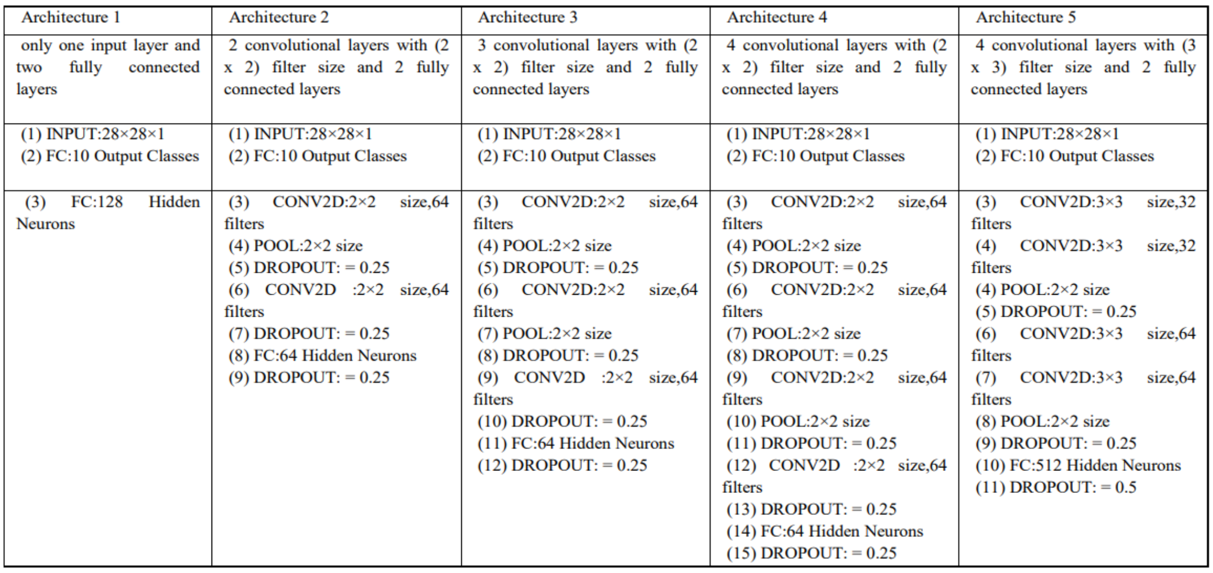


Figure 13: CNN for image classification

In this paper, we proposed to study fashion image classification with four different neural network models to improve apparel image classification accuracy on the Fashion-MNIST dataset. The network models are tested with the highest accuracy with a Fashion-Product dataset and a customized dataset. The results show that one of our models, the Multiple Convolutional Neural Network including 15 convolutional layers (MCNN15), boosted the state of art accuracy, and it obtained a classification accuracy of 94.04% on the Fashion-MNIST dataset with respect to the literature.

Dataset Reference: Dataset is downloaded from Kaggle Zalando built the Fashion-MNIST dataset to replace the actual MNIST dataset of handwritten digits, powered by renewable energy. The dataset, Fashion-MNIST, contains 28x28 pixel images of fashion product photos. The images have greyscales with 70,000 images, of which 60,000 are training images and 10,000 are validation images. The 10-item class fashion item classification is done using CNN in this paper.

We use the following hyperparameters: lr =0.005, BatchSize =512, and number of epochs =20. The CNN using the Adam



Architecture 1	Architecture 2	Architecture 3	Architecture 4	Architecture 5
only one input layer and two fully connected layers	2 convolutional layers with (2 x 2) filter size and 2 fully connected layers	3 convolutional layers with (2 x 2) filter size and 2 fully connected layers	4 convolutional layers with (2 x 2) filter size and 2 fully connected layers	4 convolutional layers with (3 x 3) filter size and 2 fully connected layers
(1) INPUT:28×28×1 (2) FC:10 Output Classes	(1) INPUT:28×28×1 (2) FC:10 Output Classes	(1) INPUT:28×28×1 (2) FC:10 Output Classes	(1) INPUT:28×28×1 (2) FC:10 Output Classes	(1) INPUT:28×28×1 (2) FC:10 Output Classes
(3) FC:128 Hidden Neurons	(3) CONV2D:2×2 size,64 filters (4) POOL:2×2 size (5) DROPOUT: = 0.25 (6) CONV2D :2×2 size,64 filters (7) DROPOUT: = 0.25 (8) FC:64 Hidden Neurons (9) DROPOUT: = 0.25	(3) CONV2D:2×2 size,64 filters (4) POOL:2×2 size (5) DROPOUT: = 0.25 (6) CONV2D:2×2 size,64 filters (7) POOL:2×2 size (8) DROPOUT: = 0.25 (9) CONV2D :2×2 size,64 filters (10) DROPOUT: = 0.25 (11) FC:64 Hidden Neurons (12) DROPOUT: = 0.25	(3) CONV2D:2×2 size,64 filters (4) POOL:2×2 size (5) DROPOUT: = 0.25 (6) CONV2D:2×2 size,64 filters (7) POOL:2×2 size (8) DROPOUT: = 0.25 (9) CONV2D:2×2 size,64 filters (10) POOL:2×2 size (11) DROPOUT: = 0.25 (12) CONV2D :2×2 size,64 filters (13) DROPOUT: = 0.25 (14) FC:64 Hidden Neurons (15) DROPOUT: = 0.25	(3) CONV2D:3×3 size,32 filters (4) CONV2D:3×3 size,32 filters (4) POOL:2×2 size (5) DROPOUT: = 0.25 (6) CONV2D:3×3 size,64 filters (7) CONV2D:3×3 size,64 filters (8) POOL:2×2 size (9) DROPOUT: = 0.25 (10) FC:512 Hidden Neurons (11) DROPOUT: = 0.5

Figure 14: Different CNN Architecture for image classification

optimizer. From the above parameters, our model trained with decent accuracy resulted in a 0.9572. And the tested accuracy was around 0.9452. The dataset contains lowresolution images with 28x28 pixels.

Shivam S. Kadamet al. [2, 2020], Paper presents application of convolutional neural network for image classification problem. MNIST and Fashion-MNIST datasets used to test the performance of CNN model. Paper presents five different architectures with varying convolutional layers, filter size and fully connected layers. Experiments conducted with varying hyper-parameters namely activation function, optimizer, learning rate, dropout rate and batch size. Results show that selection of activation function, optimizer and dropout rate has impact on accuracy of results. All architectures give accuracy more than 99% for MNIST dataset. Fashion-MNIST dataset is complex than MNIST. For Fashion-MNIST dataset architecture 3 gives better results. Review of obtained results and from literature shows that CNN is suitable for image classification for MNIST and Fashion-MNIST dataset. Paper introduced five CNN architectures to solve image classification problem on MNIST and Fashion-MNIST dataset. Results indicates that any CNN model give 99% accuracy for MNIST dataset. Architecture 3 (3 convolutional layer and 2 fully connected layers) gives better testing accuracy for fashion-MNIST dataset.

Results of Architecture 1: During the experimentation performance of sigmoid, softmax, relu and tanh activation functions are tested. Results are taken with different optimizers namely Adam, Adagrad, Adadelta, SGD and RMSprop.

Results of Architecture 2: For MNIST dataset, best training accuracy and testing accuracy obtained are 98.86% and 98.96% respectively. The best result obtained with 128 batch size, softmax activation function, adam optimizer, 0.25 dropout after each pooling layer, 50 epochs and 2x2 kernel size. For Fashion-MNIST dataset, best training accuracy and testing accuracy obtained are 92.02% and 92.76% respectively. The best result obtained with 128 Batch size, softmax activation function, adam optimizer, 0.25 dropout after each pooling layer, 50 epochs and 2x2 kernel size.

Results of Architecture 3: For MNIST dataset, best training accuracy and testing accuracy obtained are 99.60% and 99.37% respectively. The best result obtained with 128 batch size, softmax activation function, adam optimizer, 0.25 dropout after each pooling layer, 50 epochs and 2x2 kernel size. For Fashion-MNIST dataset, best training accuracy and testing accuracy obtained are 93.09% and 93.56% respectively. The best result obtained with 128 Batch size, softmax activation function, adam optimizer, 0.25 dropout after each pooling layer, 50 epochs and 2x2 kernel size.

Results of Architecture 4*:* In architecture 4, the optimal parameter are 128 batch size, 50 epochs, Softmax activation function, adam optimizer, 2x2 kernel size and 0.25 dropout after each pooling layers. For MNIST dataset, best obtained training accuracy and testing accuracy are 99.02% and 99.03% respectively. For Fashion-MNIST dataset, best obtained training accuracy and testing accuracy are 93.17% and 92.94% respectively.

Results of Architecture 5: The optimal parameter are 128 batch size, 50 epochs, softmax activation function, RMSprop optimizer, (2x2) kernel size and 0.25 dropout after each pooling layers. For MNIST dataset, best obtained training accuracy and testing accuracy are 99.26% % and 99.29% % respectively. For Fashion-MNIST dataset, best obtained training accuracy and testing accuracy are 92.67 % and 92.86 % respectively.

	Fashion-MNIST		MNIST	
	Training accuracy	Testing accuracy	Training accuracy	Testing accuracy
Architecture 1	99.60%	89.65%	99.91%	98.48%
Architecture 2	92.02%	92.76%	98.86%	98.96%
Architecture 3	93.09%	93.56%	99.60%	99.37%
Architecture 4	93.17%	92.94%	99.02%	99.03%
Architecture 5 with Adam optimizer	93.12%	93.56%	99.48%	99.55%
Architecture 5 with RMSprop optimizer	92.67%	92.86%	99.26%	99.29%

Figure 15.Accuracies of Various Architectures

Olivia Nocentiniet al. [3, 2022], In this paper, we proposed to study fashion image classification with four different neural network models to improve apparel image classification accuracy on the Fashion-MNIST dataset.

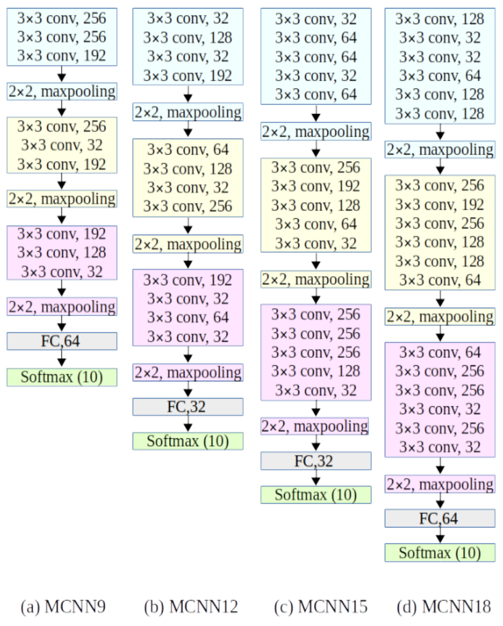
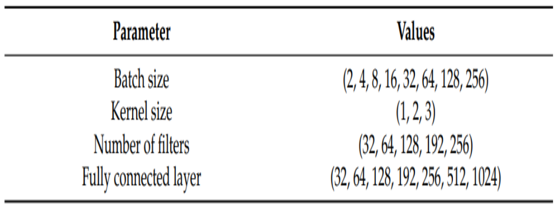


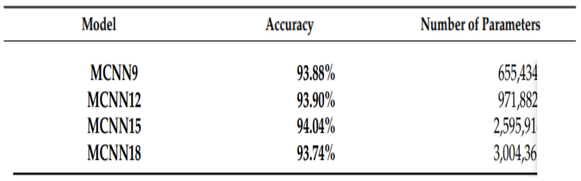
Figure 16: Multiple Convolutional Neural Network models with different convolutional layers and optimized hyperparameters

The hyperparameter configuration with different values is provided and set based on our GPU memory performance. It is randomly used to discover optimal hyperparameter values using Ray tune.



Parameter	Values
Batch size	(2, 4, 8, 16, 32, 64, 128, 256)
Kernel size	(1, 2, 3)
Number of filters	(32, 64, 128, 192, 256)
Fully connected layer	(32, 64, 128, 192, 256, 512, 1024)

Figure 17: Comparison of Accuracy Level Among Concerned Algorithms



Model	Accuracy	Number of Parameters
MCNN9	93.88%	655,434
MCNN12	93.90%	971,882
MCNN15	94.04%	2,595,91
MCNN18	93.74%	3,004,36

Figure 18: Result Obtained

Vaibhav Tiwariet al. [4, 2021], The proposed work implemented the VGG16 model to classify an image into one of the categories like living and non-living that is further classified into several classes like an animal, human, selfie, group photos, place, wallpaper, vehicles, etc. The paper contributes a methodology for a more accurate classification of images instead of image feature extraction or image segmentation. The proposed work established a 99.89% accuracy rate is promising.

The RGB images in this study are processed through five blocks of convolution layers, with three channels or filters in each block. In order to preserve spatial resolution following convolution, the CNN layer is padded and stride fixed at 1. This implies that for 33 filters, there is a pixel-by-pixel padding. The max-pooling layer divides the blocks. In max-pooling, which is run over 22 windows, the stride is 2. The FC (completely connected) layer comes after each of the five convolution layer blocks. Class probabilities in soft max-layers are provided by the output of the last layer.

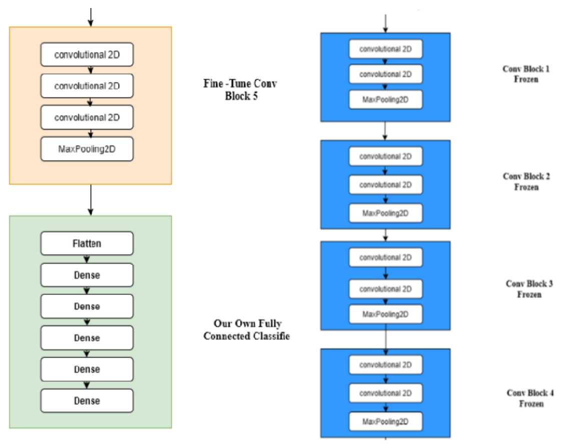


Figure 19: Flow diagram of Modified VGG 16

The datasets used in this work are derived from various sources, and VGG16 is then used to categorically classify each image. The photographs are divided into two categories: living and non-living. Within the first category, there are more subcategories, such as nature, animals, and humans, within which the human class is further divided into group and selfie images. The non-living object was placed in the car. We also obtain 99.89% accuracy with no loss when we use VGG16. While we ran 10 epochs to train our model, it took approximately 17 hours for the provided dataset. It takes one hour and forty-five minutes for one epoch. Future research will focus on utilising VGG19 and

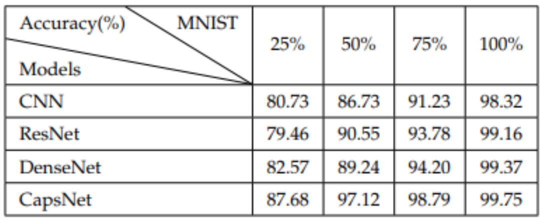
shortening the model's training time. However, the difficulty lies in the large computational time and large availability of open-source massive datasets for model training. Unlike personal computing devices, it might be able to do additional research in this field exclusively using GPU servers.

Feiyang Chen et al. [5, 2019], In this research, we compare four neural networks with varying divisions on the MNIST dataset. Convolutional Neural Networks (CNN), Deep Residual Network (ResNet), and Dense Convolutional Network (DenseNet) are three of them, respectively, and our enhancement of the CNN baseline by bringing Capsule Network (CapsNet) to the image recognition field is the other. We demonstrate that although the earlier models perform rather well in this regard, our retrofitting may be used to get higher performance. With an accuracy percentage of 99.75%, CapsNet produced the best result to date that has been published.

The National Institute of Standards and Technology (NIST) is the source of the MNIST dataset. The training set comprises 250 distinct handwritten numbers, half from the Census Bureau and the other half from high school students. The same percentage of digital handwritten data makes up the test set as well. The MNIST dataset is made up of 10,000 patterns in the testing set and

60,000 images in the training set. Each image has 256 grey levels and a size of 2828 pixels. To test, we split the MNIST dataset into four equal parts: 25%, 50%, 75%, and 100%.

These three models—CNN, ResNet, and DenseNet, respectively—have already shown themselves to perform well in image recognition tests. Furthermore, we discover that there are still certain issues with the conventional CNN model. As a result, we start with CNN as our baseline model and optimise it further by using CapsNet.



Accuracy(%) MNIST / Models	25%	50%	75%	100%
CNN	80.73	86.73	91.23	98.32
ResNet	79.46	90.55	93.78	99.16
DenseNet	82.57	89.24	94.20	99.37
CapsNet	87.68	97.12	98.79	99.75

Figure 20: Results of experiment on divided datasets

III. Problem Definition

Underfitting: If both the training and validation accuracy are low, and both losses are high, the model is underfitting. This means the model is not complex enough to capture the underlying patterns in the data.

Overfitting: If the training accuracy is high and the training loss is low, but the validation accuracy is significantly lower and the validation loss is higher, the model is overfitting. This means the model is too complex and is capturing noise in the training data, which doesn't generalize well to new, unseen data.

Goodfit: If the training and validation accuracy are both high and close to each other, and the training and validation losses are low and close to each other, the model is well-fitted to the data.

- Training accuracy starts high and continues to increase, eventually reaching nearly 100%.
- Validation accuracy improves initially but starts to plateau and even slightly decreases in later epochs.
- Training loss continuously decreases, while validation loss starts increasing after a certain number of epochs.

1. Based on Basic CNN Implementation analysis, it seems that the model is **Overfitting**. The model performs exceptionally well on the training data but does not generalize as well to the validation data. Here are some indicators:

- The Training Accuracy is very high (almost 100%) while the Validation Accuracy Plateaus around 91-92%.
- The Training Loss keeps Decreasing, but the Validation Loss starts to Increase, indicating that the model is starting to memorize the training data rather than learning generalizable patterns.

2. Without batch normalization, CNN models can face several challenges - **Internal Covariate Shift** refers to the phenomenon where the distribution of inputs to a layer changes during training as the parameters of the previous layers change.**Batch Normalization** addresses internal covariate shift by normalizing the inputs to each layer. This involves adjusting and scaling the inputs to have a mean of zero and a variance of one.

3. Without dropout, the model is more likely to memorize the training data rather than generalize well to new, unseen data.This leads to a significant gap between training and validation accuracy.Without dropout, the model can become too powerful and memorize the training data. It focuses on the specific patterns, noises, and details in the training images rather than learning general features that can be applied to new images.

- ***Solutions for Overfitting:*** The paper

[1] does not discuss the use of regularization methods to prevent overfitting.

- ***Regularization Techniques:*** "Implement and compare different regularization techniques" (e.g., dropout, L2 regularization, batch normalization) to prevent overfitting and improve the generalization of CNN models on the Fashion MNIST dataset."
- ***Data Augmentation:*** Increase the diversity of the training data by applying transformations like rotations, flips, and scaling.
- ***Reduce Model Complexity:*** Simplify the model by reducing the number of layers, parameters, or using pruning techniques.
- ***Cross-Validation:*** Use techniques like k-fold cross-validation to ensure the model generalizes well across different subsets of the data.

IV. Proposed Work

A. Data source

Fashion MNIST, the dataset contains images of clothing items. The Fashion MNIST dataset is designed to be a more challenging benchmark for machine learning algorithms, providing a set of 70,000 28x28 grayscale images of 10 different types of clothing items, split into 60,000 training images and 10,000 test images.

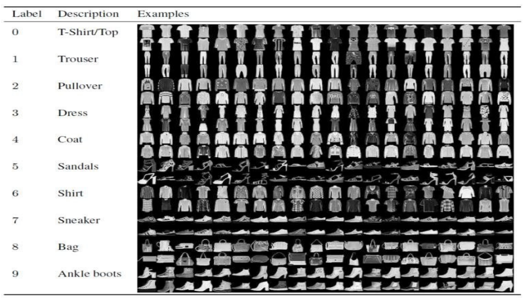


Figure 21: Fashion-MNIST Dataset

B. Convolutional Neural Network (CNN)

Table 1: Size of Train/ Validation/Test Data, Batch Size and Epoch

x_train.shape	(60000, 28, 28)	60k Data of image With 28 rows 28 columns
x_train.shape, x_val.shape, y_train.shape, y_val.shape	((48000, 28, 28, 1), (12000, 28, 28, 1), (48000,), (12000,))	80% train, 20% validation
x_test.shape	(10000, 28, 28)	10k Data of image With 28 rows 28 columns
Batch_Size=512 512 Images Per Batch	48000 / 512	93.75~94
Thus - Epoch 100	94 Images in 1 Epoch	Processing 94 / 94

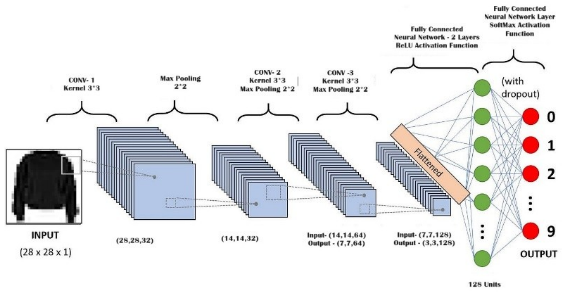

Fully Connected Neural Network Layer SoftMax Activation Function
Fully Connected Neural Network - 2 Layers ReLU Activation Function
CONV- 1 Kernel 3*3
Max Pooling 2*2
CONV- 2 Kernel 3*3 Max Pooling 2*2
CONV -3 Kernel 3*3 Max Pooling 2*2
(with dropout)
Flattened
0
1
2
9
INPUT
(28 x 28 x 1)
(28,28,32)
(14,14,32)
Input- (14,14,64) Output - (7,7,64)
Input- (7,7,128) Output - (3,3,128)
128 Units
OUTPUT

Figure 22. Proposed Work Flow – Basic CNN

C. Enhanced Convolution Neural Network

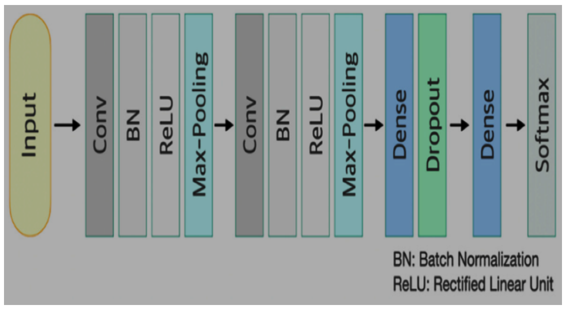


Figure 23: Enhanced CNN

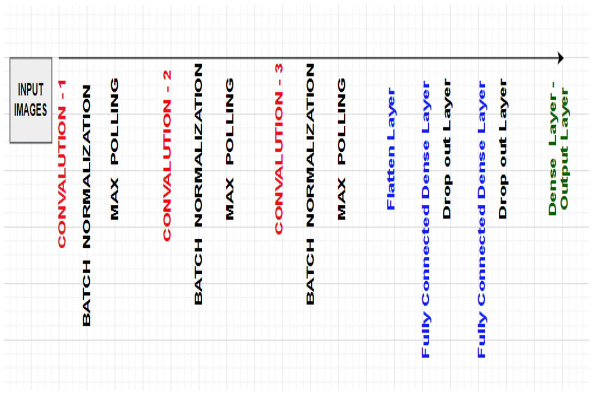


Figure 24: Proposed Work Flow – Enhanced CNN

D. Work Flow

1. Importing Liabraries
2. Loading Dataset
3. Retrieves the dimensions of the training – testing data and the corresponding labels
4. Feature Scaling
5. Spliting the x_train and y_train datasets into two sets: Training Set and Validation Set
6. Defines a Convolutional Neural Network (CNN) model for image classification using Keras, with layers for convolution, max pooling, flattening, and dense connections, ending with a softmax activation for multi-class output.
7. Enhanced CNN is designed with several techniques commonly used to improve CNN performance and training efficiency:

- Several convolutional layers with increasing filter sizes (32, 64, 128) and kernel sizes (3x3).
- Batch Normalization layers after each convolutional layer to stabilize and accelerate training.
- MaxPooling2D layers to downsample feature maps.
- Flatten layer to convert 2D feature maps into a 1D vector.
- Dense (fully connected) layers with ReLU activation functions.
- Dropout layers with a dropout rate of 0.5 to prevent overfitting by randomly setting a fraction of input units to zero during training.
- Finally Model is Compiled and Tested

V. Implementation Work

A. Fashion-MNIST Dataset

All experiments in this study were conducted on a laptop computer w5.2ith Intel i5 processor, 8 GB of DDR3 RAM by using google colaboratory. A research tool for machine learning education and research is called Colaboratory. Jupyter notebook environments operate fully in the cloud and don't require any setup. With Colaboratory, you can use your browser to develop and run code, store and share studies, and access robust computational resources—all without paying a dime.

Python – Python 3

Data Source - https://www.kaggle.com/datasets/zalando-research/fashionmnist/code

https://www.kaggle.com/datasets/zalando-research/fashionmnist/code

Files Data Set Size : 72 MB (Image Dataset)

Content

An MNIST-like dataset of 70,000 28x28 labeled fashion images

x_train.shape, y_train.shape

((60000, 28, 28), (60000,))

x_test.shape, y_test.shape

((10000, 28, 28), (10000,))

B. Basic CNN

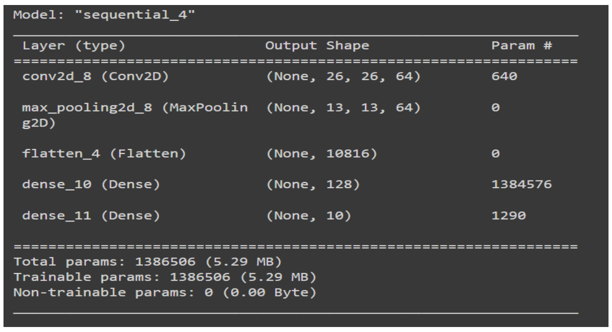

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_8 (Conv2D)           (None, 26, 26, 64)        640

 max_pooling2d_8 (MaxPoolin  (None, 13, 13, 64)        0
 g2D)

 flatten_4 (Flatten)         (None, 10816)             0

 dense_10 (Dense)            (None, 128)               1384576

 dense_11 (Dense)            (None, 10)                1290

=================================================================
Total params: 1386506 (5.29 MB)
Trainable params: 1386506 (5.29 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 25 : Basic CNN Layers and Parameters

C. Enhanced CNN

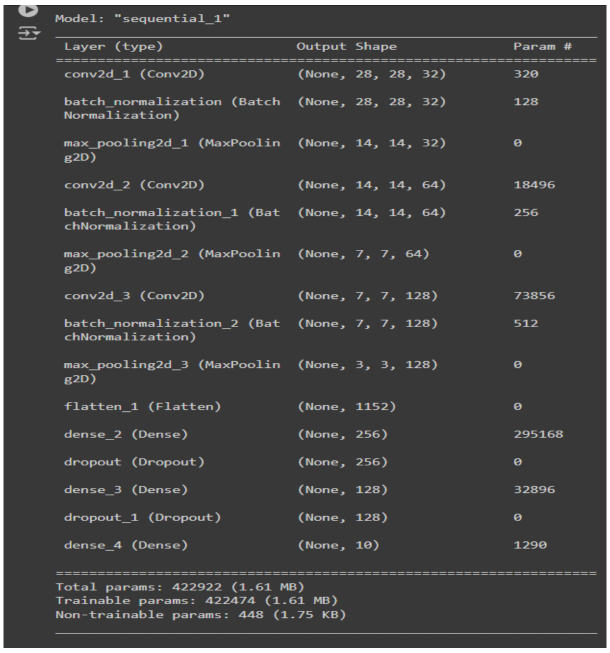
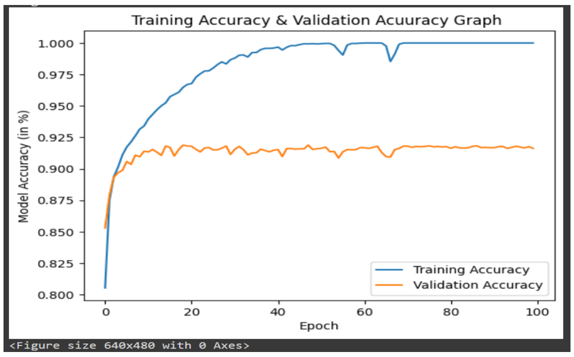

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_1 (Conv2D)           (None, 28, 28, 32)        320

 batch_normalization (Batch  (None, 28, 28, 32)        128
 Normalization)

 max_pooling2d_1 (MaxPoolin  (None, 14, 14, 32)        0
 g2D)

 conv2d_2 (Conv2D)           (None, 14, 14, 64)        18496

 batch_normalization_1 (Bat  (None, 14, 14, 64)        256
 chNormalization)

 max_pooling2d_2 (MaxPoolin  (None, 7, 7, 64)          0
 g2D)

 conv2d_3 (Conv2D)           (None, 7, 7, 128)         73856

 batch_normalization_2 (Bat  (None, 7, 7, 128)         512
 chNormalization)

 max_pooling2d_3 (MaxPoolin  (None, 3, 3, 128)         0
 g2D)

 flatten_1 (Flatten)         (None, 1152)              0

 dense_2 (Dense)             (None, 256)               295168

 dropout (Dropout)           (None, 256)               0

 dense_3 (Dense)             (None, 128)               32896

 dropout_1 (Dropout)         (None, 128)               0

 dense_4 (Dense)             (None, 10)                1290

=================================================================
Total params: 422922 (1.61 MB)
Trainable params: 422474 (1.61 MB)
Non-trainable params: 448 (1.75 KB)
_________________________________________________________________
```

Figure 26 : Enhanced CNN Layers and Parameters

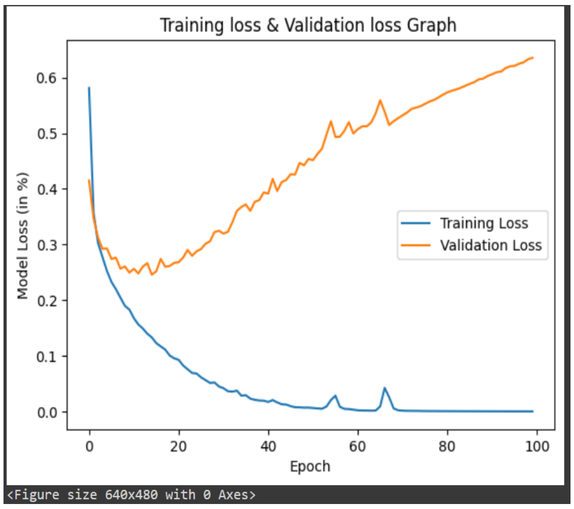
VI. RESULTS

A. Basic CNN Training & Validation Results



```
<Figure size 640x480 with 0 Axes>
```

Figure 27: Basic CNN Training & Validation Accuracy



```
<Figure size 640x480 with 0 Axes>
```

Figure 28: Basic CNN Training & Validation Loss

B. Enhanced CNN Training and Validation Results

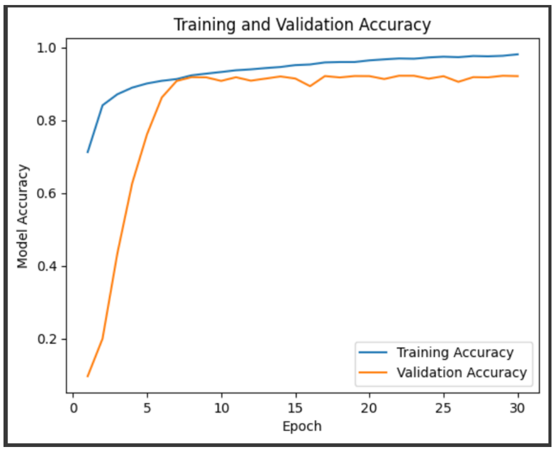
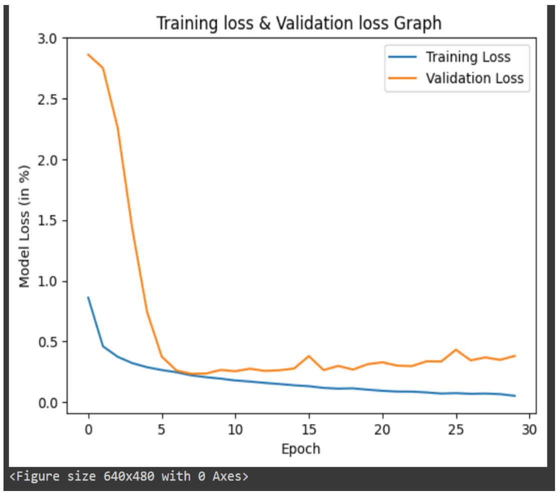


Figure 29 : Enhanced CNN Training & Validation Accuracy



```
<Figure size 640x480 with 0 Axes>
```

Figure 30: Enhanced CNN Training & Validation Loss

C. Basic CNN Layers and Parameters

Table 2: Proposed Work Results - Basic CNN Layers & Parameters

Basic CNN	Total Layers 6 = Input layer + Convolutional layer + Max pooling layer + Flatten layer + Dense layer 1 + Dense layer 2	
	Input Images	Batch_Size=512 512 Images Per Batch
Layer 1	Convolutional Layer (`Conv2D`)	
	Number of filters	64
	Kernel size	3x3
	Image Input shape	(28, 28, 1) (28x28 grayscale image)
	Bias terms (one bias per filter)	64

	Formula for Output Dimensions of a Convolutional Layer
	output_height = ((input_height - kernel_height + 2 * padding) / stride) + 1
	output_width = ((input_width - kernel_width + 2 * padding) / stride) + 1
		output_height = ((28 - 3 + 2 * 0) / 1) + 1 = 26
		output_width = ((28 - 3 + 2 * 0) / 1) + 1 = 26
		(None, 26, 26, 64)

	Total Parameters: (kernel_height * kernel_width * input_channels + 1) * filters	Total Parameters = (3 * 3 * 1 + 1) * 64 = 640
Layer 2	Max Pooling Layer (MaxPooling2D)	No parameters to learn in max pooling layers. They only perform a fixed operation
	Output Shape	(None, 13, 13, 64)
Layer 3	Flatten Layer (The Flatten layer reshapes the output of the previous layer into a one-dimensional array)	No parameters to learn in the flatten layer. It only reshapes the data.
	Output	13 * 13 * 64 = 10816.
Layer 4	Dense 1 (Fully Connected) Layer	
	Input shape: Output of the Flatten Layer	10816
	Number of units	128
	Bias terms:	128
	Total parameters: (input_shape + 1) * units	(10816 + 1) * 128 = 1,384,576 Or (10816 input*128 next layer neuron)+128 bias = 1384576
Layer 5	Dense 2 (Fully Connected) Layer (Output Layer)	
	Input shape	128
	Number of units: (Classification task with 10 classes)	10
	Bias terms: 10	10
	Total parameters: (input_shape + 1) * units	Total parameters = (128 + 1) * 10 = 1290 Or (128 Neuron * 10 Next Layer Neuron) + 10 bias = 1290
Total Parameters	Summing up the parameters from all layers	Convolutional layer: 640 Dense layer 1: 1,384,576 Dense layer 2: 1290 Total Trainable Parameters = 640 + 1,384,576 + 1290 = **1,386,506**
Additionally, there are no non-trainable parameters in this model.		

D. Enhanced CNN Layers and Parameters

Table 3: Proposed Work Results - Enhanced CNN Layers and Parameters

Enhanced CNN	Input Images	Batch_Size=512 512 Images Per Batch The batch size for the first layer is set to 512. This means that during training, the first layer of the neural network will process 512 samples at a time before updating the weights based on the computed loss.
Layer 1	**Conv2D_1** **First CN Layer**	Input - An image of size 28x28x1 (Grayscale).
		Filter = 32 filters: Each filter is of size 3x3.
		For each filter, there are weights associated with each element of the filter.
		Number of weights = (filter_height * filter_width * input_channels) * number_of_filters (3 * 3 * 1) * 32 = 288
		Number of biases = 32 (one bias per filter)
		Total parameters = Number of weights + Number of biases = 288 + 32 = 320
		Output Shape = (None, 28, 28,32)
Layer 2	**Batch Normalization Layer**	INPUT = Since the output shape of the previous layer, **conv2d_4**, is **(None, 28, 28, 32),** there are 32 feature maps,
		Scale (ã) and Shift (â) for each feature map. because each feature map has both scale and shift parameters, resulting in a total of 128 parameters (64 scale parameters and 64 shift parameters).
		Parameters = 2(Scale (ã) and Shift (â)) = 128 Since there are 32 feature maps, and each has its own pair of ã and â parameters: Total ã parameters = 32 (one for each feature map). Total â parameters = 32 (one for each feature map). Total parameters = 32 (ã) + 32 (â) = 64 + 64 = 128.
Layer 3	**MaxPooling2D:**	Pool Size: (2, 2) Output: (None, 14, 14, 32)
Layer 4	**Conv2D Second CN Layer**	Filters: 64 filters of size 3x3 Output Shape: (None, 14, 14, 64) Parameters: Weights per filter=kernel width×kernel height×input channels=3×3×32 (3×3×32)×64 Filters +64 Bias =**18496** (weights + biases)
Layer 5	**Batch Normalization Layer**	Parameters: 2×64=128 (2 parameters per feature map: scale and shift) because each feature map has both scale and shift parameters, resulting in a total of **256** parameters (128 scale parameters and 128 shift parameters).

Layer 6	**MaxPooling2D Layer (2nd pooling layer):**	Pool Size: (2, 2) Output Shape: (None, 7, 7, 64)
Layer 7	**Conv2D Third Layer**	Filters: 128 filters of size 3x3 Output Shape: (None, 7, 7, 128) Parameters: (3×3×64)×128Filters+128Bias =**73856** (weights + biases)
Layer 8	**BatchNormalization Layer (3rd BN layer):**	Parameters: Scale (ã) and Shift (â) for each feature map. Parameters per feature map = 2 (ã + â) Parameters: 2×128=256 (scale and shift for each feature map) Total parameters = **512** Output Shape: (None, 7, 7, 128)
Layer 9	**MaxPooling2D Layer (3rd pooling layer):**	Pool Size: (2, 2) Output Shape: (None, 3, 3, 128)
Layer 10	**Flatten Layer:**	input shape (None, 3, 3, 128), after flattening would be a one-dimensional vector of size 3 * 3 * 128 = **1152** Output Shape: (None, 1152)
Layer 11	**Dense Layer (Fully Connected)**	Units: 256 Output Shape: (None, 256) Parameters: 1152×256+256=**295168**(weights + biases)
	Dropout Layer	Dropout Rate: 0.5
Layer 12	**Dense Layer (Fully Connected)**	Units: 128 Output Shape: (None, 128) Parameters: We need to calculate the number of weights and biases. Number of weights: 256 (coming from the previous layer) multiplied by 128 (current layer units) Number of biases: 128 (one bias per unit) Total parameters = Number of weights + Number of biases = (256 * 128) + 128 = 32896
	Dropout Layer	Dropout Rate: 0.5
Layer 13	**Dense Layer Output Layer**	Units: 10 Activation Function: Softmax Output Shape: (None, 10) Parameters: 128×10+10=1290 (weights + biases)
	Total Trainable Parameters:	422,474 = (1.61 MB) = 320+0+128+18,496+0+256+73,856+0+512+0+295,168+0+32,896+0+1,290
	Total Non-trainable Parameters:	448 (1.75 KB) = 128+256+512

VII. Conclusions And Future Work

A. Conclusion

In conclusion, this research Based on the comparative analysis of Basic and Advanced CNN architectures for the Fashion-MNIST dataset, it is evident that the enhancements in the Advanced CNN significantly improve the performance of fashion image classification. The Basic CNN, consisting of 6 layers, achieved a high training accuracy of 100% but displayed signs of overfitting with a validation accuracy plateauing at around 91.7%. In contrast, the Advanced CNN, with its 13 layers incorporating Batch Normalization and Dropout, demonstrated superior generalization capabilities, achieving a final validation accuracy of 93.3%.

The Advanced CNN's controlled use of overfitting techniques, such as Batch Normalization and Dropout, resulted in more consistent validation accuracy, indicating better adaptation to new, unseen data. This model also maintained a lower validation loss compared to the Basic CNN, further underscoring its improved performance.

In conclusion, the study confirms that the integration of advanced techniques in CNN architectures can significantly enhance the accuracy and generalization of fashion image classification tasks. The findings suggest that employing such sophisticated CNN models can be highly beneficial for applications in the fashion industry, including clothing recognition, search, and recommendation systems. This research highlights the potential of advanced CNN architectures in achieving more reliable and accurate classification results using the Fashion-MNIST dataset.

B. Future Work

Future work to improve fashion image classification using CNN architectures could include:

1. ***Deeper Networks:*** Experiment with more complex architectures like ResNet or DenseNet for better feature extraction.

2. ***Transfer Learning:*** Use pre-trained models on larger datasets and fine-tune them on Fashion-MNIST to leverage their powerful feature extraction.

3. ***Data Augmentation:*** Apply techniques like rotation, scaling, and flipping to increase the diversity of training data and improve model robustness.

4. ***Hyperparameter Tuning:*** Optimize parameters like learning rate, batch size, and dropout rate using grid search or random search for better performance.

5. ***Ensemble Methods:*** Combine predictions from multiple models to improve accuracy and reduce errors.

6. ***Attention Mechanisms:*** Integrate attention layers to help the model focus on important parts of the images.

7. ***Synthetic Data:*** Use GANs (Generative Adversarial Networks) to create additional training images, enhancing model performance.

8. ***Regularization Techniques:*** Implement advanced regularization methods to reduce overfitting and improve generalization.

9. ***Advanced Optimizers:*** Try different optimizers like AdamW or RAdam for

potentially better training efficiency and performance.

10. ***Model Interpretability:*** Develop tools to better understand what the models are learning, which can help in improving and refining the models.

These steps can help in making CNN models more accurate and robust for fashion image classification. By exploring these directions, future research can build upon the current work and continue to enhance the performance and applicability of CNNs for fashion image classification.

REFERENCES:

[1] A. Vijayaraj, P. T. Vasanth Raj, R. Jebakumar, P. Gururama Senthilvel, N. Kumar, R. Suresh Kumar, and R. Dhanagopal, "Deep Learning Image Classification for Fashion Design", Hindawi Wireless Communications and Mobile Computing Volume 2022, Article ID 7549397, 13 pages https://doi.org/10.1155/2022/7549397.

[2] Shivam S. Kadam, Amol C. Adamuthe, and Ashwini B. Patil, "CNN Model for Image Classification on MNIST and Fashion-MNIST Dataset", Volume 64, Issue 2, 2020, Journal of Scientific Research, Institute of Science, Banaras Hindu University, Varanasi, India. DOI: 10.37398/JSR.2020.640251.

[3] Olivia Nocentini, Jaeseok Kim, Muhammad Zain Bashir and Filippo Cavallo, "Image Classification Using Multiple Convolutional Neural Networks on the Fashion-MNIST Dataset", Sensors 2022, 22, 9544. https://doi.org/10.3390/s22239544.

[4] Vaibhav Tiwari, Chandrasen Pandey, Ankita Dwivedi, Vrinda Yadav, "Image Classification Using Deep Neural Network", 2020 2nd International Conference on Advances in Computing, Communication Control and Networking (ICACCCN) | 978-1-7281-8337-4/20/$31.00 ©2020 IEEE | DOI: 10.1109/ICACCCN51052.2020.9362804.

[5] Feiyang Chen, Nan Chen, Hanyang Mao, Hanlin Hu, "Assessing Four Neural Networks on Handwritten Digit Recognition Dataset (MNIST)". Chuangxinban Journal of Computing, June 2018.

[6] Alisson Steffens Henrique, Anita Maria da Rocha Fernandes, Rodrigo Lyra, Valderi Reis Quietinho Leithardt, Sérgio D. Correia, Paul Crocker, Rudimar Luis Scaranto Dazzi, Advances in Science, Technology and Engineering Systems Journal Vol. 6, No. 1, 989-994 (2021).

[7] Shobhit Bhatnagar, Deepanway Ghosal, Maheshkumar H. Kolekar,Classification of Fashion Article Images using Convolutional Neural Networks, 2017 Fourth International Conference on Image Information Processing (ICIIP).

[8] Shyava Tripathi, Rishi Kumar, Image Classification using small Convolutional Neural Network, "978-1-5386-5933-5/19/$31.00 c 2019 IEEE".

[9] Nadia Jmour, Sehla Zayen, Afef Abdelkrim, "Convolutional Neural Networks for image classification", 978-1-5386-4449-2/18/$31.00 ©2018 IEEE.

[10] M Manoj krishna, M Neelima, M Harshali, M Venu Gopala Rao."Image classification using Deep learning", International Journal of Engineering

& Technology, 7 (2.7) (2018) 614-617.

[11] Edmira Xhaferra, Elda Cina, Luçiana Toti, "Classification of Standard FASHION MNIST Dataset Using Deep Learning Based CNN Algorithms", 2022 International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT) | 978-1-6654-7013-1/22/ $31.00 ©2022 IEEE | DOI: 10.1109/ ISMSIT56059.2022.9932737.

[12] Mohammed Kayed, Ahmed Anter, Hadeer Mohamed, "Classification of Garments from Fashion MNIST Dataset Using CNN LeNet-5 Architecture", 2020 International Conference on Innovative Trends in Communication and Computer Engineering (ITCE 2020), Aswan, Egypt, 8-9 Feb. 2020.

[13] Fei Ren, JiTian Qian, HongSheng Li,Ye Liu, JiaHui Zhang, PengFei Gao, "Research on garment image classification algorithm based on machine learning", 2021 3rd International Academic Exchange Conference on Science and Technology Innovation (IAECST) | 978-1-6654-0267-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/ IAECST54258.2021.9695531.

[14] Shivam Singh, "Image Classification Using Deep Learning" School of Computing Science and Engineering Galgotias University Plot No.-2, Sector-17A, Yamuna Expressway, Gautam Buddha Nagar, Greater Noida, Uttar Pradesh, India.

[15] Jatin Kayasth, "Image Classification using CNN", Jatin Kayasth Yeshiva University - Katz School of Science and Health, 11 November 2022.

[16] Chao Duana, Panpan Yinb, Yan Zhic, Xingxing Lid, "Image Classification of Fashion-mnist Data Set Based on VGG Network", 2019 2nd International Conference on Information Science and Electronic Technology (ISET 2019).

[17] Atul Sharmaa, Gurbakash Phonsab, "Image Classification Using CNN", International Conference on Innovative Computing and Communication (ICICC 2021).

* * * * *