Dissertation on
Classification of National Flags using Convolutional Neural Network (CNN)

Thesis submitted towards partial fulfilment of the requirements for the degree of

Master of Technology in IT (Courseware Engineering)

Submitted by
Karobi Sarkar

EXAMINATION ROLL NO.: **M4CWE23011**
UNIVERSITY REGISTRATION NO.: **160386 of 2021-2022**

Under the guidance of
Prof. Dr. Matangini Chattopadhyay

School of Education Technology
Jadavpur University

Course affiliated to
Faculty of Engineering and Technology
Jadavpur University
Kolkata-700032
India

2023

M.Tech. IT (Courseware Engineering)
Course affiliated to
Faculty of Engineering and Technology
Jadavpur University
Kolkata, India

CERTIFICATE OF RECOMMENDATION

This is to certify that the thesis entitled **"Classification of National Flags using Convolutional Neural Network (CNN)"** is a bonafide work carried out by **Karobi Sarkar** under our supervision and guidance for partial fulfillment of the requirements for the degree of **Master of Technology in IT (Courseware Engineering)** in School of Education Technology, during the academic session 2021-2023.

SUPERVISOR
School of Education Technology
Jadavpur University,
Kolkata-700 032

DIRECTOR
School of Education Technology
Jadavpur University,
Kolkata-700 032

DEAN - FISLM
Jadavpur University,
Kolkata-700 032

M.Tech. IT (Courseware Engineering)
Course affiliated to
Faculty of Engineering and Technology
Jadavpur University
Kolkata, India

CERTIFICATE OF APPROVAL **

This foregoing thesis is hereby approved as a credible study of an engineering subject carried out and presented in a manner satisfactory to warranty its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not endorse or approve any statement made or opinion expressed or conclusion drawn therein but approve the thesis only for purpose for which it has been submitted.

Committee of final examination for evaluation of Thesis

** Only in case the thesis is approved.

DECLARATION OF ORIGINALITY AND COMPLIANCE OF ACADEMIC ETHICS

I hereby declare that this thesis contains literature survey and original research work by the undersigned candidate, as part of her **Master of Technology in IT (Courseware Engineering)** studies during academic session **2021-2023**.

All information in this document has been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by this rule and conduct, I have fully cited and referenced all materials and results that are not original to this work.

NAME: **Karobi Sarkar**

EXAMINATION ROLL NUMBER: **M4CWE23011**

THESIS TITLE: Classification of National Flags using Convolutional Neural Network (CNN)

SIGNATURE: DATE:

ACKNOWLEDGEMENT

I wish to express my whole hearted and deep gratitude to my respected supervisor **Prof. Dr. Matangini Chattopadhyay,** Director & Professor, School of Education Technology, Jadavpur University, for her valuable guidance, constant support and encouragement throughout my research work.

This has been a precious opportunity for me not only to gain knowledge and skill but also to learn much more about approaches, attitudes towards work and interpersonal relationship.

I am sincerely thankful and indebted to Dr. Saswati Mukherjee, Dr. Susovan Jana, Mr. Joydeep Mukherjee for their constant encouragement and continuous valuable suggestions throughout my research work.

Finally, but most importantly, my friends and my family members have always been there for me during the ups and downs, sharing my excitement and frustration. Their love and understanding have helped me to prepare this thesis successfully.

_____

KAROBI SARKAR

CONTENTS

EXECUTIVE SUMMARY

Various countries are recognized for their unique symbols and a combination of special colors. Flag classification using image processing is a challenging task due to a variety of colors and textures. Sometimes, the colors and layouts of some flags are very similar.

Therefore, we build a custom dataset which is a collection of flag images for 110 countries. Furthermore, we have applied different augmentation techniques to increase the number of images in our dataset. The augmentations are 10° rotation, 20° rotation, 30° rotation, 40° rotation, 50° rotation, 60° rotation, 70° rotation, 80° rotation, 90° rotation, adding ‘salt & pepper’ noise, ‘Gaussian’ noise and translation (20, 20).

Each class contains 13 images after applying image augmentation. In total, our custom dataset contains 1430 images. We have proposed a flag classification technique based on Convolutional Neural Network (CNN). The proposed CNN contains 19 layers. The proposed model's validation accuracy and testing accuracy are 98.18% and 97.58%, respectively.

We have calculated class wise accuracy. It has been observed that 97 classes out of 110 classes have been detected with 100% accuracy, whereas the previous classification approaches of flag using Gray-Level Co-Occurrence Matrix (GLCM) features (Contrast, Correlation, Energy, Homogeneity) and Support Vector Machine(SVM) classifier showed 19.70% overall accuracy. The Discriminant analysis classifier with the same GLCM features showed 32.73% accuracy. The Decision Tree classifier showed 32.42% accuracy, and the Naïve Bayes classifier showed 32.42% accuracy with the same GLCM features.

LIST OF FIGURES

LIST OF TABLES

Chapter 1

1.0 Introduction:

Digital image processing is the computation of digital images through several algorithms. It is essential in many applications, such as automatic visual inspection systems, Bio-medical imaging techniques, moving object tracking, intelligent transportation system, and face recognition. As a result, image processing is a rapidly growing technology and forms a core research area.

Flag recognition systems are part of image processing applications. Researchers have proposed flag identification using various data modeling techniques, features, and classifiers. We have proposed of a flag classification system using a convolution neural network (CNN).

1.1 Background Concepts:

Image processing is used to find out various patterns and aspects in images and pattern recognition is used for Computer-aided medical diagnosis, handwriting analysis,image recognition. In recent years, the huge amount of digital image data have increased exponentially with the help of latest internet technology.In high resolution field of digital image processing increasing the contrast of planetary details and reducing the noise. Research scientists in various nations of the world created digital species, new intelligent approaches in image processing applications.

1.2 Problem statement:

Classification of National Flags using Convolutional Neural Network (CNN)

1.3 Objective:

- To study existing flag classification approach using CNN
- To collect images for making datasets
- To find out a suitable model for the flag classification.
- To implement the proposed classification approach in MATLAB

1.4 Concept and Problem Analysis:

A convolutional neural network(CNN) has been widely applied to image data. CNN helps to detect objects, classify images and recognize patterns by extracting features from input image data. Figure 1 shows an example of image classification using CNN.

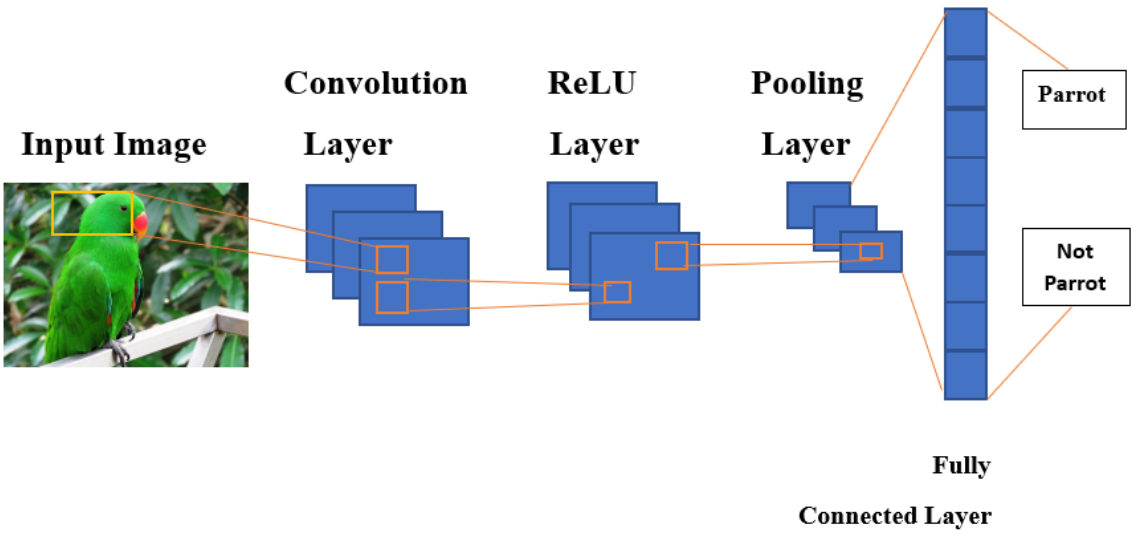


Figure 1: Image classification using CNN

In Figure 1 shows the image classification using CNN.

The most significant component in CNN architecture is the convolutional layer. The convolutional layer consists of a collection of convolutional filters. These convolutional filters are also called kernels. The input image represents N-Dimensional metrics. This N-Dimensional metric convolves with these convolutional filters and generates the output feature map as shown in Figure 2.

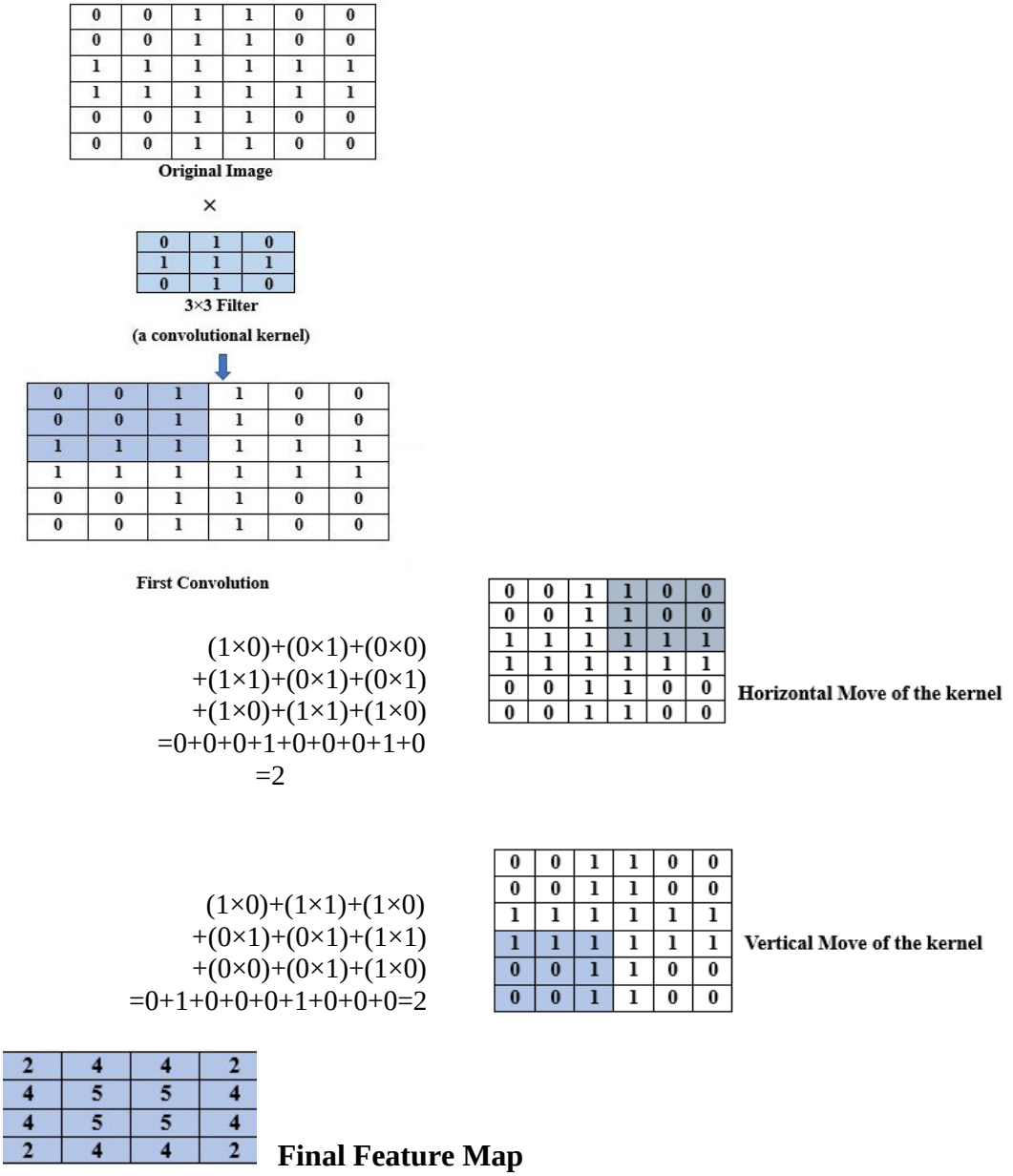


Figure 2: A simple demonstration of 2D convolution over a 2D image

In Figure 2 shows a simple demonstration of 2D convolution over a 2D image.

The number of pixels that escapes in a single move is called stride. The use of larger strides reduces the feature map but increases the chance of missing small features. After extracting a basic feature map, a Rectified Linear Unit(ReLU) has been added for nonlinearity.

Pooling is used for reducing the feature map with minimal information loss. The maximum value from each region is used for max pooling (see Figure 3).

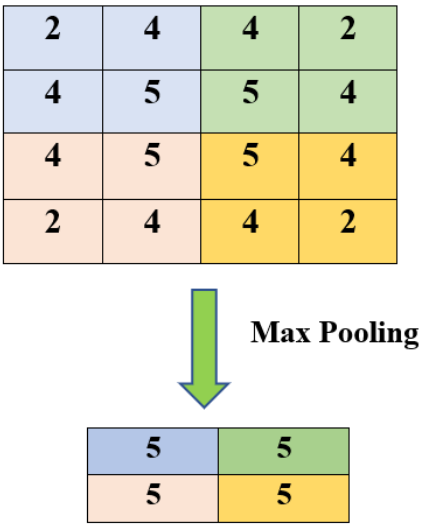


Figure 3: Demonstration of Max Pooling

In Figure 3, we have chosen the maximum value from each 2×2 block. So, we get a 2×2 matrix from a 4×4 feature map.

The Fully connected layer, is located at the end of each CNN architecture. Each neuron is connected to all neurons of the previous layer. In CNN, a loss function is used to compare actual and predicted classes.

1.5 Organization of the Thesis:

In this paper, Chapter 2 describes Literature Survey, Chapter 3 describes Proposed Approach in CNN, Chapter 4 discusses experimentations and results Chapter 5 presents experimentations and results, and Chapter 6 concludes our work and future scope.

.

Chapter 2

2.0 Literature Survey:

National Flags often appear in various places, such as stadiums, gatherings, parades, in airports. Sometimes, it is challenging to identify flags correctly. This problem arises because of light intensity variations, vertical and horizontal measurements of different colour patches, and other colour shades of flags. Sometimes, this problem is complicated because of local deformation, different types of noise, and partial obstruction. Therefore, nowadays, many researchers work on identifying flags with the help of deep-learning algorithms and image-processing technology. Different classifiers and data modelling techniques have been proposed for flag identification by different researchers.

Ming Gu et al. [1] proposes a flag identification algorithm based on Convolutional Neural Network(CNN). They use a 1×1 convolutional layer structure by replacing the traditional full-connected layer to improve the network performance. Their flag detection rule is based on a multi-scale matching strategy. They have achieved an accuracy of 91.71%.

Yahia Said et al.[2] proposes an approach based on color-based features and a Convolutional Neural Network(CNN) with a unique local context neural network. They are adding a local context neural network to enhance the localization task and a color-based descriptor to improve the identification task. They have achieved a mean Average Precision of 89.5% and a real-time processing speed.

Avishikta et al. [3] proposes an approach for recognizing the flags of various countries worldwide. First, they partition the images into 4×3 non-overlapping cells and extract statistical features from the color channels of each cell. Then, these statistical features are combined and generate feature vectors. These feature

vectors help to discriminate between the flags. Finally, they used a K-NN classifier for class discrimination with k set to 1 and Euclidean distance as a metric. Through this approach, they get 99% accuracy in flag recognition.

Shou-Fang Wu et al. [4] proposes a flag-detection approach for multi-nation flag with instance segmentation in the wild which is based on data augmentation and Mask-RCNN PointRend. They build a multi-nation flag image dataset which is based on data aggregation, annotation, hallucination, and augmentation. They combine the trained binary Mask-RCNN segmentation weights for fine-tuning with the new multi-nation classifier.

Soe Moe Myint et al. [5] proposes a feature extraction-based recognition system for Myanmar's national flag using principal component analysis(PCA). Principal component analysis(PCA) is a statistical approach which is used for reducing the number of features in Myanmar National flags.The recognition accuracy of their proposed system is from 90% to 95%.

Eduardo Hart et al. [6] proposed an interactive flag recognition system for identifying flags in photos of natural scenes. In their proposed system, users are asked to crop the region of the flag images. The system does not automatically identify the flag to its respective country. But, the lists of the countries flags of colour similarity have been shown.
The accuracy of this system is from 82% to 93%.

Ivan Zatezalo et al. [7] proposes a functional mobile application called "FlagAFlag". This application helps to identify the National Flags of different countries. Convolutional Neural Network is chosen for this architecture. The testing is done on 120 images, and more than 95% images are correctly classified.

Riadh Ayachi et al. [8] proposes a traffic sign detection application by using the deep learning technique. This deep learning technique is based on convolutional neural networks(CNN). They build a dataset that contains 10,500 images (built from 73 traffic sign classes). Their proposed application achieves a mean average precision of 84.22% on this proposed dataset.

Mouna Afif et al. [9] proposes a new indoor object detector based on a deep convolutional neural network-based framework called “RetinaNet”. ResNet, DenseNet, and VGGNet are used for evaluation to improve detection performance and processing time. They obtained 84.61% mAP as detection precision.

Susovan Jana et al. [10] proposes a state-of-the-art deep-learning technique for detecting rotten fruit and vegetable. To classify the rotten and fresh fruit and vegetable from captured images of fruits and vegetables, a convolutional neural network(CNN) architecture has been proposed. They created four different sets of images from the actual dataset. Data have been chosen randomly from there for training and testing purposes. The fine-tuned AlexNet is tested on the same each of the datasets which are tested for their proposed CNN model. Features extraction is more complex in AlexNet than in the proposed CNN model. Nevertheless, the proposed CNN model accurately identifies rotten fruits and vegetables.

Chapter 3

3.0 Proposed Approach:

Convolutional neural networks (CNNs) are the most utilized and most popular deep learning architectures. We have used the state-of-the-art architectures of CNN.

Here, a convolutional neural network architecture has been used to overcome the challenges of various countries' flag identification. Furthermore, this network architecture is sequential. Figure 4. shows the detailed architecture of our proposed CNN model.

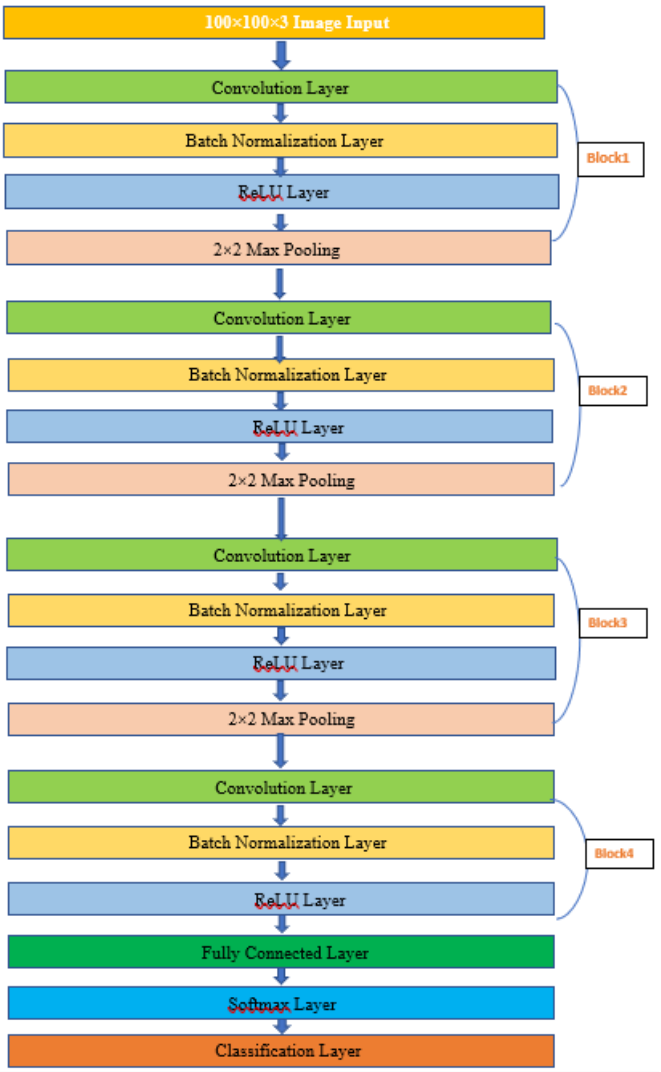


Figure 4: The architecture of the proposed CNN model

In Figure 4 shows our proposed CNN model with 19x1 Layer array with layers.

The first layer is Image Input of 100x100x3 images with 'zerocenter' normalization, the second layer is Convolution 8 (3x3x3 convolutions with stride [1 1] and padding 'same'),the third layer is Batch Normalization layer with 8 channels, the fourth layer is ReLU layer ,the fifth layer is Max Pooling layer. We are used 2x2 max pooling with stride [2 2] and padding [0 0 0 0]

The sixth layer is Convolution layer of 16 3x3x8 convolutions with stride [1 1] and padding 'same', the seventh layer is Batch Normalization layer with 16 channels. The eighth layer is ReLU Layer.The ninth layer is Max Pooling Layer. 2x2 max pooling with stride [2 2] and padding [0 0 0 0] are used.

The tenth layer is Convolution layer. 32, 3x3x16 convolutions with stride [1 1] and padding 'same' are used.The eleventh layer is Batch Normalization Layer. Batch normalization with 32 channels. The twelveth layer is ReLU layer.

The thirteen layer is Max Pooling layer. 2x2 max pooling with stride [2 2] and padding [0 0 0 0] are used.

The fourteen layer is Convolution layer. 64, 3x3x32 convolutions with stride [1 1] and padding 'same' are used.The fifteenth layer is Batch Normalization layer with 64 channels. The sixteenth layer is ReLU layer. The seventeenth layer is Fully Connected layer. 111 fully connected layer are used. The eighteenth layer is SoftMax layer. The nineth layer is Classification Output layer.

We follow different steps in our proposed Convolutional neural network(CNN) model.

These steps are:

- Image channels
- Convolution
- Batch normalization
- ReLU
- Pooling
- Fully Connected
- Softmax
- Classification

Image channels:

The first step of the process is to make the image compatible with the proposed CNN algorithm. So, we find a way to represent the image numerically. The image is represented using its pixel, and each pixel within the image is mapped to a number between 0 to 255. Each number represents a colour ranging from 0 (white) to 255 (black).

In our proposed model, we work with a dataset of RGB images. Each image of the dataset is 100×100 size, and a 3-dimensional array is used. Each pixel from the image is represented by its corresponding pixel values in the red, green, and blue channel.

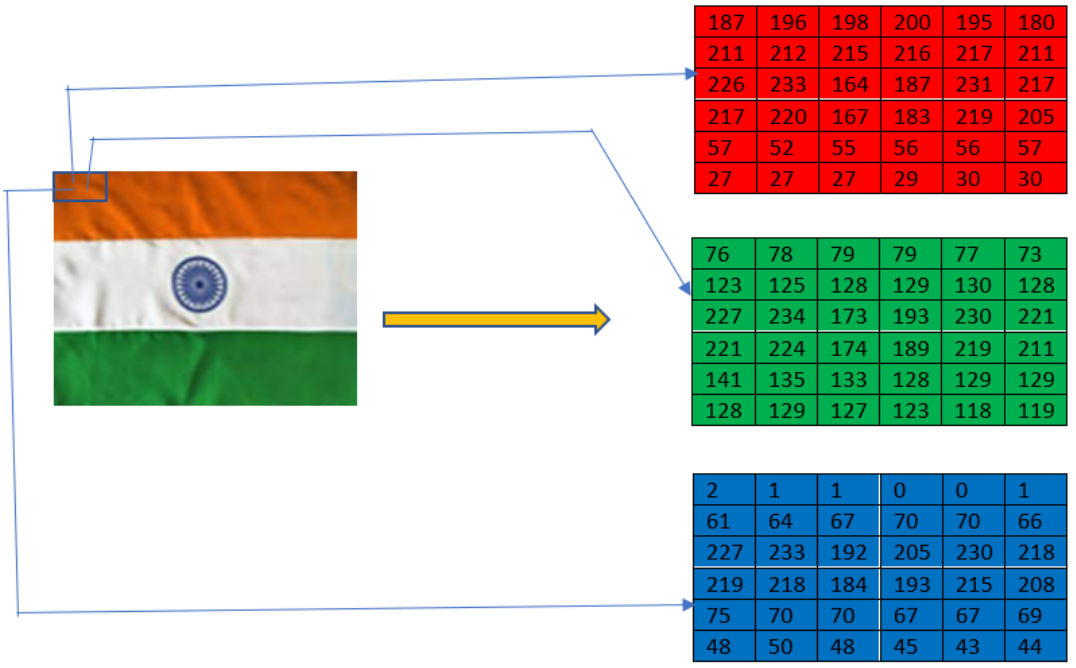


Figure 5: The image represents with values of RGB channels

Each image is represented as a 3-dimensional array, with each channel representing red, green, and blue values. It is shown in the Figure 5 as an example.

Convolution Layer:

Now, the above image is represented as a combination of numbers. In the next step, it is needed to identify the key features within the image. A method by which we can extracted the features from an image, is known as convolution.

In Convolutional Neural Networks (CNN), convolutions are applied to extract the prominent features within the images.

A filter or kernel is used for extracting key features from an image.This filter is strided over the input 2-dimentional array.This array contains the corelation of the image with respect to the filter that was applied.The output array is referred as feature map.

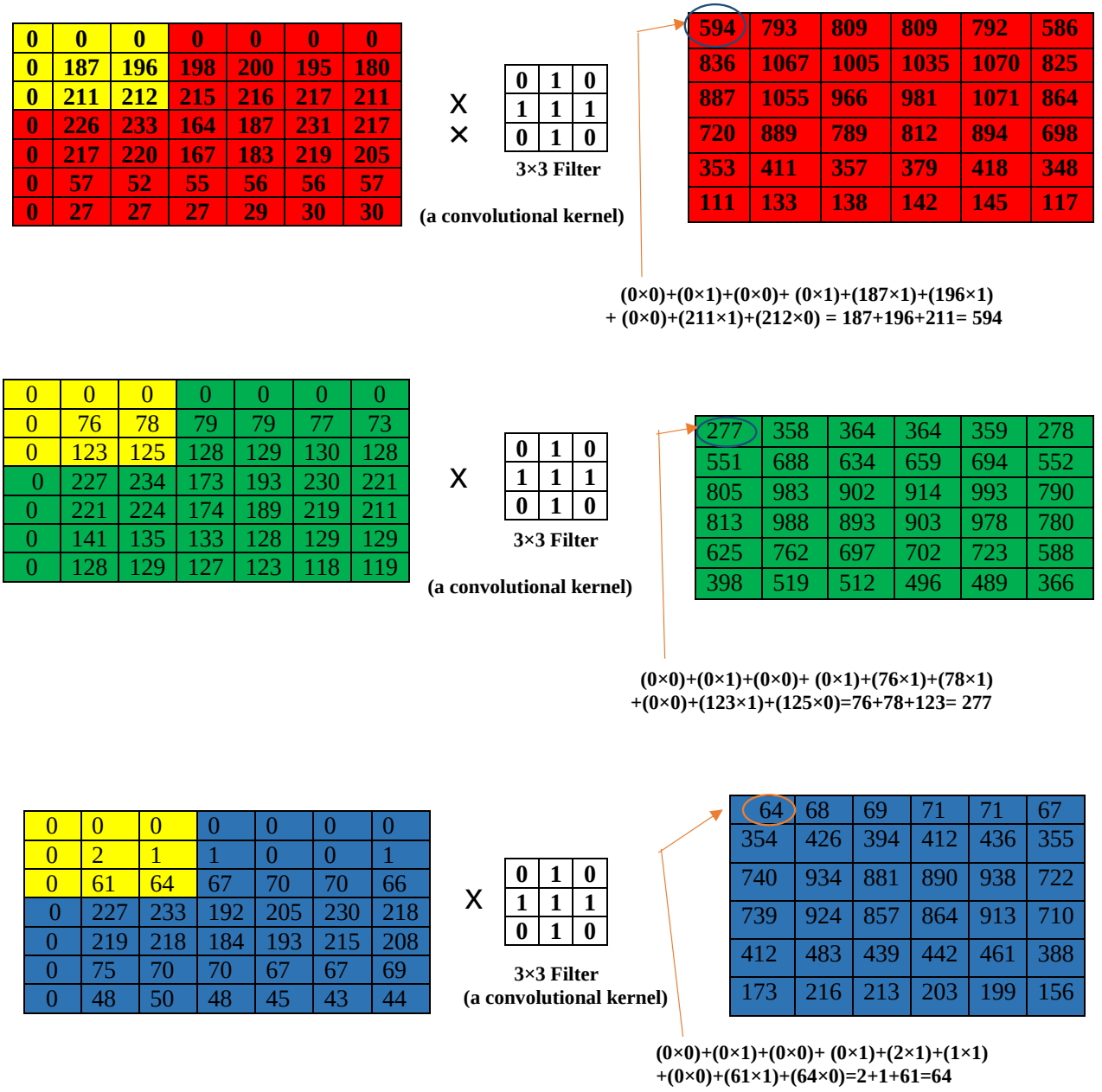


Figure 6: 2D convolution over a 2D RGB image

In the Figure 6, the filter size 3×3 is applied to the input image of size 6×6.

The input layer receives 100×100 RGB color images with zero center normalization. The input layer adds next to the convolution layer. The layer contains eight numbers of 3×3 convolution filters with stride 1 pixel in the X-direction and 1 pixel in the Y-direction, and zero padding.

The padding is the process of adding dummy pixels on the different sides of the image to generate the feature map of the same dimension as the image. The convolution extracts features from the input image as long as the training progresses.

Batch normalization layer:

After convolution layer, batch normalization layer and a ReLU layer are added. The batch normalization layer normalizes the extracted features from the convolution layer. It helps to speed up the training process.

ReLU Layer:

ReLU layer uses nonlinearity with a nonlinear activation function, as mentioned in equation (1).

$$f(x) = \begin{cases} -x, & x < 0 \\ x, & x \geq 0 \end{cases} \quad (1)$$

Pooling:

A 2×2 max-pooling layer is added next to the ReLU layer. stride 2 pixel in the X-direction and 2 pixel in the Y-direction, and zero padding.

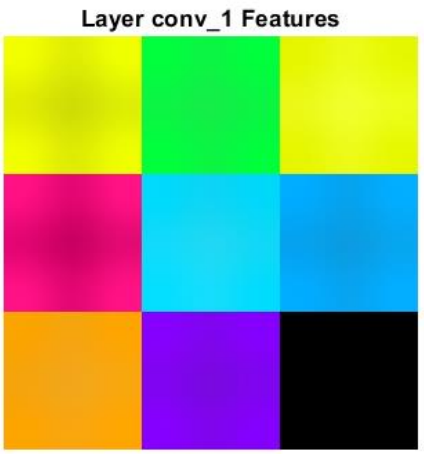


Figure 7: Features map after processing of Convolutional Layer in Block 1

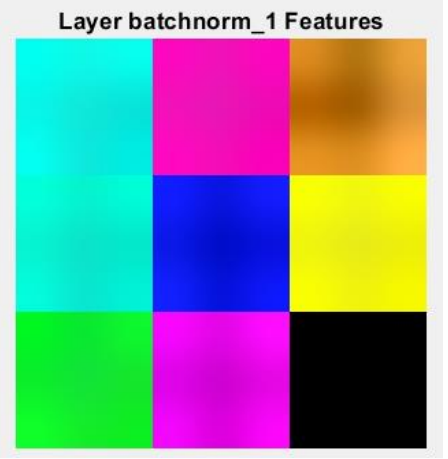


Figure 8: Features map after Processing of Batch normalization layer in Block 1

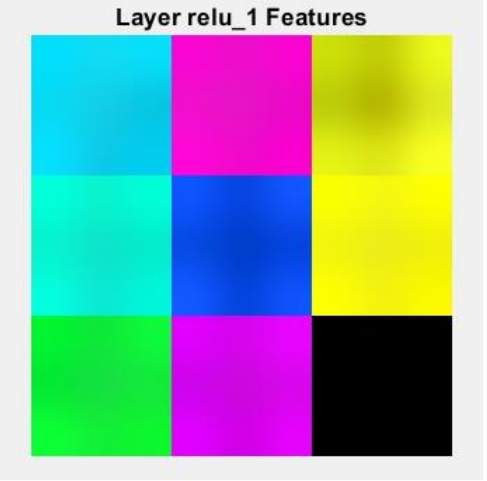


Figure 9: Features map after processing of ReLU layer in Block 1

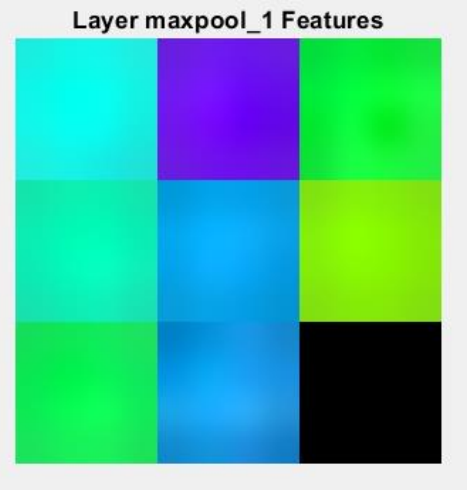


Figure 10: Features map after processing of max-pooling layer in Block 1

These images mostly contain colours, which indicates that the filters at this layer in Figure 7, Figure 8, Figure 9, Figure 10 are colour filter.

One after another, similar types of three blocks are added sequentially. Only the number of filters in the convolution layer and the number of channels in batch normalization layers have been doubled.

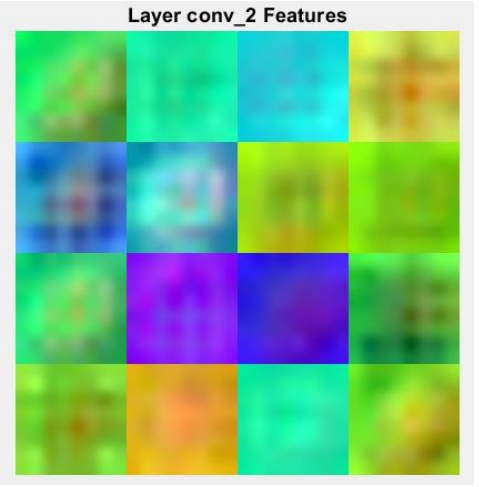


Figure 11: Features map after processing of Convolutional layer in Block 2

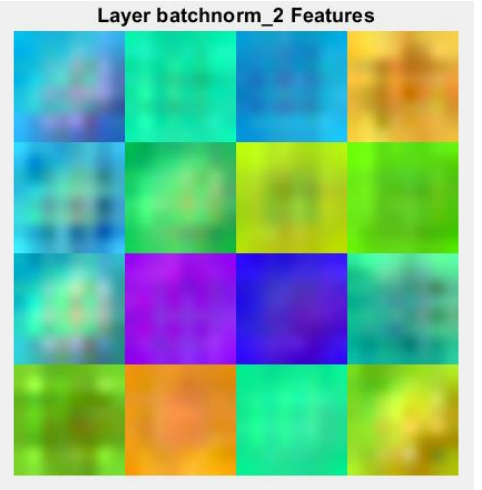


Figure 12: Features map after processing of Batch normalization layer in Block 2

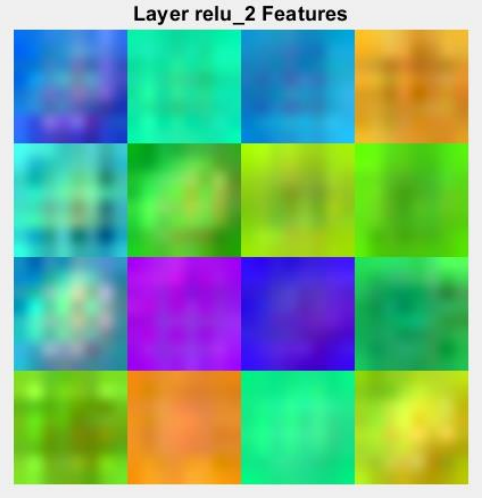


Figure 13 : Features map after processing of ReLU layer in Block 2

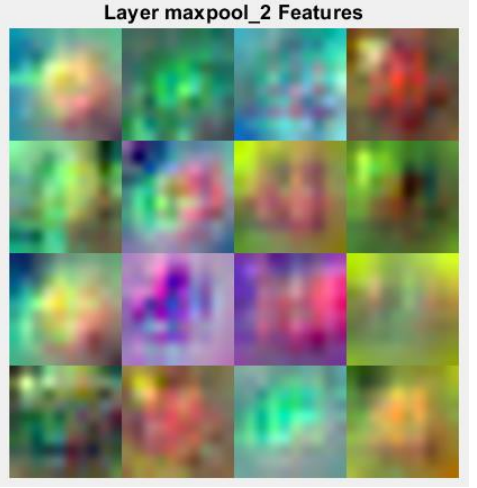


Figure 14: Features map after processing of max-pooling layer in Block 2

These images mostly contain edges and colours, which indicates that the filters at this layer in Figure 11, Figure 12, Figure 13, Figure 14 are edge detectors and colour filter.

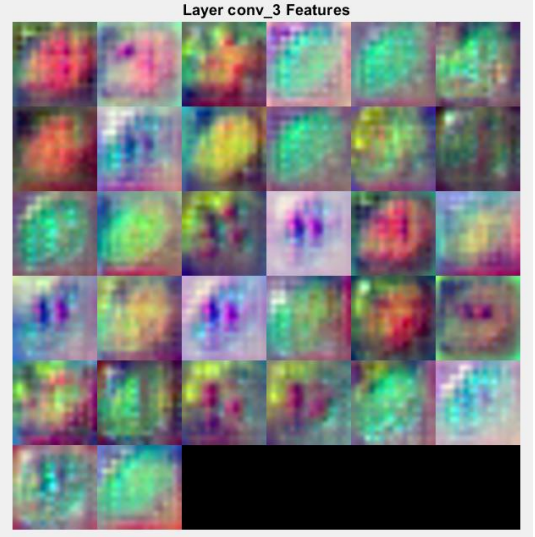


Figure 15: Features map after processing of Convolutional Layer in Block 3

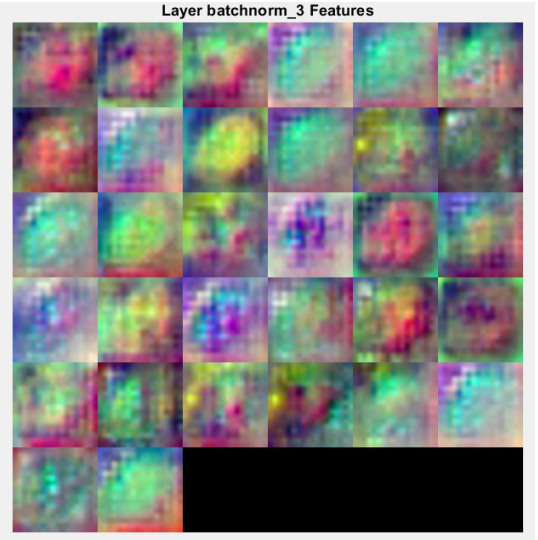


Figure 16: Features map after processing of Batch normalization layer in Block 3

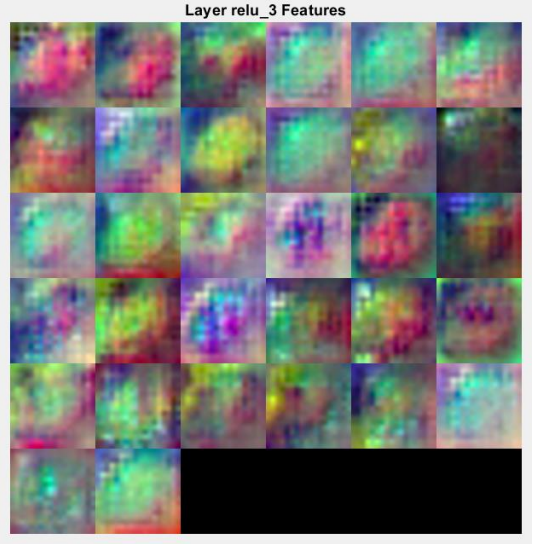


Figure 17: Features map after processing of ReLU layer in Block 3

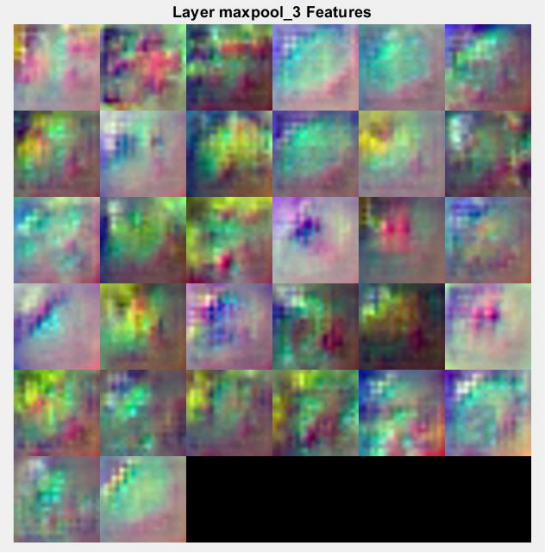


Figure 18: Features map after processing of max-pooling layer in Block 3

In Figure 15, Figure 16, Figure 17, Figure 8 it is shown that Filters of this layer detect more complex patterns than the first, second, third convolution layers.

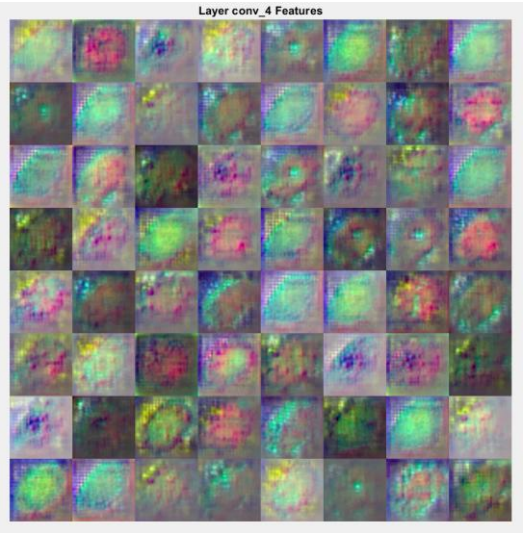


Figure 19: Features map after processing of Convolutional Layer in Block 4

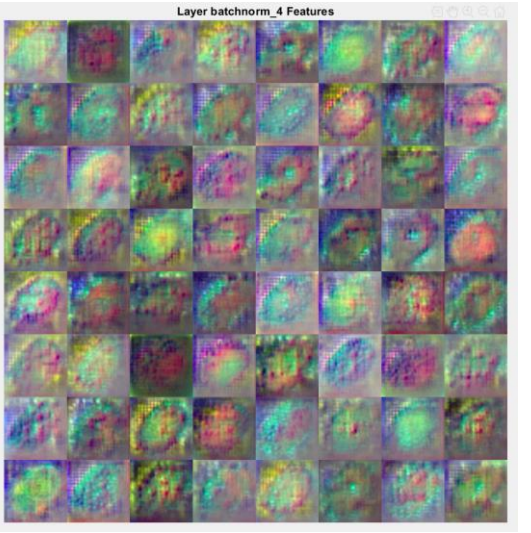


Figure 20 : Features map after processing of Batch normalization layer in Block 4

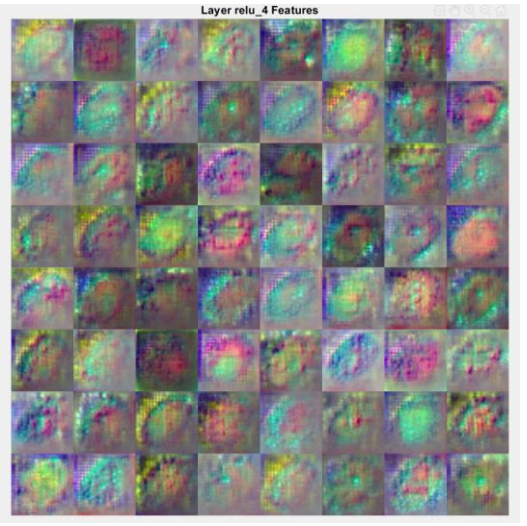


Figure 21: Features map after processing of ReLU layer in Block 4

In the above Figure 19, Figure 20, Figure 21 shows that the layers which are deeper into the network yield more detailed filters which have learned more complex patterns and textures.

In the final block max-pooling layer is replaced by the fully connected layer. Then a softmax layer is added before the final classification layer. The softmax layer is used to increase the probabilities. It helps to normalize the output of the fully connected layer. The classification Layer uses these probabilities and predicts the class of unknown test samples.

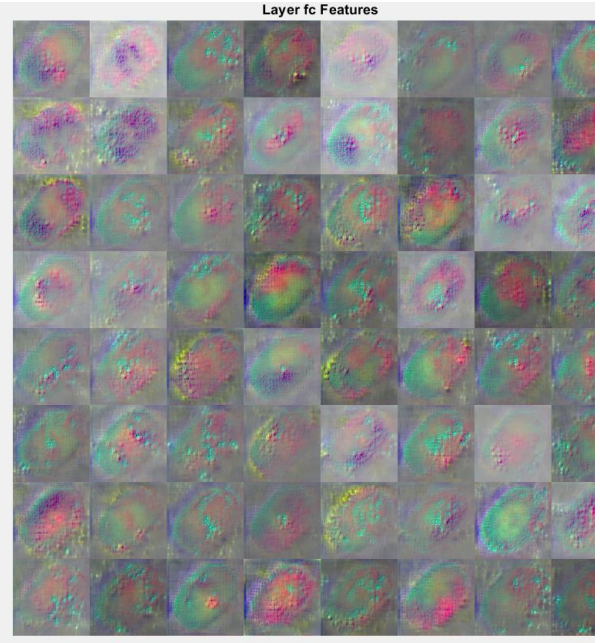


Figure 22: Features map after processing of Fully Connected Layer

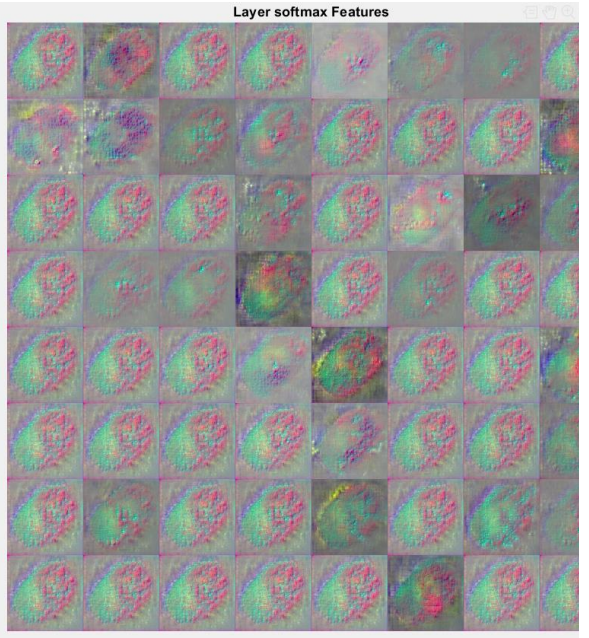


Figure 23: Features map after processing of Softmax Layer

In the Figure 22 and Figure 23 it is shown that images are strongly activated the selected classes.

Chapter 4:

Experimentations and results:

4.0 Experimental setup:

We are used Matlab version 2018, Windows 11 operating system and 16.0 GB RAM. Processor is used Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz 1.99 GHz.

4.1 Dataset

We have used a dataset of 1430 flag images for 110 countries to test the performance of the proposed model. Each class contains 13 images. This dataset has been downloaded from the Kaggle repository [11]. The images are in RGB color format and stored in PNG format. The sample flags are shown in this table.

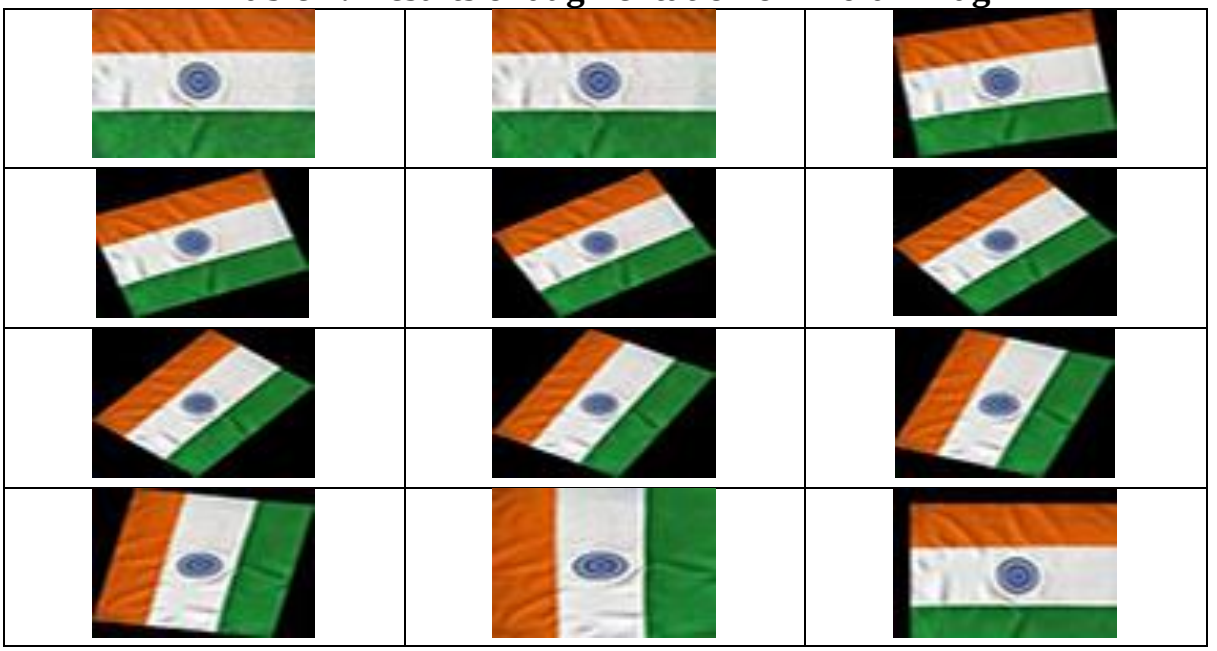
 Afghanistan	 Albania	 Algeria	 Andorra	 Angola	 Anguilla	 Antigua & Barbuda	 Argentina	 Armenia	 Australia
 Austria	 Azerbaijan	 Bahamas	 Bahrain	 Barbados	 Bangladesh	 Belarus	 Belgium	 Belize	 Benin
 Bermuda	 Bhutan	 Bolivia	 Bosnia & Herzegovina	 Botswana	 Brazil	 Brunei	 Bulgaria	 Burkina Faso	 Burundi
 Cambodia	 Cameroon	 Canada	 Cape Verde	 Chile	 China	 Christmas Island Union	 Colombia	 Congo	 Czech Republic
Democratic Republic of the Congo	Djibouti	Dominica	Dominican Republic	Ecuador	Egypt	Equatorial Guinea	Eritrea	Fiji	Germany
Grenada	Guatemala	Haiti	India	Iraq	Kazakhstan	Kenya	Kiribati	Korea	Kosovo
Kyrgyzstan	Lebanon	Liechtenstein	Malaysia	Malta	Marshall Islands	Mexico	Moldova	Mongolia	Montenegro
			Namibia	Nepal					Panama

Morocco	Mozambique	Myanmar			New Zealand	Nicaragua	Oman	Pakistan	
 Papua New Guinea	 Paraguay	 Philippines	 Portugal	 Rwanda	 Saint Kitts and Nevis	 Saint Lucia	 San Marino	 Saudi Arabia	 Serbia
 Slovakia	 Slovenia	 Solomon Islands	 South Africa	 Spain	 Sri Lanka	 Sweden	 Tanzania	 The Central African Republic	 Trinidad and Tobago
 Turkmenistan	 Tuvalu	 UAE	 Uganda	 United States of America	 Uruguay	 Vanuatu	 Vatican City	 Western Sahara	 Zambia
 Zimbabwe									

Table1: Country names along with flag images

We build a custom dataset of flag images for 110 countries. Furthermore, we have applied different image augmentation techniques to increase the number of images in our dataset. The augmentations are 10° rotation, 20° rotation, 30° rotation, 40° rotation, 50° rotation, 60° rotation, 70° rotation, 80° rotation, 90° rotation, adding 'salt & pepper' noise, 'Gaussian' noise and translation (20 pixels along X-axis, 20 pixels along Y-axis). Each class contains 13 images after applying the above image augmentation techniques.

Table 2: Results of augmentation on Indian Flag



4.2 Training Phase

The training data is chosen randomly, and eight images are taken out of thirteen images for each class for training purposes. We have taken two images randomly for validation purposes. The images are resized into 100 × 100 before putting it into CNN. The training parameters are vital in building a highly accurate CNN model. The optimized values of the training parameters are listed in Table3.

Table 3: Training Parameters and corresponding values

Parameters	Optimized values
InitialLearnRate	0.01
MaxEpochs	50
Shuffle	every-epoch
ValidationFrequency	10

After defining the neural network structure, specify the training options. Train the neural network using stochastic gradient descent with momentum (SGDM) with an initial learning rate of 0.01. Set the maximum number of epochs to 50.

An epoch is a full training cycle on the entire training data set. It monitors the neural network accuracy during training by specifying validation data and validation frequency. Shuffle the data every epoch. The software trains the neural network on the training data and calculates the accuracy on the validation data at regular intervals during training. The validation data is not used to update the neural network weights. In the next step, train the neural network using the architecture defined by layers, the training data, and the training options.

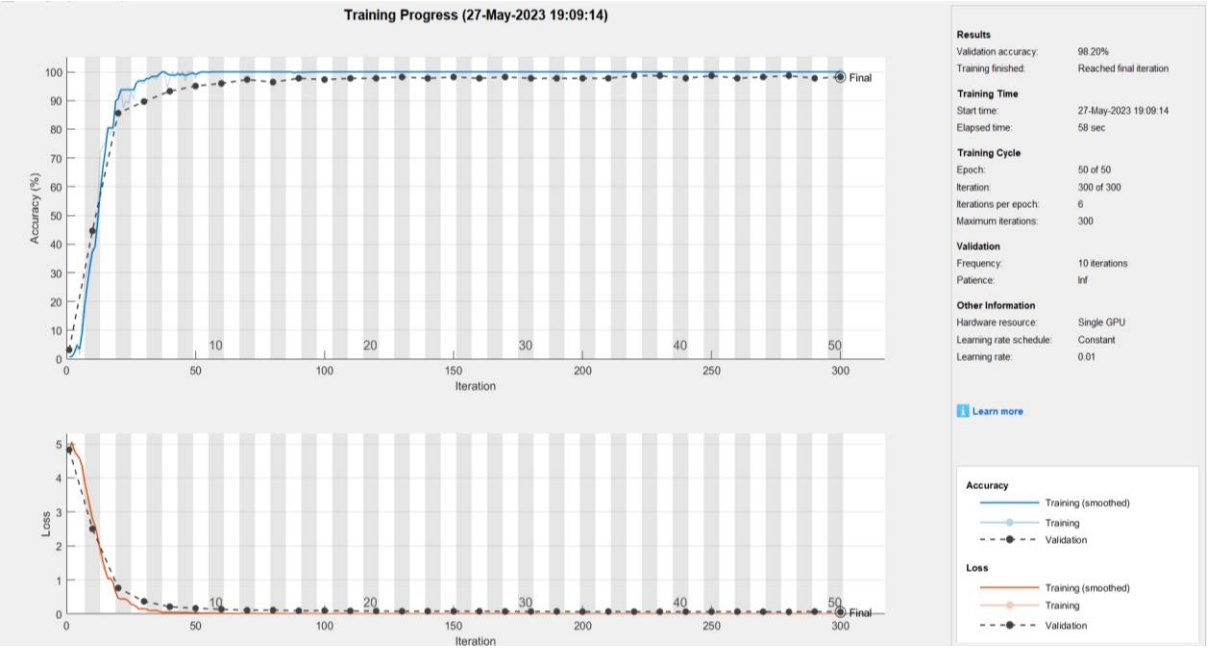


Figure 24: Train Neural Network Using Training Data
(1st run of training of our proposed CNN model,MaxEpochs-50)

In the Figure 24, we are trained our proposed model with MaxEpochs-50 (in the first run).

Table 4: Training Progress of our proposed model and MaxEpochs

MaxEpochs	Validation Accuracy	Time Taken
50	98.20%	58 sec

In the Table 4, we are trained our proposed model with MaxEpochs-50 and get the validation accuracy 98.20%.

The training progress plot shows the mini-batch loss and accuracy and the validation loss and accuracy. The loss is the cross-entropy loss. The accuracy is the percentage of images that the neural network classifies correctly.

4.3 Testing Phase

We split the dataset into training, validation, and testing sets for each class. Therefore, we randomly take three out of thirteen images for each class for testing purposes. Classification accuracy for each class is shown in figure. We have used equation (1) for accuracy calculation, where TP = True Positive, TN = True Negative, FP = False Positive, FN = False Negative

$$\text{Accuracy} = \frac{\text{TP}+\text{TN}}{\text{TP}+\text{FP}+\text{FN}+\text{TN}} \times 100 \qquad (1)$$

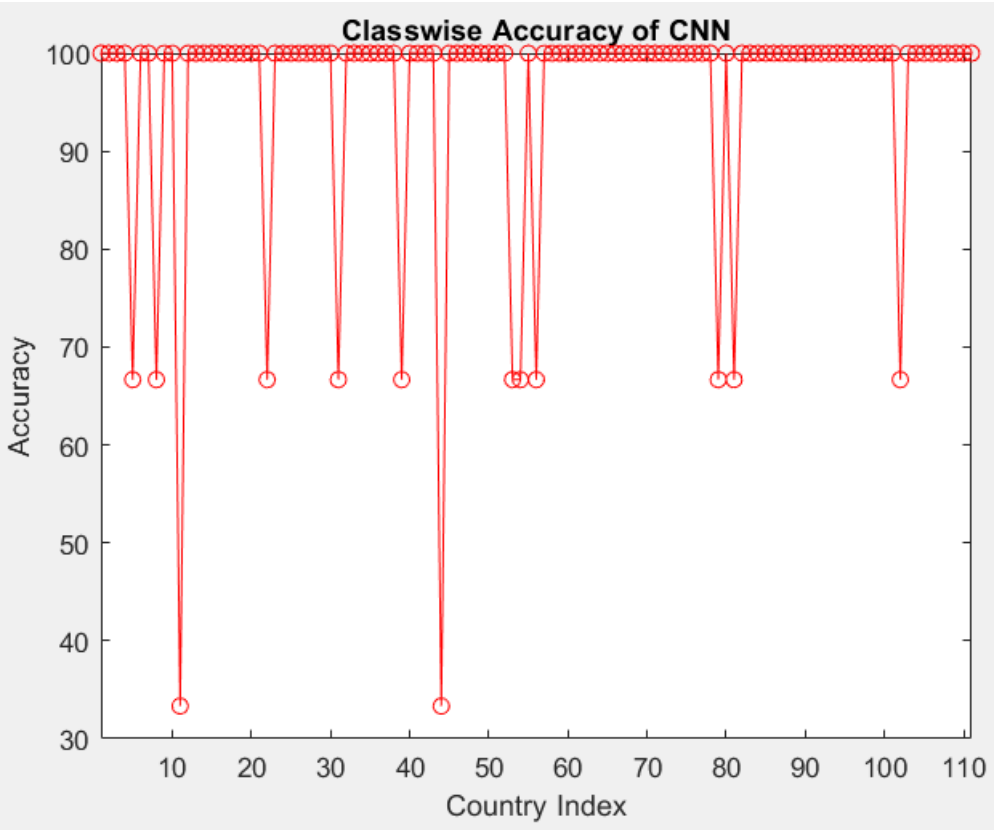


Figure 25: Classification accuracy for each class

In the Figure 25, it is shown each class's accuracy using Convolutional neural networks (CNNs).

4.4: Comparative analysis

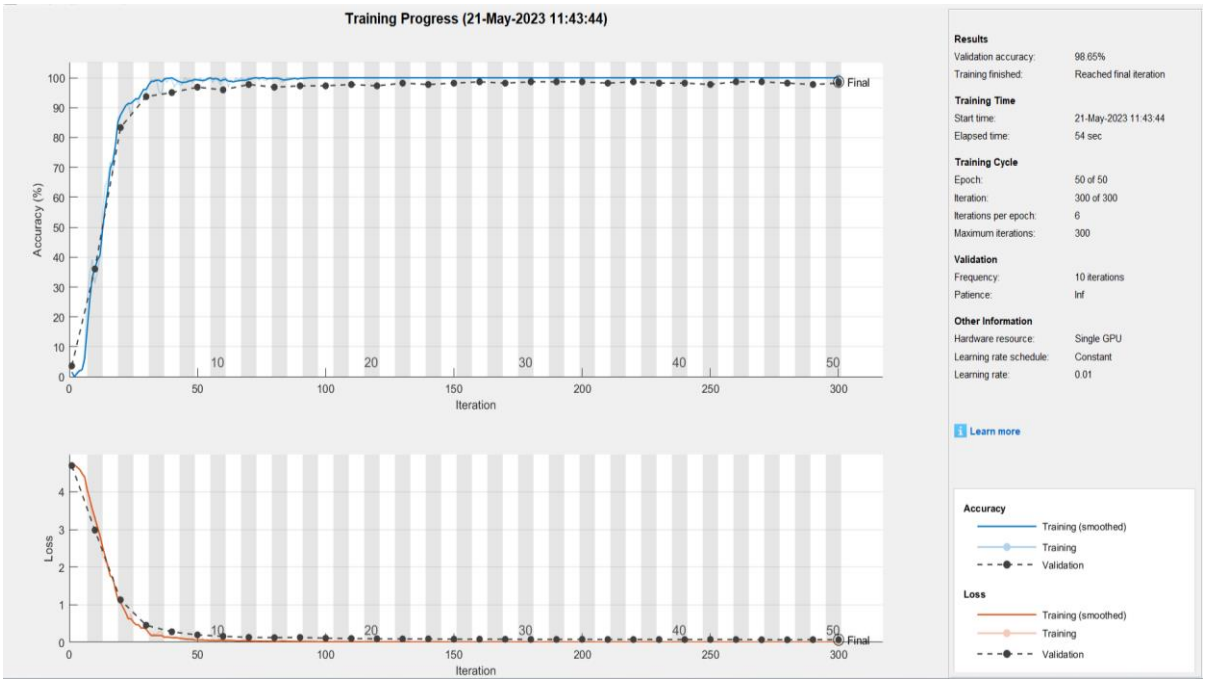


Figure 26: 1st run of training with an additional block (convolution layer, batch normalization layer, ReLU layer) and MaxEpochs-50

In the Figure 26, we are trained our proposed model with an additional block (convolution layer, batch normalization layer, ReLU layer) and MaxEpochs-50 (in the first run).

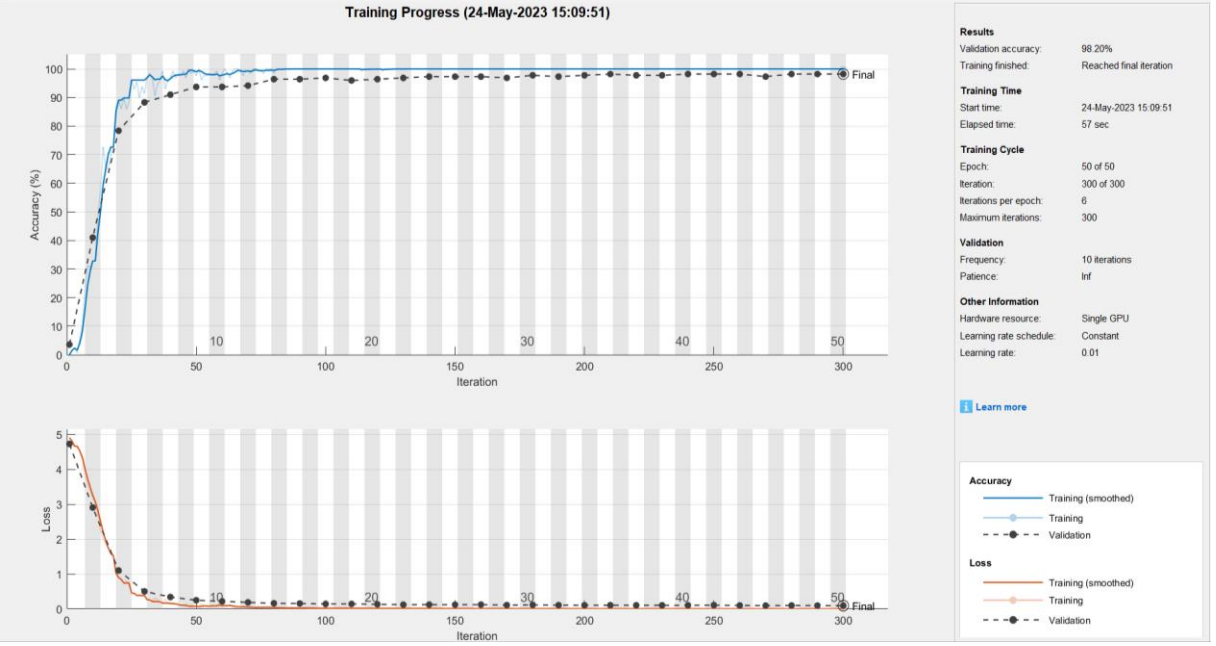


Figure 27: 2nd run of training with additional block (convolution layer, batch normalization layer, ReLU layer) and MaxEpochs-50

In the Figure 27, we are trained our proposed model with an additional block (convolution layer, batch normalization layer, ReLU layer) and MaxEpochs-50 (in the second run).

Table 5: Training Progress with additional block and MaxEpochs

MaxEpochs	Validation Accuracy	Time Taken
50	98.65%	54 sec
50	98.20%	57 sec

We have also tried to train the CNN model by adding convolution, batch normalization, and ReLU layers. We have achieved 98.65% and 98.20% validation accuracy in Run1 and Run2, respectively. In the Table 5, it is shown these values.

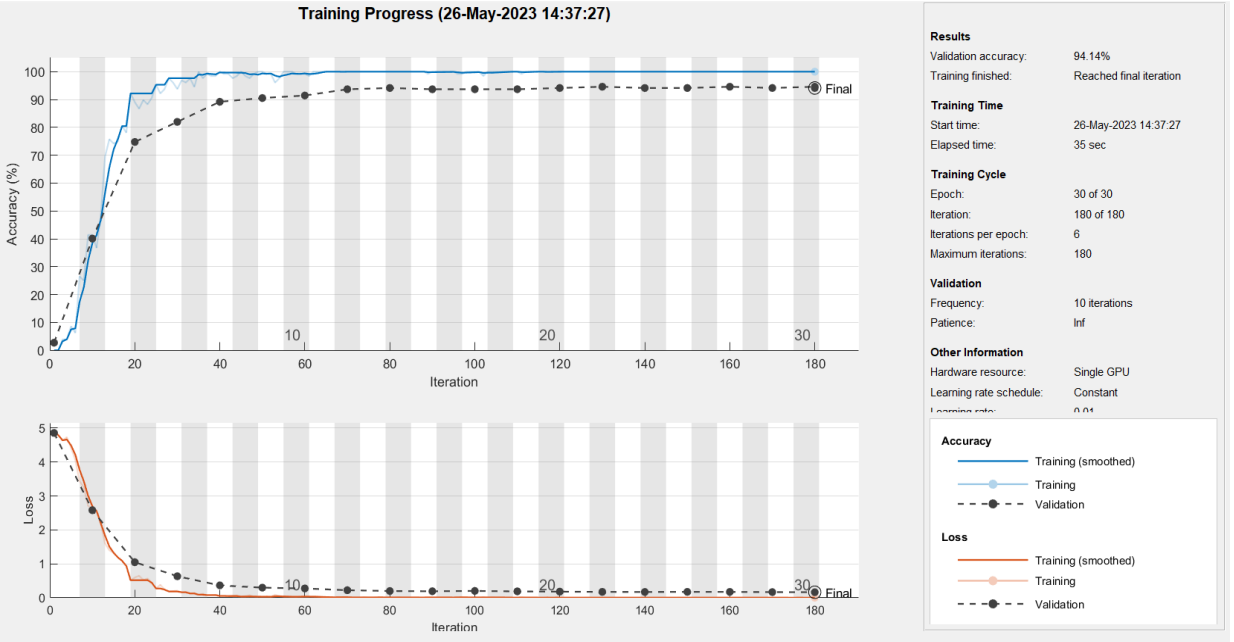


Figure 28: 3rd run of training process (MaxEpochs-30)

In the Figure 28, we are trained our proposed model with MaxEpochs-30 (in the third run).

Table 6: Training Progress of our proposed model and MaxEpochs

MaxEpochs	Validation Accuracy	Time Taken
30	94.14%	35 sec

In the Table 6, after trained our proposed model with MaxEpochs-30 (in the third run) and get the accuracy 94.14% and time taken 35 sec.

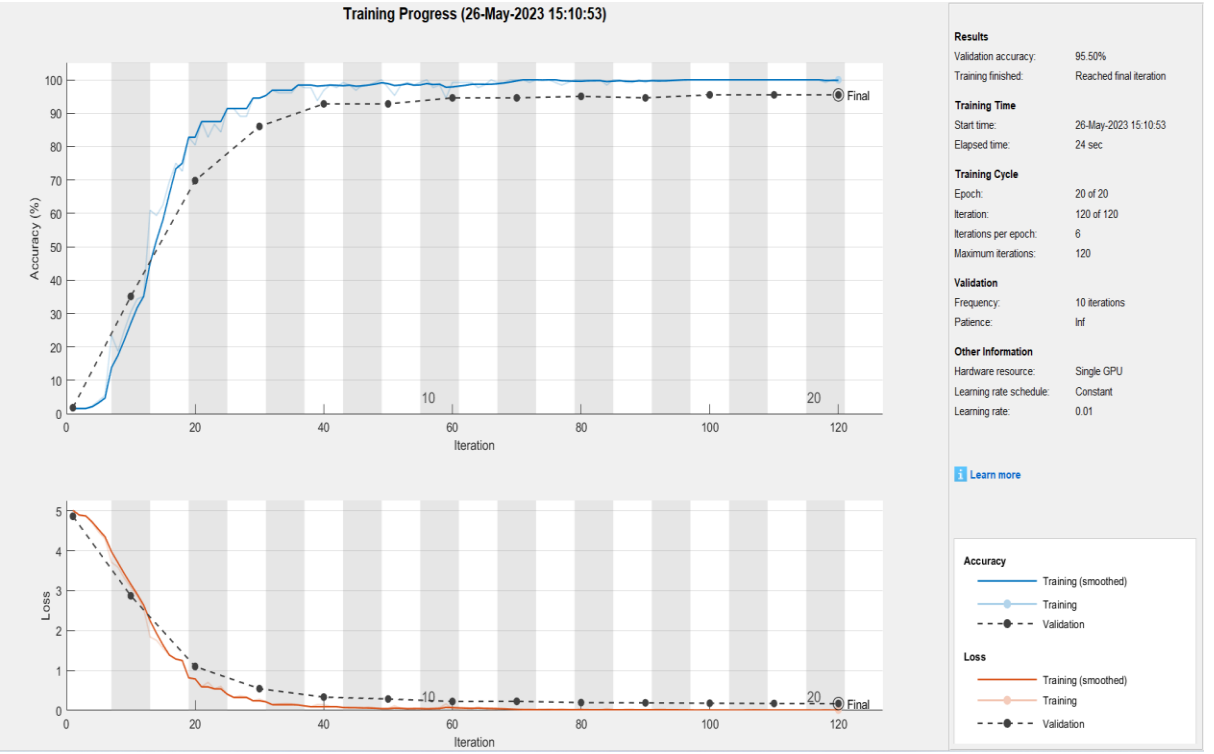


Figure 29: 4th run of training (MaxEpochs-20)

In the Figure 29, we are trained our proposed model with MaxEpochs-20 (in the fourth run).

Table 7: Training Progress of our proposed model and MaxEpochs

MaxEpochs	Validation Accuracy	Time Taken
20	95.50%	24 sec

In the Table 7, it is shown that after trained our proposed model with MaxEpochs-20 and get the accuracy 95.50% and take 24 sec time.

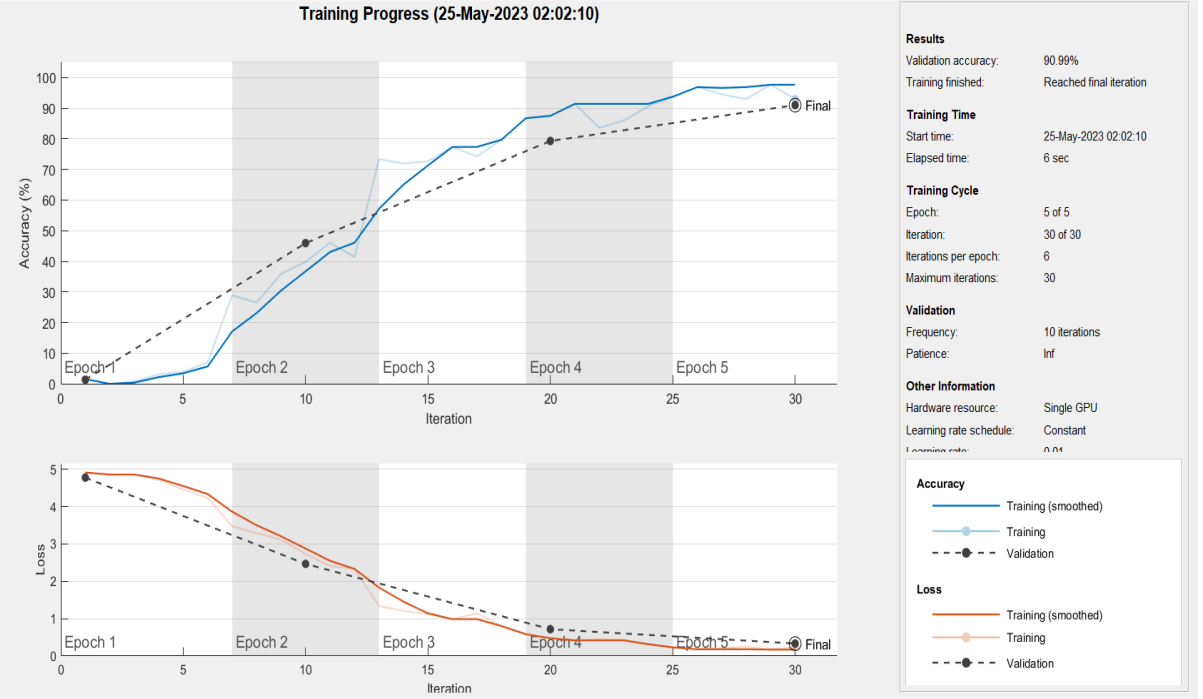


Figure 30: 6th run of training process (MaxEpochs-5)

In the Figure 30, we are trained our proposed model with MaxEpochs-5 (in the sixth run).

Table 8: Training Progress of our proposed model and MaxEpochs

MaxEpochs	Validation Accuracy	Time Taken
5	90.99%	6 sec

In the Table 8, it is shown that after trained our proposed model with MaxEpochs-5, we get the accuracy 95.50% and take 24 sec time.

We can conclude from the comparative analysis of different parameters that increasing the number of MaxEpochs gives better results in overall accuracy but slightly increases the training time.

Table 9: Comparison between our proposed model and model with additional block

Model	Validation Accuracy	Time Taken
Our proposed model	98.20%	54 sec
Our proposed model with an additional block	98.65%	58 sec

In the Table 9, after trained our proposed model with 198 layers and with an additional block get validation accuracy 98.20% and 98.65% respectively. The times taken 54 sec and 58 sec respectively.

We also observed that adding additional blocks does not significantly change in performance and timing. But, it may be increased the complexity of the model.

4.5 Comparative analysis between different machine learning algorithms

In GLCM, image texture is characterized by calculating the frequency of pairs of pixels with specific values. A specified spatial relationship has occurred in an image, and then statistical measures are extracted from this matrix.

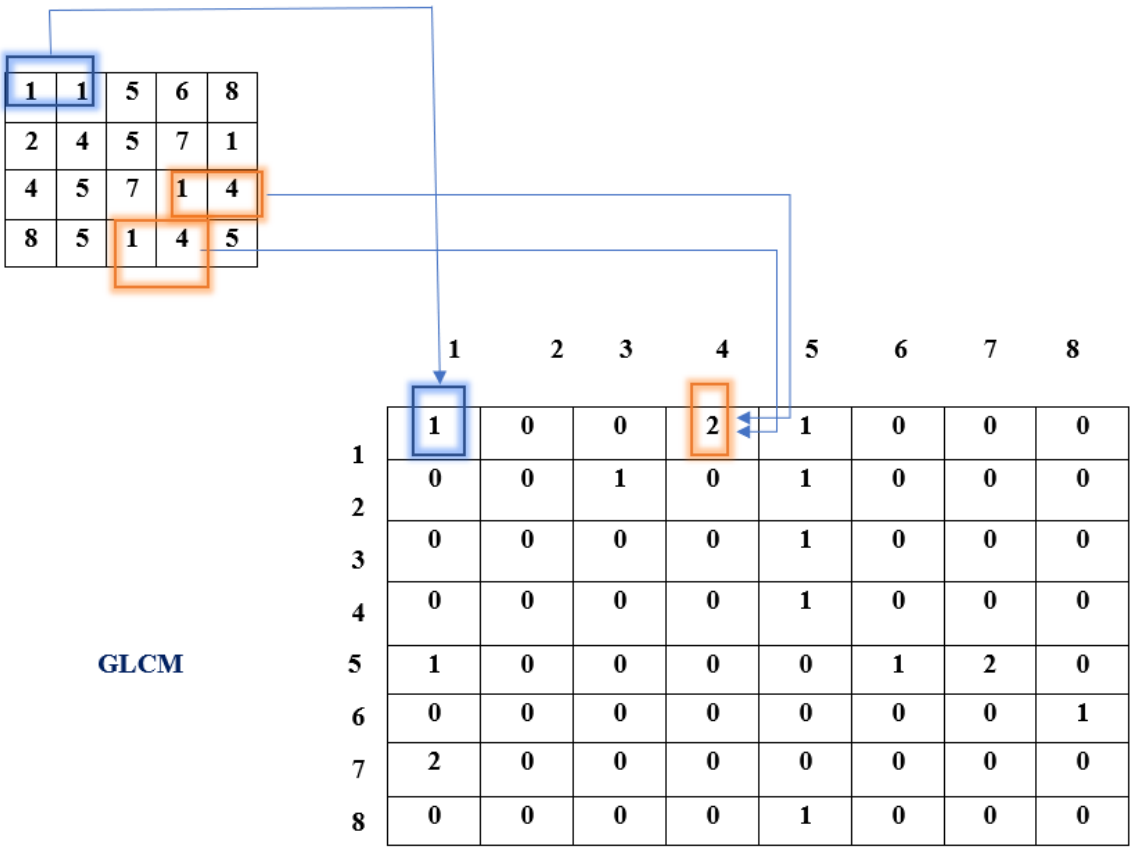


Figure 31: Approaches for creating GLCM

In the Figure 31, each element (i, j) is the resultant GLCM representing the pixel frequency counts with the value i, which occurred in the specified spatial relationship to a pixel with value j in the input image.

4.5.1 Support vector machine (SVM) classifier with GLCM-based features:

With the help of GLCM-based features, we have used SVM classifier to classify different countries' flags. We get 19.22% accuracy with this approach. Support vector machine (SVM) is one type of deep learning algorithm. It performed supervised learning. This supervised learning system, SVM used for flag classification.

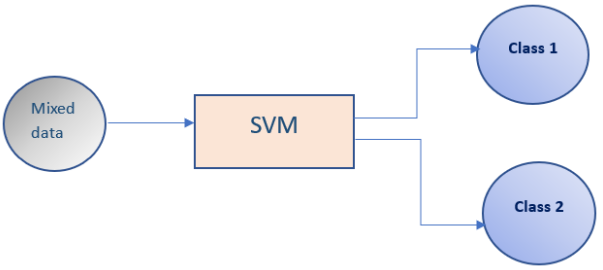


Figure 32: SVM Classifier

The Figure 32 shows the schematic diagram of Support vector machine (SVM) classifier.

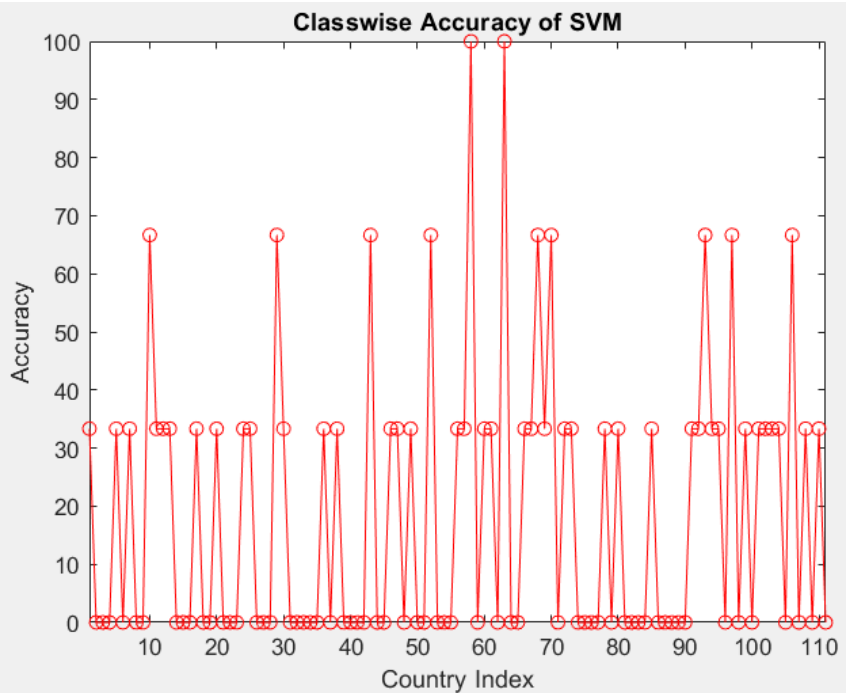


Figure 33: Classwise accuracy of SVM

In Figure 33, it is shown accuracy of each class with the use of GLCM features and SVM classifier.

4.5.2 K-Nearest Neighbours (KNN) classifier with GLCM-based Features:

With the help of GLCM-based features, we have used the KNN classifier to classify different countries' flags. We get 54.05% accuracy with this approach. K-Nearest Neighbours is one of the classification algorithms in Machine Learning. It belongs to the supervised learning domain. It's used in many different areas, such as handwriting detection, image recognition, and video recognition.

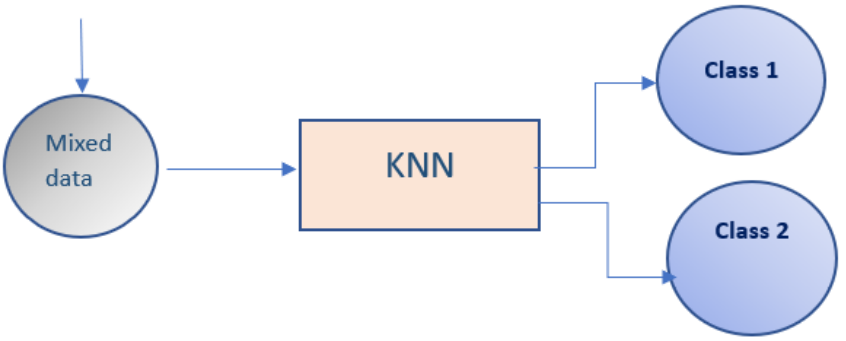


Figure 34: KNN Classifier

The Figure 34 shows the schematic diagram of K-Nearest Neighbours (KNN) classifier.

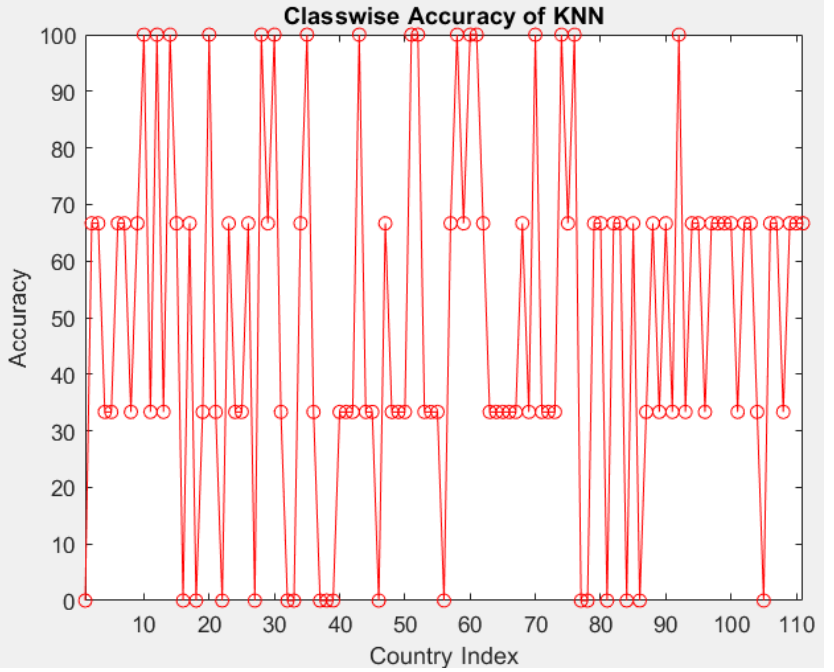


Figure 35: Classwise accuracy of KNN

In the Figure 35 show the accuracy of each class using the GLCM features and KNN classifier. We get 48.05% accuracy with this approach.

4.5.3 Decision Tree classifier with GLCM-based Features:

We are trying another approach, working with a Decision Tree classifier using GLCM features. Decision Tree classification algorithm is a supervised learning algorithm. It is a tree-structured classifier where internal nodes, branches, and each leaf node represent the features of a dataset, decision rules, and the outcome, respectively. It is a graphical representation of all possible solutions to a problem or decision based on given conditions.

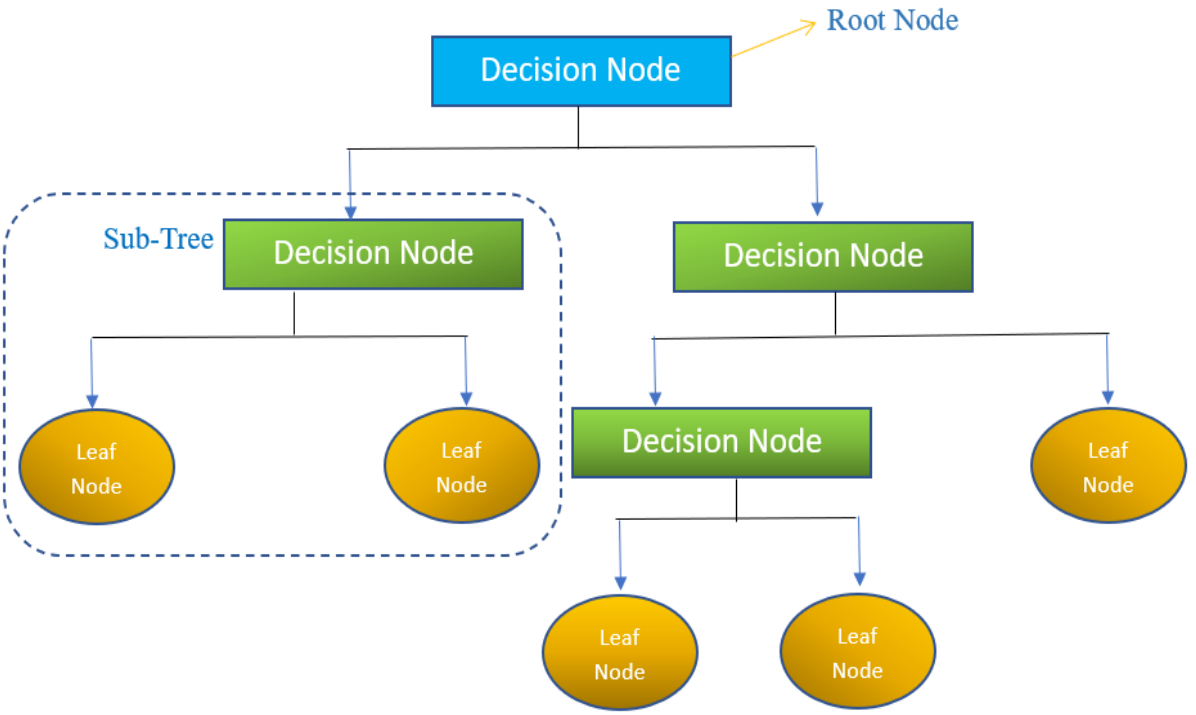


Figure 36: Diagram of Decision Tree

In the Figure 36 shows the graphical representation of Decision Tree.

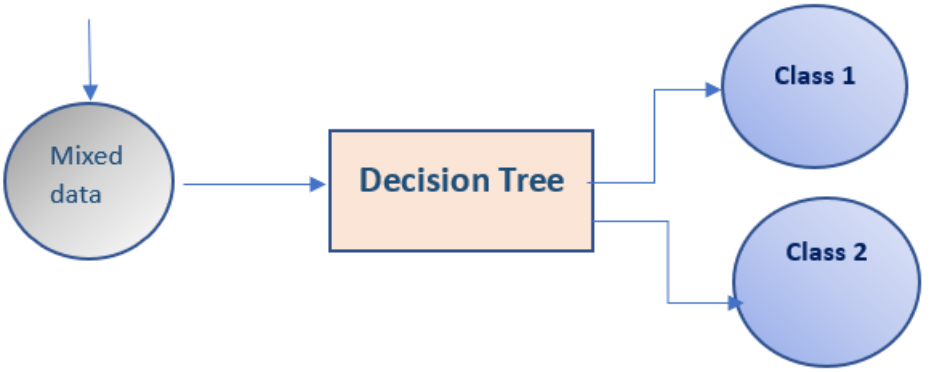


Figure 37: Decision Tree Classifier

The Figure 37 shows the schematic diagram of Decision Tree Classifier.

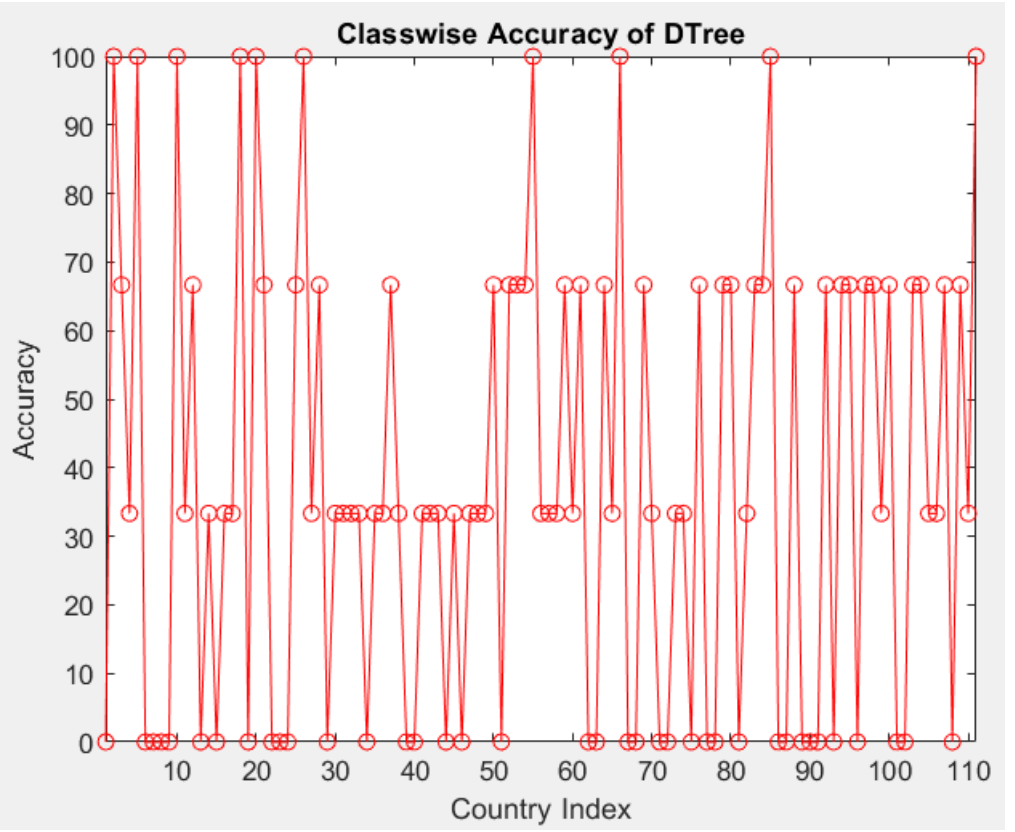


Figure 38: Classwise accuracy of Decision Tree

In the Figure 38 shows the accuracy of each class using the GLCM features and Decision Tree classifier. We get 34.83% accuracy.

4.5.4 Linear Discriminant classifier with GLCM-based Features:

Next, we are trying another approach, working with a Linear Discriminant classifier using GLCM features. Linear Discriminant analysis is used as a dimensionality reduction technique in machine learning. Using this classifier, we can transform a 2-D or 3-D graph into a 1-dimensional plane. Hence, we can maximize the separation between these classes and reduce the 2-D plane into 1-D.

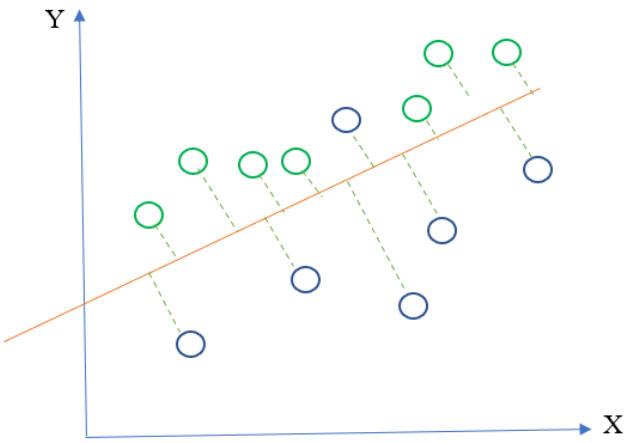


Figure 39: Create a new axis in Linear Discriminant Analysis

In the Figure 39, it is shown that a new axis in Linear Discriminant Analysis.

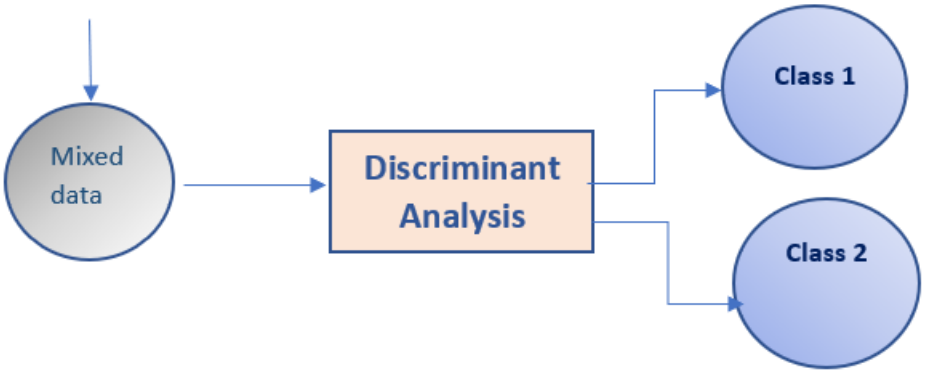


Figure 40: Linear Discriminant Analysis Classifier

The Figure 40 shows the schematic diagram of Decision Tree Classifier.

By creating a new axis, Linear Discriminant Analysis maximizes the distance between the means of two classes and minimizes the variance within the individual class.

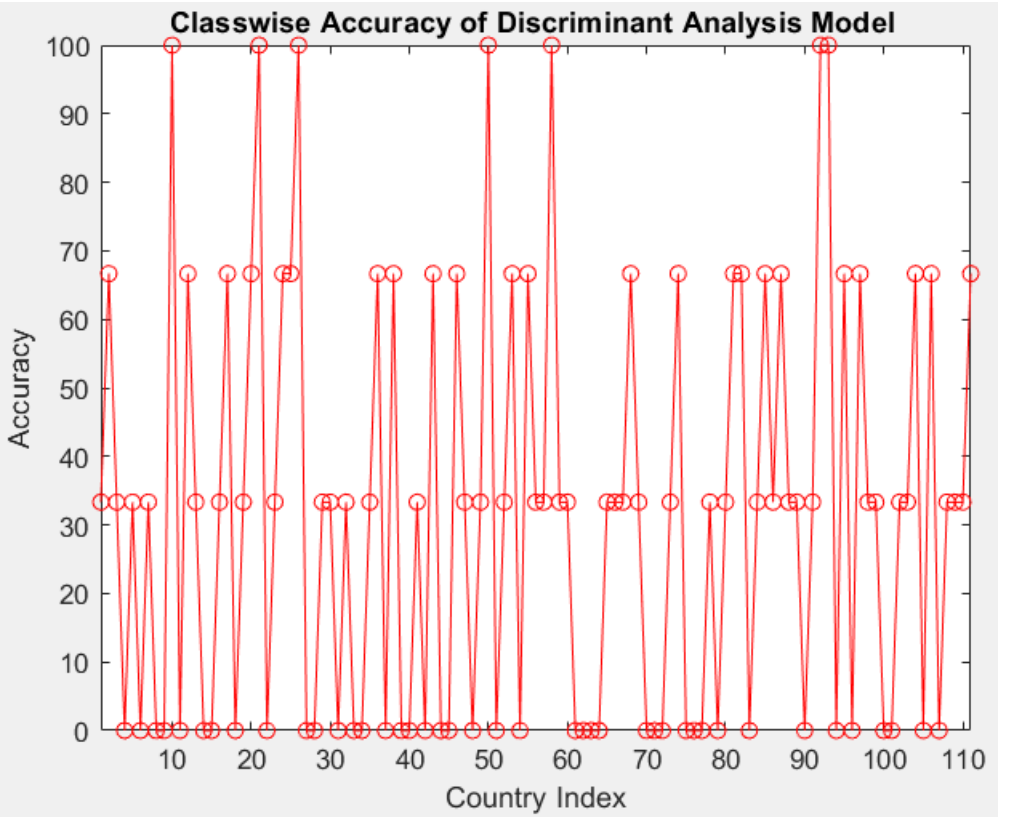


Figure 41: Classwise accuracy of Discriminant Analysis Model

In the Figure 41 shows each class's accuracy using GLCM features and Discriminant Analysis Model classifier. We get 28.53% overall accuracy.

4.5.4 Naïve Bayes classifier with GLCM-based Features:

Next, we are trying another approach, working with a Naïve Bayes classifier using GLCM features. The Naïve Bayes algorithm is a supervised learning algorithm based on **Bayes theorem** and used for solving classification problems. It is a probabilistic classifier that predicts based on an object's probability**.**

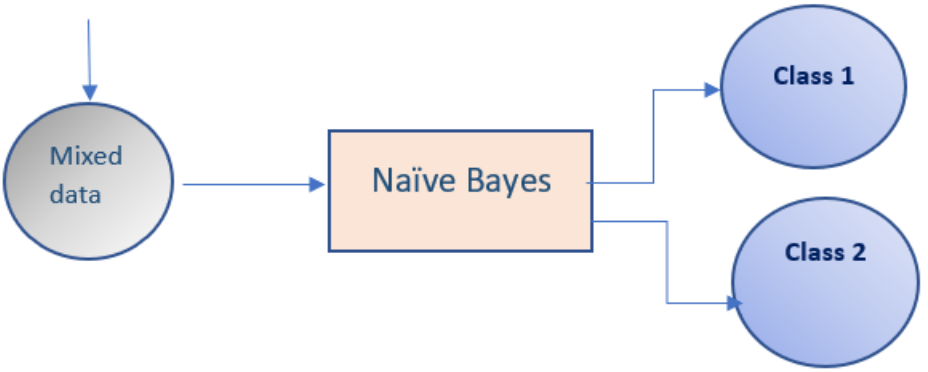


Figure 42: Naïve Bayes Classifier

The Figure 37 shows the schematic diagram of Naïve Bayes Classifier

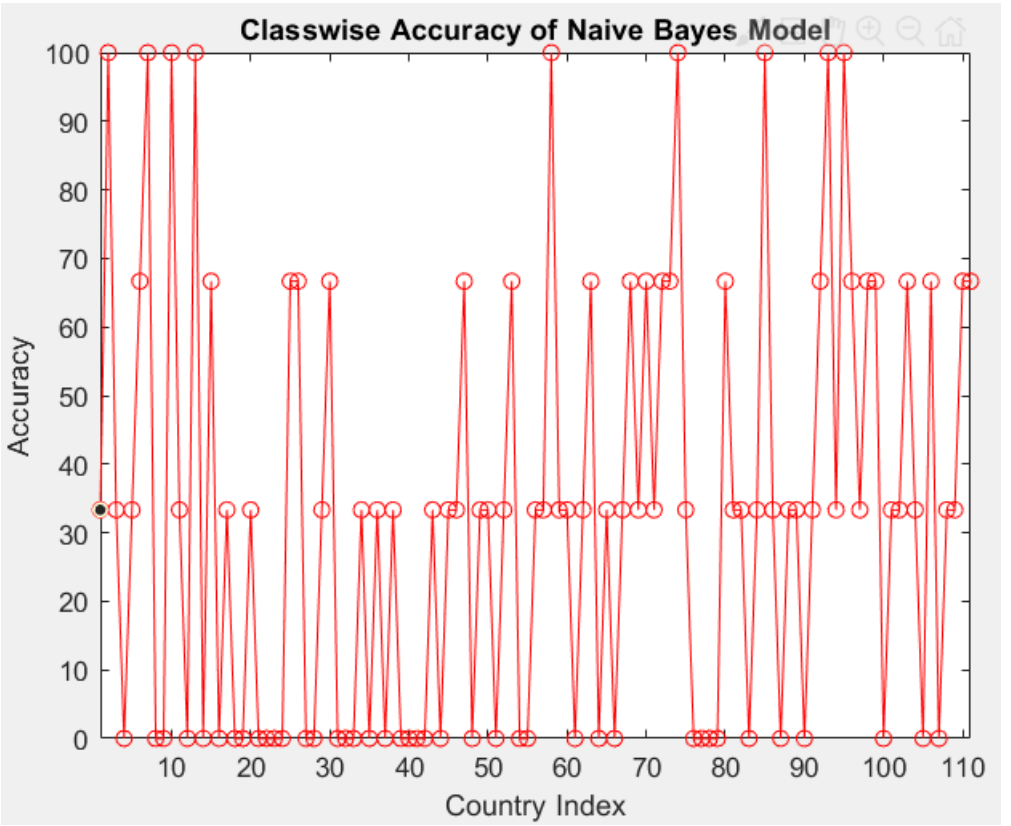


Figure 43: Classwise accuracy of Naive Bayes Model

In the Figure 43 shows each class's accuracy using GLCM features and Naive Bayes Model classifier. We get 32.43% overall accuracy.

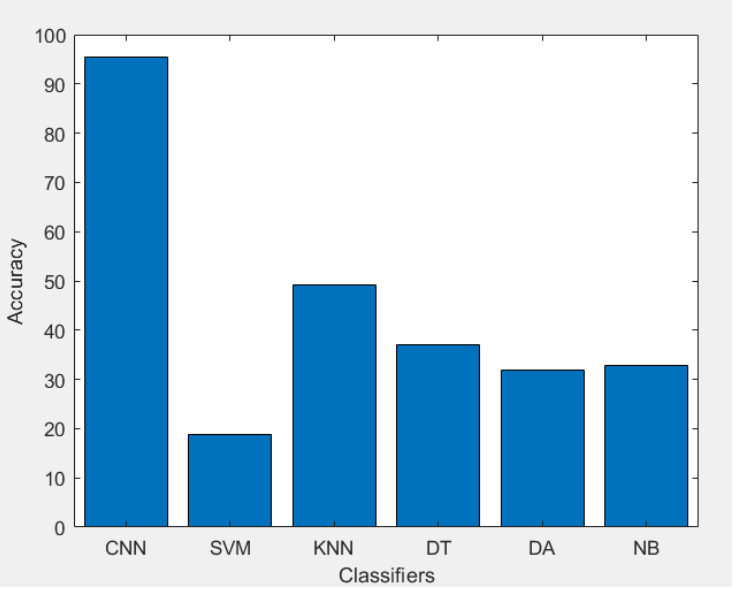


Figure 44: Comparison between different classifiers

In the Figure 44, it is shown the Comparison between different classifiers. The different classifiers are Convolutional Neural Network (CNN), Support Vector Machine (SVM), K-Nearest Neighbor (KNN), Decision Tree (DT), Discriminant Analysis (DA), Naïve Bayes (NB).

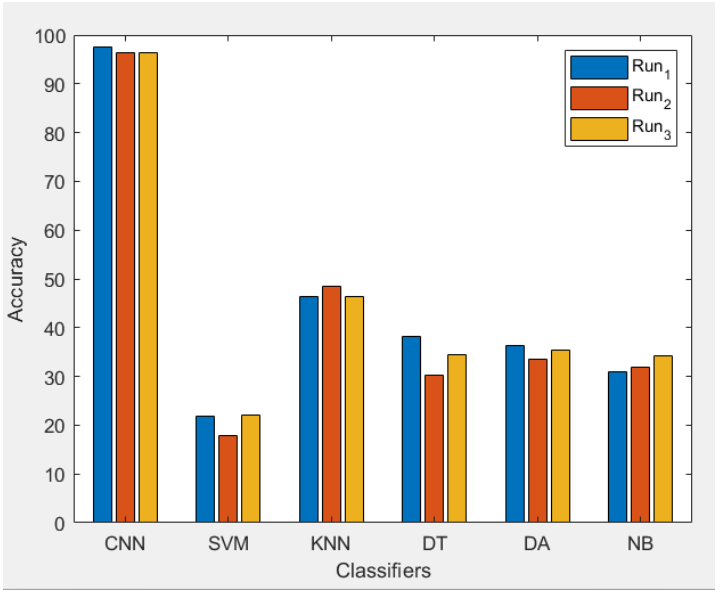


Figure 45: Comparison between different classifiers in 3 Runs

In the Figure 45, it is shown the Comparison between different classifiers in three runs. The different classifiers are Convolutional Neural Network (CNN), Support

Vector Machine (SVM), K-Nearest Neighbor (KNN), Decision Tree (DT), Discriminant Analysis (DA), Naïve Bayes (NB).

Table 10: Overall accuracy of different classifiers

Classifier Names	% accuracy
CNN	97.30
SVM	22.82
KNN	50.45
Decision Tree	30.63
Discriminant Analysis	35.74
Naïve Bayes	34.83

In the above Table 11, it is shown the accuracies of different classifier. The different classifiers are Convolutional Neural Network (CNN), Support Vector Machine (SVM), K-Nearest Neighbor (KNN), Decision Tree (DT), Discriminant Analysis (DA), Naïve Bayes (NB).

Therefore, by analysing the data of class wise accuracy for each classifier and the overall accuracy of each classifier from the above Table 10 and Table 11, it has been proved that the proposed flag detection algorithm is the most efficient and gives more accurate results than the other machine learning based detection algorithm.

Chapter 5

5.0 Conclusion

We have proposed a Convolutional Neural Network (CNN) architecture for identifying different country flag images. In the proposed model, the input layer receives 100×100 RGB color images. The convolution layer is added next to the input layer. This layer consists of eight numbers of 3×3 convolution filters. The batch normalization layer has been added with eight channels in the first block, and the ReLU layer follows it. Finally, a max pooling layer has been added after the ReLU layer in the first block. Similar types of three blocks are added sequentially. The only difference is that the number of filters in the convolution layer, and the number of channels in batch normalization layers have been doubled in each consecutive block. The final block consists of a fully connected layer, a softmax layer, and a classification layer. The overall accuracy of the proposed model is 97.30%. The proposed CNN model for flag detection gives more accurate results than the other prior approaches for flag detection.

5.1 Future Scope

In future, the researchers may work on the following challenges of flag classification-

(a) real-time 2D, 3D, and wavy images of flags

(b) different feature extraction methods and machine learning models can also be tested in this context.

REFERENCES

[1]. Ming Gu, Kun Hao, Zhiyi Qu, “Flag Detection with Convolutional Network ”, CSAI '18, Association for Computing Machinery. ACM ISBN 978-1-4503- 6606, December 8–10, 2018.

[2]. Yahia Said & Mohammad Barr, “Countries flags detection based on local context network and color features”, Multimedia Tools and Applications, 80:14753–14765, 2021.

[3]. Avishikta Lodh, Ranjan Parekh, “Computer Aided Identification of Flags using Color Features”**,** International Journal of Computer Applications (0975 – 8887),Volume 149, No.11, September 2016.

[4]. Shou-Fang Wu, Ming-Ching Chang, Siwei Lyu, Cheng-Shih Wong, Abhineet Kumar Pandey, Po-Chi Su,“FlagDetSeg: Multi-Nation Flag Detection and Segmentation in the Wild”, 2021 17th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS),2021.

[5]. Soe Moe Myint, Moe Moe Myint, Aye Aye Cho, “National Flags Recognition Based on Principal Component Analysis”, International Journal of Trend in Scientific Research and Development (IJTSRD), Volume 3, Issue 5, August 2019.

[6]. Eduardo Hart, Sung-Hyuk Cha, Charles Tappert, “Interactive Flag Identification using Image Retrieval Techniques”, Proceedings of the International Conference on Imaging Science, Systems and Technology, CISST '04, June 21-24, 2004.

[7]. Ivan Zatezalo, Ivan Dunđer, “Development of a mobile application for flag identification based on artificial neural networks”, Vol. 10, No.1, pp. 393-409, 2022.

[8]. Riadh Ayachi, Mouna Afif, Yahia Said, Mohamed Atri, "Traffic Signs Detection for Real-World Application of an Advanced Driving Assisting System Using Deep Learning", 2019.

[9]. Mouna Afif, Riadh Ayachi, Yahia Said "An Evaluation of RetinaNet on Indoor Object Detection for Blind and Visually Impaired Persons Assistance Navigation", NeuralProcessingLetters, 51(10), June 2020.

[10] Susovan Jana , Ranjan Parekh, and Bijan Sarkar, "Detection of Rotten Fruits and Vegetables Using Deep Learning", Algorithms for Intelligent Systems, ISBN 978-981-33-6423-3,ISBN 978-981-33-6424-0 (eBook), 2021

[11] Country flags of the world (URL : www.flagpedia.net)

APPENDIX

```
clc;
clear all;
%% Accessing the Data
DatasetPath = fullfile('E:\Karobi\Assignment-Project\thesis(flag)\dataset1\');

imds = imageDatastore(DatasetPath, ...
'IncludeSubfolders',true,'LabelSource','foldernames');

nc=110;

% Display Sample Images
figure,
perm = randperm(1430,20);
for i = 1:20
subplot(4,5,i);
imshow(imds.Files{perm(i)});
end

% Count Label for Each Class
labelCount = countEachLabel(imds)

% Size of an Image
img = readimage(imds,1);
size(img)

% Splitting into Training & Validation Set for CNN
numTrainFiles = 8;
numValidationFiles=2;
[imdsTrain,imdsX] = splitEachLabel(imds,numTrainFiles,'randomize');
[imdsValidation,imdsTest] =
splitEachLabel(imdsX,numValidationFiles,'randomize');

%% Creating and Configuring Network Layers
layers = [
imageInputLayer([100 100 3])

convolution2dLayer(3,8,'Padding','same')
batchNormalizationLayer
reluLayer
```

```
maxPooling2dLayer(2,'Stride',2)

convolution2dLayer(3,16,'Padding','same')
batchNormalizationLayer
reluLayer

maxPooling2dLayer(2,'Stride',2)

convolution2dLayer(3,32,'Padding','same')
batchNormalizationLayer
reluLayer

maxPooling2dLayer(2,'Stride',2)

convolution2dLayer(3,64,'Padding','same')
batchNormalizationLayer
reluLayer


fullyConnectedLayer(nc)
softmaxLayer
classificationLayer];

%% Train the Network
options = trainingOptions('sgdm', ...
'InitialLearnRate',0.01, ...
'MaxEpochs',50, ...
'Shuffle','every-epoch', ...
'ValidationData',imdsValidation, ...
'ValidationFrequency',10, ...
'Verbose',false, ...
'Plots','training-progress');

net = trainNetwork(imdsTrain,layers,options);


%% Checking Network Accuracy
YPred_CNN = classify(net,imdsTest);
YTest_CNN = imdsTest.Labels;

accuracy_CNN = round(sum(YPred_CNN ==
YTest_CNN)*100/numel(YTest_CNN),2);
[cf_CNN cl] = confusionmat(YTest_CNN,YPred_CNN);
```

```
classwiseaccuracy_CNN = cell(nc,2);
cl=cellstr(cl);
classwiseaccuracy_CNN(:,1) = cl;

for i = 1:nc
for j = 1:nc
if i == j
classwiseaccuracy_CNN{i,2} = (cf_CNN(i,j)*100)/3;
fprintf("Accuracy of %s is %d \n ", classwiseaccuracy_CNN{i,1},
classwiseaccuracy_CNN{i,2})
end
end
end
%% support vector machine (SVM)
numTestFiles = 3;

%% Training support vector machine (SVM)Feature extraction
trainingLabels = imdsTrain.Labels;
trainingFeatures = zeros(numTrainFiles*nc,16);
for i=1:size(imdsTrain.Files)
file_name=imdsTrain.Files{i};
I=imread(file_name);
Inew=imresize(I,[100 100]);
%     imwrite(Inew,file_name)
%     [a fs]=size(file_name);


gray_image = rgb2gray(I);
glcm_img = graycomatrix(gray_image);
pixel_dist = 2;
GLCM = graycomatrix(gray_image,'Offset',[0 pixel_dist;...
-pixel_dist pixel_dist; -pixel_dist 0; -pixel_dist pixel_dist]);
stats = graycoprops(GLCM,{'Contrast','Correlation','Energy','Homogeneity'});

trainingFeatures(i,:)=[stats.Contrast,stats.Correlation,stats.Energy,stats.Homoge
neity];
end

%% Testing support vector machine (SVM)Feature extraction
testingLabels = imdsTest.Labels;
testingFeatures = zeros(numTestFiles*nc,16);
for i=1:size(imdsTest.Files)
file_name=imdsTest.Files{i};
```

```
I=imread(file_name);
Inew=imresize(I,[100 100]);
gray_image = rgb2gray(I);
glcm_img = graycomatrix(gray_image);
pixel_dist = 2;
GLCM = graycomatrix(gray_image,'Offset',[0 pixel_dist;...
-pixel_dist pixel_dist; -pixel_dist 0; -pixel_dist pixel_dist]);
stats = graycoprops(GLCM,{'Contrast','Correlation','Energy','Homogeneity'});

testingFeatures(i,:)=[stats.Contrast,stats.Correlation,stats.Energy,stats.Homogen
eity];
end

%% Train support vector machine (SVM) Model
svmModel = fitcecoc(trainingFeatures,trainingLabels);

%% Test support vector machine (SVM) Model
predictedLabels_SVM = predict(svmModel,testingFeatures);

YPred_SVM= predictedLabels_SVM;
YTest_SVM = testingLabels;

accuracy_SVM = round(sum(YPred_SVM ==
YTest_SVM)*100/numel(YTest_SVM),2);

[cf_SVM cl] = confusionmat(YTest_SVM,YPred_SVM);
classwiseaccuracy_SVM = cell(nc,2);
cl=cellstr(cl);
classwiseaccuracy_SVM(:,1) = cl;

for i = 1:nc
for j = 1:nc
if i == j
classwiseaccuracy_SVM{i,2} = (cf_SVM(i,j)*100)/numTestFiles;
fprintf("Accuracy of %s is %d \n ", classwiseaccuracy_SVM{i,1},
classwiseaccuracy_SVM{i,2})
end
end
end
%% Train KNN Model
knnModel = fitcknn(trainingFeatures,trainingLabels);
%% Test KNN Model
predictedLabels_KNN = predict(knnModel,testingFeatures);
```

```
YPred_KNN= predictedLabels_KNN;
YTest_KNN = testingLabels;

accuracy_KNN = round(sum(YPred_KNN ==
YTest_KNN)*100/numel(YTest_KNN),2);
[cf_KNN cl] = confusionmat(YTest_KNN,YPred_KNN);
classwiseaccuracy_KNN = cell(nc,2);
cl=cellstr(cl);
classwiseaccuracy_KNN(:,1) = cl;
for i = 1:nc
for j = 1:nc
if i == j
classwiseaccuracy_KNN{i,2} = (cf_KNN(i,j)*100)/numTestFiles;
fprintf("Accuracy of %s is %d \n ", classwiseaccuracy_KNN{i,1},
classwiseaccuracy_KNN{i,2})
end
end
end

%% Train Decision Tree Model
ctreeModel = fitctree(trainingFeatures,trainingLabels);

%% Test Decision Tree Model
predictedLabels_DTree = predict(ctreeModel,testingFeatures);

YPred_DTree= predictedLabels_DTree;
YTest_DTree = testingLabels;

accuracy_DTree = round(sum(YPred_DTree ==
YTest_DTree)*100/numel(YTest_DTree),2);
[cf_DTree cl] = confusionmat(YTest_DTree,YPred_DTree);
classwiseaccuracy_DTree = cell(nc,2);
cl=cellstr(cl);
classwiseaccuracy_DTree(:,1) = cl;

for i = 1:nc
for j = 1:nc
if i == j
classwiseaccuracy_DTree{i,2} = (cf_DTree(i,j)*100)/numTestFiles;
fprintf("Accuracy of %s is %d \n ", classwiseaccuracy_DTree{i,1},
classwiseaccuracy_DTree{i,2})
end
```

```
end
end

%% Train a Discriminant Analysis Model
discriminantModel = fitcdiscr(trainingFeatures,trainingLabels);

%% Test a Discriminant Analysis Model
predictedLabels_discriminant = predict(discriminantModel,testingFeatures);

YPred_discriminant= predictedLabels_discriminant;
YTest_discriminant = testingLabels;

accuracy_discriminant = round(sum(YPred_discriminant ==
YTest_discriminant)*100/numel(YTest_discriminant),2);
[cf_discriminant cl] = confusionmat(YTest_discriminant,YPred_discriminant);
classwiseaccuracy_discriminant = cell(nc,2);
cl=cellstr(cl);
classwiseaccuracy_discriminant(:,1) = cl;

for i = 1:nc
for j = 1:nc
if i == j
classwiseaccuracy_discriminant{i,2} =
(cf_discriminant(i,j)*100)/numTestFiles;
fprintf("Accuracy of %s is %d \n ", classwiseaccuracy_discriminant{i,1},
classwiseaccuracy_discriminant{i,2})
end
end
end

%% Train a Naive Bayes Model
Naive_BayesModel = fitcnb(trainingFeatures,trainingLabels);

%% Test a Naive Bayes Model
predictedLabels_NaiveBayes = predict(Naive_BayesModel,testingFeatures);

YPred_NaiveBayes= predictedLabels_NaiveBayes;
YTest_NaiveBayes = testingLabels;

accuracy_NaiveBayes = round(sum(YPred_NaiveBayes ==
YTest_NaiveBayes)*100/numel(YTest_NaiveBayes),2);
[cf_NaiveBayes cl] = confusionmat(YTest_NaiveBayes,YPred_NaiveBayes);
classwiseaccuracy_NaiveBayes = cell(nc,2);
```

```
cl=cellstr(cl);
classwiseaccuracy_NaiveBayes(:,1) = cl;

for i = 1:nc
for j = 1:nc
if i == j
classwiseaccuracy_NaiveBayes{i,2} = (cf_NaiveBayes(i,j)*100)/numTestFiles;
fprintf("Accuracy of %s is %d \n ", classwiseaccuracy_NaiveBayes{i,1},
classwiseaccuracy_NaiveBayes{i,2})
end
end
end

%% Accuracy Calculation Graph
figure,
x = categorical({'CNN', 'SVM', 'KNN', 'DT','DA','NB'});
x = reordercats(x,{'CNN', 'SVM', 'KNN', 'DT','DA','NB'});
y =
[accuracy_CNN;accuracy_SVM;accuracy_KNN;accuracy_DTree;accuracy_dis
criminant;accuracy_NaiveBayes];
bar(x, y);
xlabel('Classifiers');
ylabel('Accuracy');
%b = bar(1:6,'FaceColor','flat');
%b.CData(2,:) = [0 0.8 0.8];
%% CNN Classwise Accuracy Graph
x= 1:nc;
class_name=  [classwiseaccuracy_CNN{:,1}]
y = [classwiseaccuracy_CNN{:,2}]
figure;
plot (x,y,'r-o')
xlim([1 nc])
xlabel('Country Index');
ylabel('Accuracy');
title('Classwise Accuracy')
figure;
bar(y);
xlim([1 nc])
xlabel('Country Index');
ylabel('Accuracy');
title('Classwise Accuracy')
```