Report on

Thumbs-Up Vs Peace Sign Binary Classification With a Fully Connected Neural Network Based Binary Classifier and Convolutional Neural Network Based Binary Classifier

Submitted by - Group 6

Joy Karmoker

Student ID: 1810176127

Masum Billah

Student ID: 1910576118

Jannatun Ferdous

Student ID: 1912076147

Shahin Alam

Student ID: 1810176115

Contents

1 Introduction

1.1 Overview of the used model

In our transfer learning approach, we employed two distinct models: VGG16 and ResNet. These models serve as powerful architectures pre-trained on the ImageNet dataset, providing a robust foundation for feature extraction.

2 Dataset Description

2.1 Source of data

There are two types of captured data:

i) Thumbs-up sign: This gesture, formed by raising the thumb, symbolizes a positive affirmation or agreement. Although the thumb is not raised, this gesture is still considered a thumbs-up sign, particularly in this context.

ii) Peace sign: Formed by raising the index and middle fingers together, this gesture signifies the number two or conveys a message of peace.

some examples of source data:

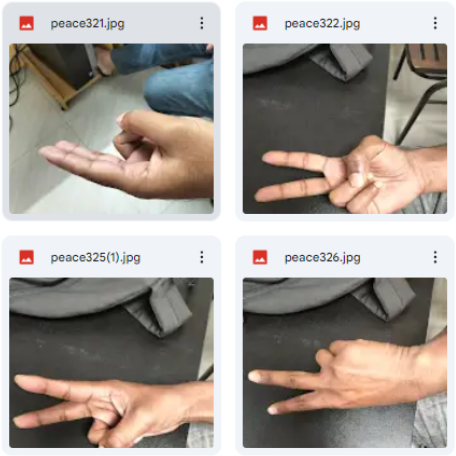

Figure 1: Peace sign test dataset

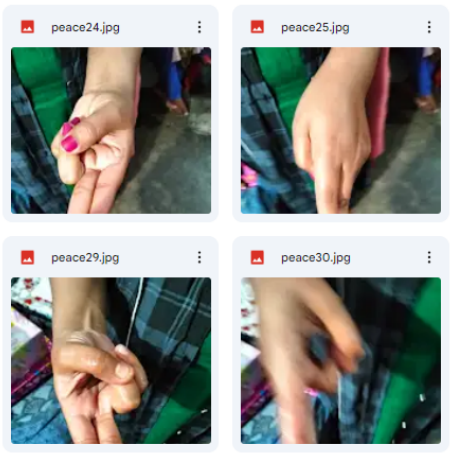

Figure 2: Peace sign train dataset

We have captured 800 (400 images for each classification) RGB images with the help of two conventional mobile cameras and collected the dataset from 100 people by capturing 8 images from each person from different angles. All the images are taken with permission from the individual. All the images are kept in

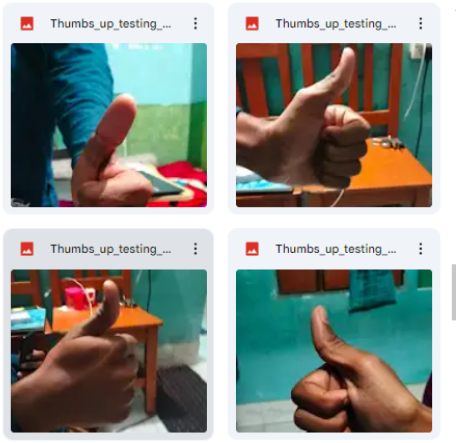

Figure 3: Thumbs-Up sign test dataset

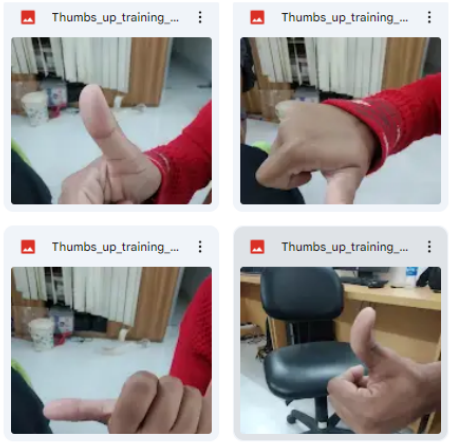

Figure 4: Thumbs-Up sign train dataset

Google Drive after collecting from our group members.
Drive link: Click here to get overall data

2.2 Categorical classification

Our binary classification dataset is divided into four categories:

- Peace sign for train: This section contains images of the peace sign used for training.
- Peace sign for test: Here, you'll find images of the peace sign reserved for testing the model.
- Thumbs-up sign for train: Images of the thumbs-up sign are included here for model training.
- Thumbs-up sign for test:This section holds images of the thumbs-up sign for evaluating the trained model.

2.3 Splitting the data

To manage the training and testing data splits for the "Peace" and "Thumbs-up", we can follow these steps:

Data Preparation: To manage the training and testing data splits for the "Peace" and "Thumbs-up" categories, we can follow these steps. Firstly, in the

data preparation phase, we divide the dataset into training and testing sets. Specifically, 80% of the data is reserved for training purposes, while the remaining 20% is allocated for testing.

Training Set Splitting: Within the training data, 90% is utilized for the actual training process. The remaining 10% of the training data is set aside for validation purposes. This ensures that the model is trained on a significant portion of the data while still having a subset reserved for validating its performance during the training process.

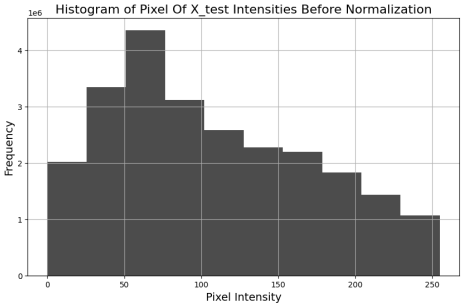

Figure 5:
Pixel Intensities Histogram Of Test and Train Data Before Normalization

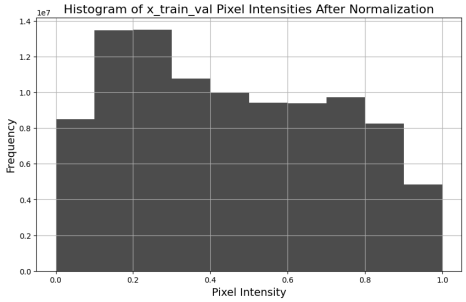

Figure 6:
Pixel Intensities Histogram Of Test and Train Data After Normalization

Testing Set Usage:

- Use the entire testing set (20% of the original data) for evaluating the models trained on the respective training datasets.

2.4 Data Normalization

X_train_val and X_test data are normalized. The data is split into training and validation sets, with 90% used for model training and 10% reserved for validation.

2.4.1 Pixel Intensity Distribution (Before Normalization)

Histograms are plotted for X_train_val and X_test before normalization to visualize pixel intensity distribution. The highest and lowest intensity of pixels are printed.

2.4.2 Pixel Intensity Distribution (After Normalization)

Histograms are plotted for X_train_val and X_test after normalization to visualize pixel intensity distributions. Additionally, the highest and lowest intensity of pixels are printed.

2.5 Class Distribution in Training and Validation Set

A histogram is displayed showing the distribution of classes (Peace and ThumbsUp) in both the training set and validation set. We plot the Train Set, Test Set and Validation Set.

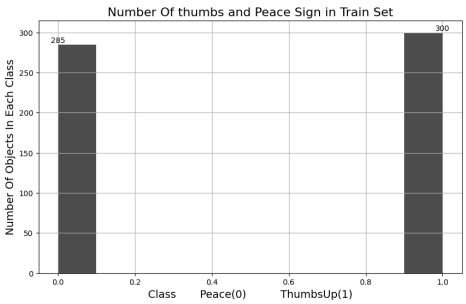

Figure 7:
Number of peace and thumbs-up sign in Train set

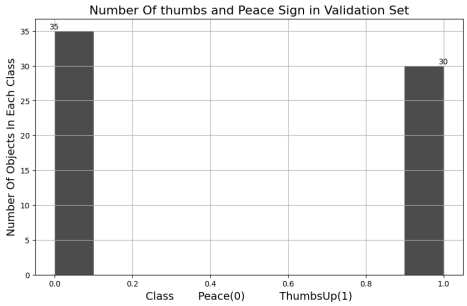

Figure 8:
Number of peace and thumbs-up sign in Validation set

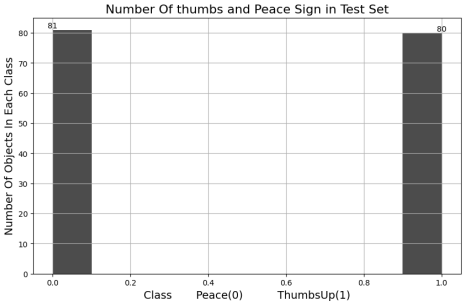

Figure 9:
Number of peace and thumbs-Up sign in Test set

2.6 Labeling the Data

- We have 4 categories of data sets which were defined previously. The first two datasets are for peace sign which is labeled as 0 and the last two datasets are for Thumbs-up which is labeled as 1.
- resized images: all the images are resized into 224x224 pixels.

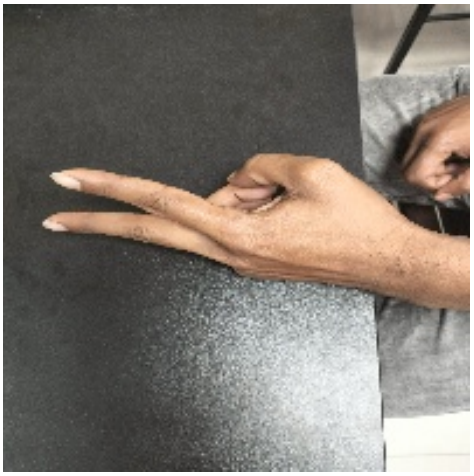

Figure 10: Label: 0

Figure 11: Label: 0

Figure 12: Label: 0

Figure 13: Label: 1

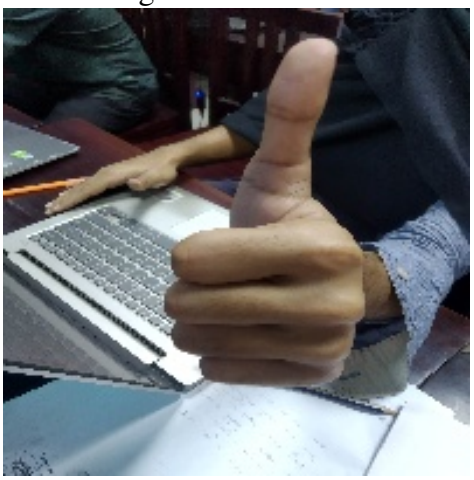

Figure 14: Label: 1

Figure 15: Label: 1

Figure 16:
Images labelled with Peace sign (Label: 0) and Thumbs-up sign (Label: 1)

2.7 Shapes of Different data

The shape of the training data (x) is (585, 224, 224, 3), indicating that there are 585 samples with an image size of 224x224 pixels and 3 colour channels (RGB). Similarly, the validation data (x) has a shape of (65, 224, 224, 3), containing 65 samples with the same image dimensions.

The test data (x) has a shape of (161, 224, 224, 3), consisting of 161 samples with the same image size and colour channels as the training and validation sets.

Regarding the labels (y), the shape of the training data is (585,), indicating there are 585 corresponding labels. The validation data's label shape is (65,), containing 65 labels, while the test data's label shape is (161,), indicating 161 labels corresponding to the test samples.

3 Architecture

In this section, we delve into the architectural details of two distinct models utilized for binary classification in the Thumbs-Up vs. Peace Sign dataset. The models employed are a Convolutional Neural Network (CNN) and a Fully Connected Neural Network (FCNN).

3.1 CNN Architecture

The Convolutional Neural Network (CNN) is designed for image-related tasks, specializing in capturing spatial hierarchies and patterns within images. The CNN architecture comprises convolutional layers, which apply filters to the input data, extracting features and preserving spatial relationships. Max-pooling layers are employed for downsampling, reducing the spatial dimensions of the extracted features. The fully connected layers towards the end of the network aggregate these features for the final classification.

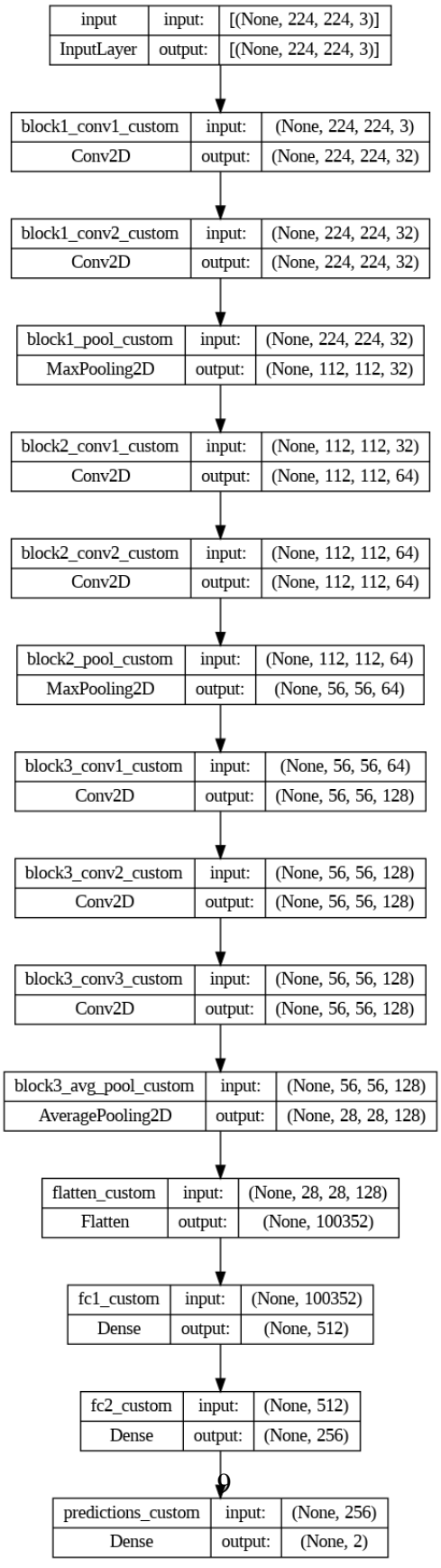

Figure 17: Architecture of Convolutional Neural Network (CNN) for Binary Classification

Table 1 details the layers and parameters of the CNN model.

Table 1: CNN Model Architecture and Parameters

Layer (Type)	**Output Shape**	**Param #**	**Activation**
input	(None, 224, 224, 3)	0	-
block1_conv1	(None, 224, 224, 32)	896	ReLU
block1_conv2	(None, 224, 224, 32)	9248	ReLU
block1_pool	(None, 112, 112, 32)	0	-
block2_conv1	(None, 112, 112, 64)	18,496	ReLU
block2_conv2	(None, 112, 112, 64)	36,928	ReLU
block2_pool	(None, 56, 56, 64)	0	-
block3_conv1	(None, 56, 56, 128)	73,856	ReLU
block3_conv2	(None, 56, 56, 128)	147,584	ReLU
block3_conv3	(None, 56, 56, 128)	147,584	ReLU
block3_avg_pool	(None, 28, 28, 128)	0	-
flatten_custom	(None, 100352)	0	-
fc1_custom	(None, 512)	51,380,736	ReLU
fc2_custom	(None, 256)	131,328	ReLU
predictions_custom	(None, 2)	514	Softmax
Total params: 51,947,170 Trainable params: 51,947,170 Non-trainable params: 0			

3.2 FCNN Architecture

The Fully Connected Neural Network (FCNN) is a classical architecture commonly used for image classification tasks. Unlike the CNN, the FCNN includes only dense layers, where each neuron is connected to every neuron in the subsequent layer. The input layer matches the flattened size of the image data, and the hidden layers gradually decrease in the number of neurons. The final output layer contains two neurons with a sigmoid activation function, suitable for binary classification.

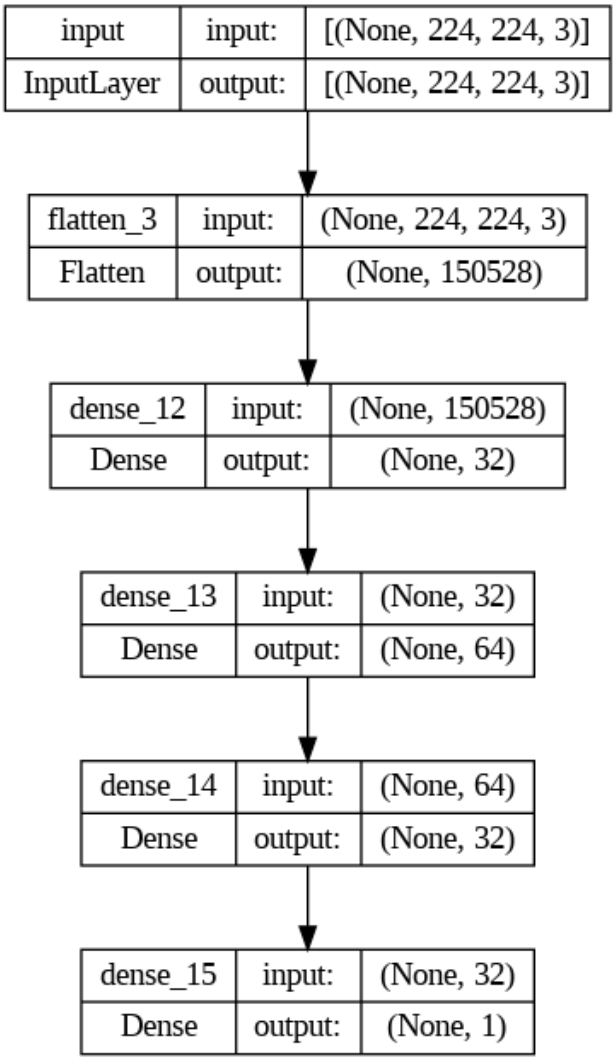

Figure 18: Architecture of Fully Connected Neural Network (FCNN) for Binary Classification

Table 2 details the layers and parameters of the FCNN model.

Table 2: FCNN Model Architecture and Parameters

Layer (Type)	**Output Shape**	**Param #**	**Activation**
input	(None, 224, 224, 3)	0	-
flatten	(None, 150528)	0	-
dense	(None, 32)	4,816,928	ReLU
dense_1	(None, 64)	2,112	ReLU
dense_2	(None, 32)	2,080	ReLU
dense_3	(None, 1)	33	Sigmoid
Total params: 4,821,153 Trainable params: 4,821,153 Non-trainable params: 0			

4 Results

4.1 Overview of the used model

In our project, we apply two distinct models to classify thumbs up and peace sign using deep learning binary classification. These two architecture models are CNN-based with 2 convolution layers and Fully Connected Neural Network model.

4.2 Learning curve for train loss and accuracy

CNN model: The learning curve for the model model shows how the model learns over time.

In the first plot, we see how the training loss (in purple) increases as the model learns from the data. The validation loss (in orange) also decreases.

The second plot illustrates the training accuracy (in purple) and validation accuracy (in orange) across epochs. We aim for both accuracies to increase, but if the gap between them widens, it might indicate overfitting, meaning our model is getting too good at the training data but not generalizing well to new data.

FCNN model: The learning curve for the model model shows how the model learns over time.

The first plot displays the training loss (in purple) and validation loss (in orange) over different epochs.The validation loss decreases.Train loss also decreases but may start to level off or increase later, indicating potential overfitting.

In the second plot, we observe the training accuracy (in purple) and validation accuracy (in orange) increasing over epochs.

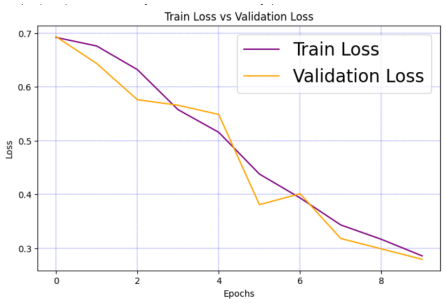

Figure 19: Train Loss vs Validation Loss for CNN Classifier

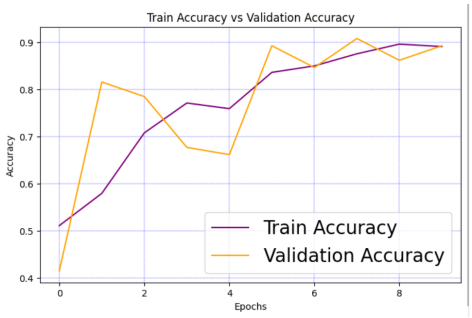

Figure 20: Train Accuracy vs Validation Accuracy for CNN Classifier

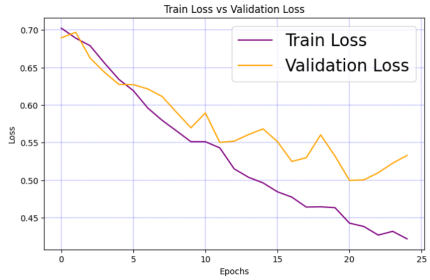

Figure 21: Train Loss vs Validation Loss for FCNN Classifier

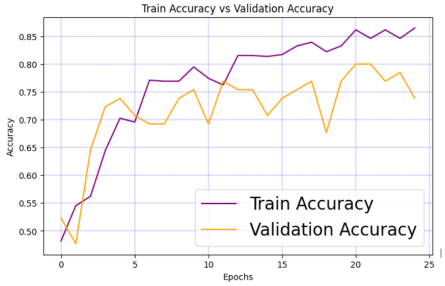

Figure 22: Train Accuracy vs Validation Accuracy for FCNN Classifier

4.3 Prediction of Test Data

The confusion matrix visualizes the performance of the CNN model. Each cell in the matrix represents the count of true positive, true negative, false positive, and false negative predictions made by the model. The diagonal elements indicate correct predictions, while off-diagonal elements represent incorrect predictions. The plot includes customized font sizes for better readability of labels and values. The x-axis and y-axis represent predicted and true labels, respectively. This visualization helps to assess the model's classification accuracy and identify any potential misclassifications.

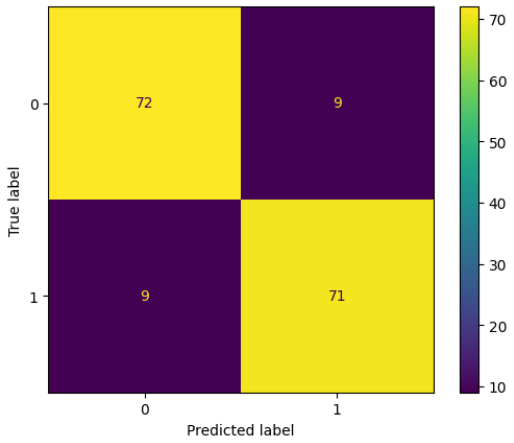

Figure 23: Confusion Matrix of CNN

Similarly, this confusion matrix evaluates the performance of FCNN model. It provides insights into the model's ability to correctly classify instances from the test dataset. The customized font sizes enhance the clarity of labels and values. The x-axis represents predicted labels, while the y-axis represents true labels. By examining the distribution of predictions across different classes, we can assess the model's classification accuracy and identify areas for improvement.

5 Discussion

Thumbs_Up and Peace were successfully classified by both CNN and FCNN models. Compared to FCNN, the CNN model performs more accurately. The accuracy of CNN's validation and test data is 89%. However, we find that the accuracy of the FCNN is 74% for validation data and 86% for test data. The performances of both models were developed through numerous trials and errors. We experimented on both models throughout varying epochs and learning rates. Large

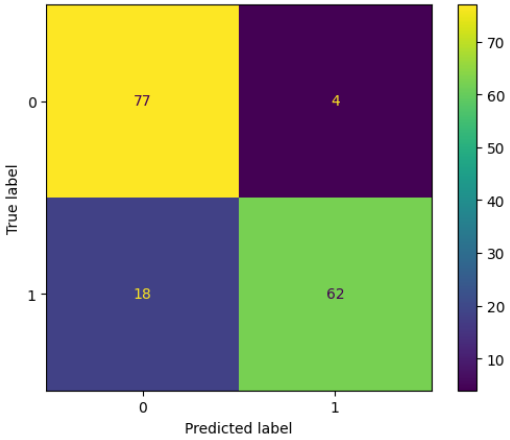

Figure 24: Confusion Matrix of FCNN

datasets allow us to improve the model's performance.