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

Md. Shahin Alam

Student ID: 1810176115

Contents

1 Experiments

We built a project to classify thumbs up and peace signs using deep learning binary classification. From this experiment, we can detect two distinct symbols of sign language. These gestures hold significant importance in various contexts. We used two architecture models - the first is CNN-based with 2 convolution layers; the other is a Fully Connected neural net model.

1.1 CNN Architecture

Input Layer:

- Input shape: (224, 224, 3)

Convolutional Blocks:
Block 1:

- Two convolutional layers with 32 filters each.
- Filter size: (3, 3) for each convolution.
- ReLU activation function.
- Padding is set to 'same' to preserve the spatial dimensions.
- MaxPooling layer with pool size (2, 2).

Block 2:

- Two convolutional layers with 64 filters each.
- Filter size: (3, 3) for each convolution.
- ReLU activation function.
- Padding is set to 'same'.
- MaxPooling layer with pool size (2, 2).

Block 3:

- Three convolutional layers with 128 filters each.
- Filter size: (3, 3) for each convolution.
- ReLU activation function.
- Padding is set to 'same'.
- AveragePooling layer with pool size (2, 2).

Flatten Layer:

- Flatten the output from the convolutional layers to a 1D tensor.

Fully Connected Layers:

- Dense layer with 512 neurons and ReLU activation (`fc1_custom`).
- Dense layer with 256 neurons and ReLU activation (`fc2_custom`).

Output Layer:

- Dense layer with 2 neurons (assuming it's for binary classification) and softmax activation (`predictions_custom`).

So, there are three convolutional blocks. Each convolutional block contains multiple convolutional layers, followed by a pooling layer. There are two fully connected layers following the flatten layer. The output layer has two neurons.

1.2 FCNN Architecture

Input Layer:

- Input shape: (224, 224, 3)

Flatten Layer:

- Flattens the input tensor into a 1D tensor.

Fully Connected Layers:

- Dense layer with 32 neurons and ReLU activation.
- Dense layer with 64 neurons and ReLU activation.
- Dense layer with 32 neurons and ReLU activation.

Output Layer:

- Dense layer with 1 neuron (assuming it's for binary classification) and sigmoid activation.

So, there is no convolutional layer in this architecture. There are three fully connected layers. The output layer has one neuron.

2 Dataset Description

2.1 Source of data

- There are two types of data.
  i) Thumbs-up sign
  ii) Peace sign

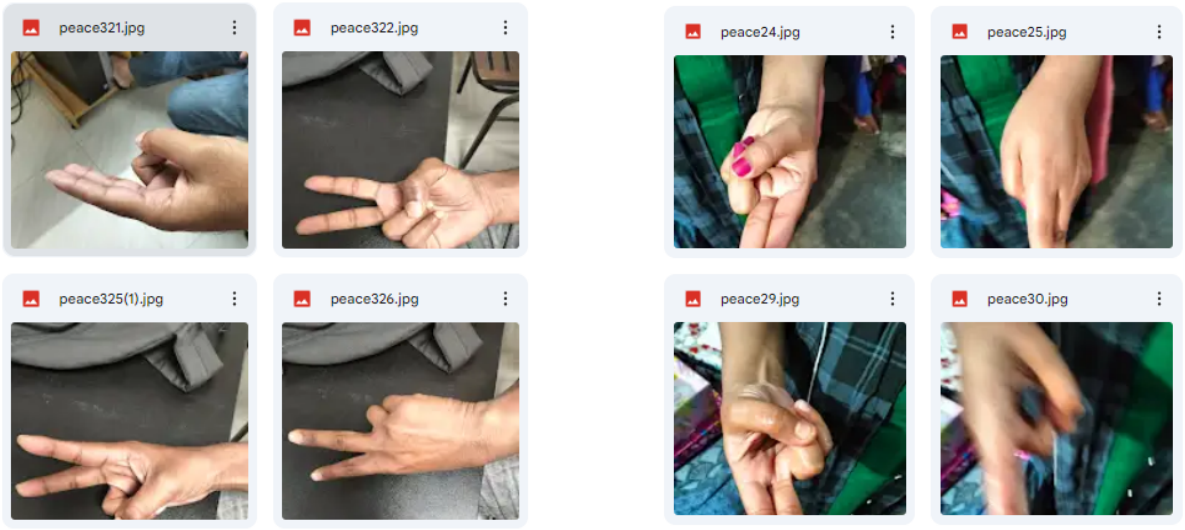

Figure 1: Peace sign test and train dataset

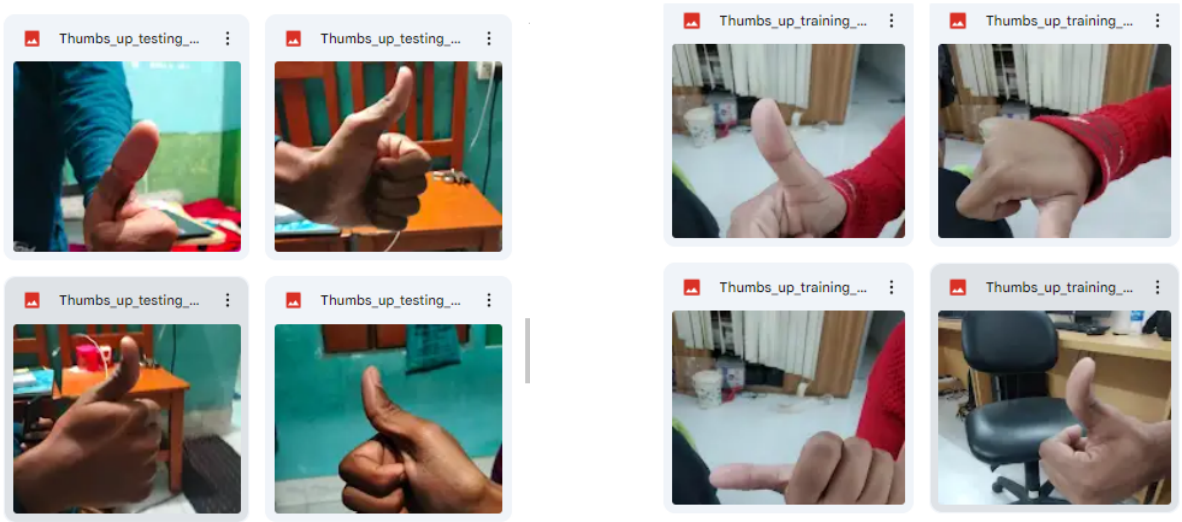

Figure 2: Thumbs-up sign test and train dataset

- We have captured 800 (400 images for each classification) RGB images with the help of conventional mobile camera.
- We have collected the dataset from 100 person by capturing 8 images from each person in different angle. All the images are taken with permission from individual.

- All the images are kept into the google drive after collecting from our group members.
- Drive link: https://drive.google.com/drive/u/3/folders/1DSKOBP34OtMqk2Vxmd2pEsHfAZpjpx0d

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:

- For each category ("Peace" and "Thumbs-up" signs):
  - Divide the data into training and testing sets.
  - Reserve 80% of the data for training and 20% for testing.

Training Set Splitting:

- For the training data:
  - Take 90% of the training data for actual training.
  - Reserve 10% of the training data for validation.

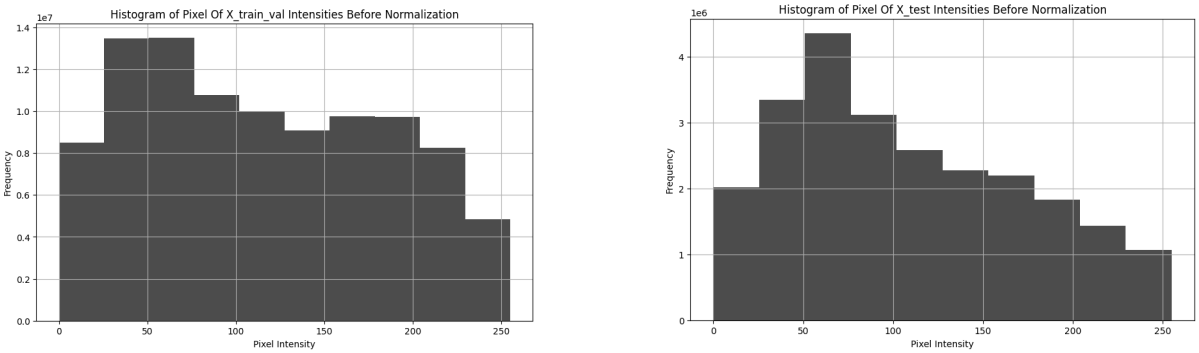

Figure 3: fig:X_train_val and X_test

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.

2.5 Labeling the Data

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

2.6 Shapes of Different data

- Shape of x trained data: (585, 224, 224, 3)
- Shape of x validation data: (65, 224, 224, 3)
- Shape of x test data: (161, 224, 224,
- Shape of y trained data: (585,)
- Shape of y validation data: (65,)
- Shape of y test data (161,)

3 Results

3.1 Total Data Overview

Total number of training and test data samples are printed to provide an overview of the dataset.

3.2 Pixel Intensity Distribution (Before Normalization)

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

3.3 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.

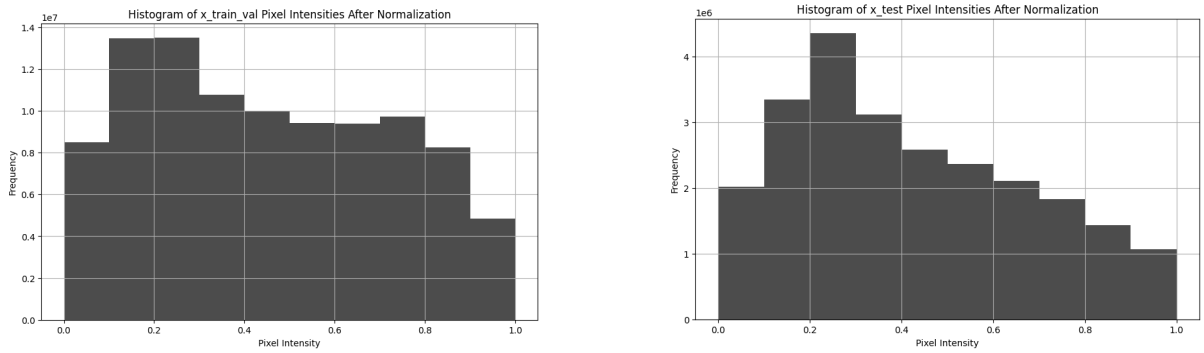

Figure 4: fig:X_train_val and X_test

3.4 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.

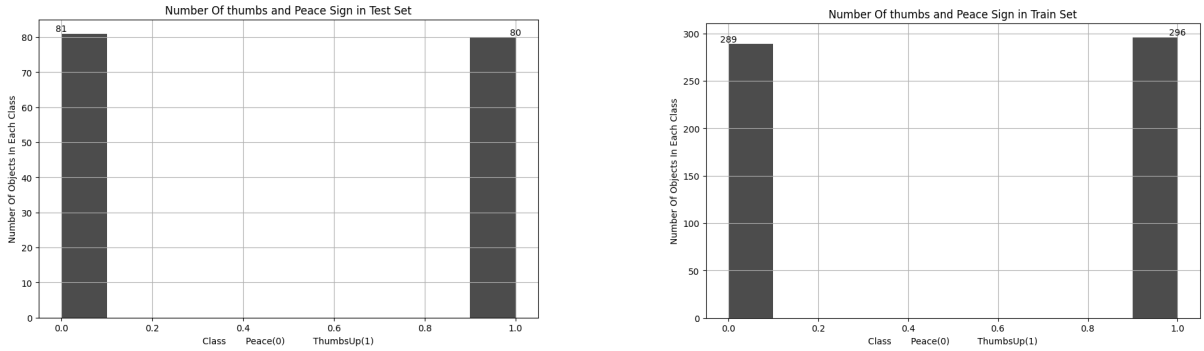

Figure 5: Test Set and Train Set

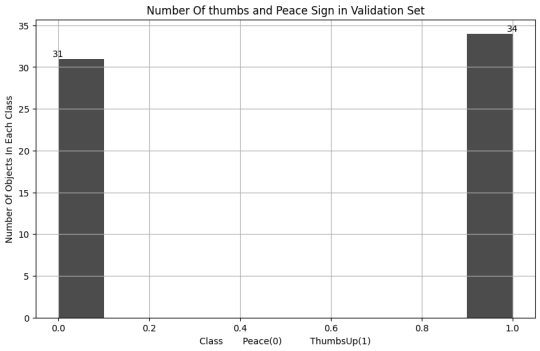

Figure 6: validation Set

3.5 Custom Neural Network Training

A custom neural network is built and trained with X and y data. The Performance_dict prints the training history, including metrics such as loss and accuracy, captured during model training.

3.6 Training Progress Visualization

Two subplots are used to visualize the training progress, showing the performance of 'Train Loss Vs Validation Loss' and 'Train Accuracy vs Validation Accuracy'.We also apply learning curves which indicate the performance of learing curves.

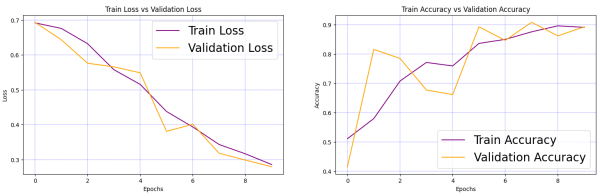

Figure 7: Train vs Validation test

3.7 Model Evaluation on Validation and Test Set

The model's performance is evaluated on the validation and test sets using the model.evaluate() function. Evaluation results are stored, and a confusion matrix of the CNN is generated to provide insights into classification accuracy compared to actual labels.We evaluate the model on validation set.The accuracy on validation data is 89 % and accuray of test data is 88.8%.

3.8 Prediction on Test Data

The model's predictions on the test data are checked.Learning curves for FCNN,We evaluate the FCNN model with Validation Data.we get accuray 74%.Here accuracy is decreased. When We evaluate the FCNN model on test Data,we get accuracy 86%.

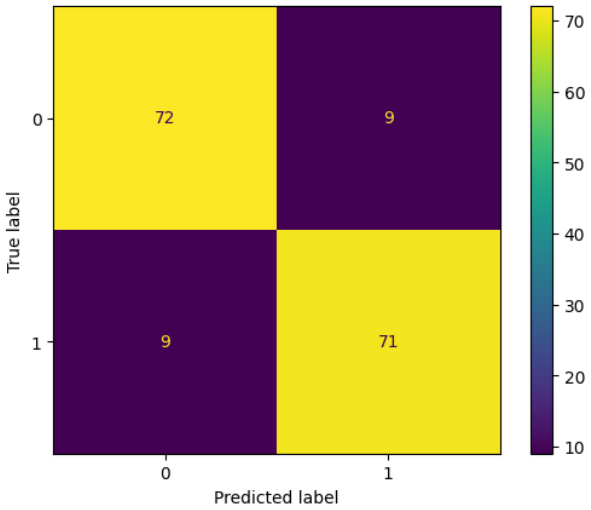

Figure 8: Confussion Matrix of CNN

3.9 FCNN Model Evaluation with Test Data

The FCNN model is evaluated on both validation and test data. Evaluation metrics include loss and accuracy. Additionally, a confusion matrix is generated to visualize classification performance on the test data.

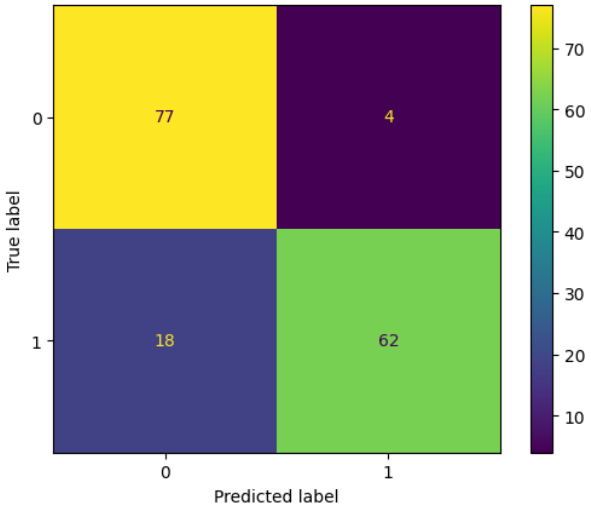

Figure 9: Confussion Matrix of FCNN

4 Discussion

4.1 Class Distribution Visualization

The histogram visualizes the distribution of classes within the training set and the validation set.

4.2 Training Progress Monitoring

Metrics such as loss and accuracy are essential for evaluating model performance and guiding adjustments.

4.3 Validation Set Evaluation

The model's performance on the validation data is assessed using evaluation metrics such as accuracy. The model's predictions on the test data are also checked.

4.4 Class and Text Prediction

Predicted class labels are converted to text labels ('Peace' or 'Thumbs Up') for improved readability and interpretation.

4.5 Confusion Matrix Interpretation

The confusion matrix evaluates the FCNN model's performance on test data, revealing classification accuracy and errors. True positives (TP) and true negatives (TN) indicate correct classifications, while false positives (FP) and false negatives (FN) highlight misclassifications.

4.6 FCNN Validation and Test Evaluation

The FCNN model is evaluated on validation and test datasets to assess its performance using metrics like accuracy. These evaluations provide insight into the model's classification abilities on unseen data.