UNIVERSITY PARTNER

# Artificial Intelligence and Machine Learning

# (6CS012)

Deep Learning CNN

Student Id : 2227097

Student Name : Radhika Neupane

Group: L6CG10

Module Leader : Siman Giri

Lecturer : Siman Giri

Tutor : Sunita Parajuli

Submitted on: 14th May 2024

## Title

**Image Classification of Traffic Signs**

## Abstract

This report has the overall process done for the image classification on the traffic signs by creating the models. We have the process of building the models along with evaluation and predictions done on those images. As a result, between FCNN,CNN and Transfer Learning, comparatively transfer learning gave more accuracy. Additionally this report has the feasibility of image classification done on **2227097_RadhikaNeupane_Code_ImageClassification.**

## Introduction:

For the CNN we have a dataset which consists of the traffic signs images, where images are grouped into 5 different categories. In these categories we have represented different types of traffic signs. From each category the images are stored separately and those data are organized in it. The main aim and the goal of this analysis is to accurately classify the unseen traffic signs using Convolutional Neural Networks (CNNs). It is one of the most used deep learning algorithms which is used for data processing those images which have the grid topology. The dataset consists of following images:

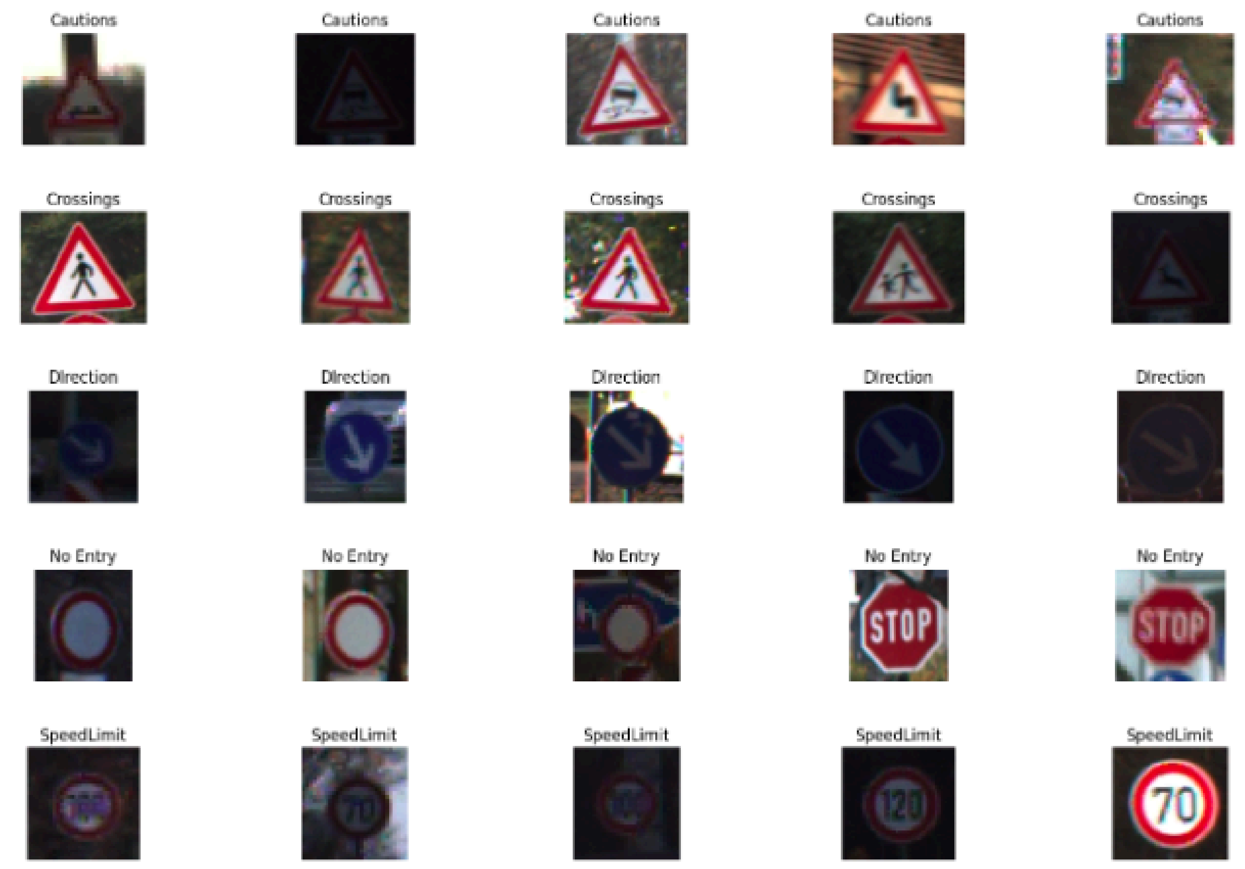
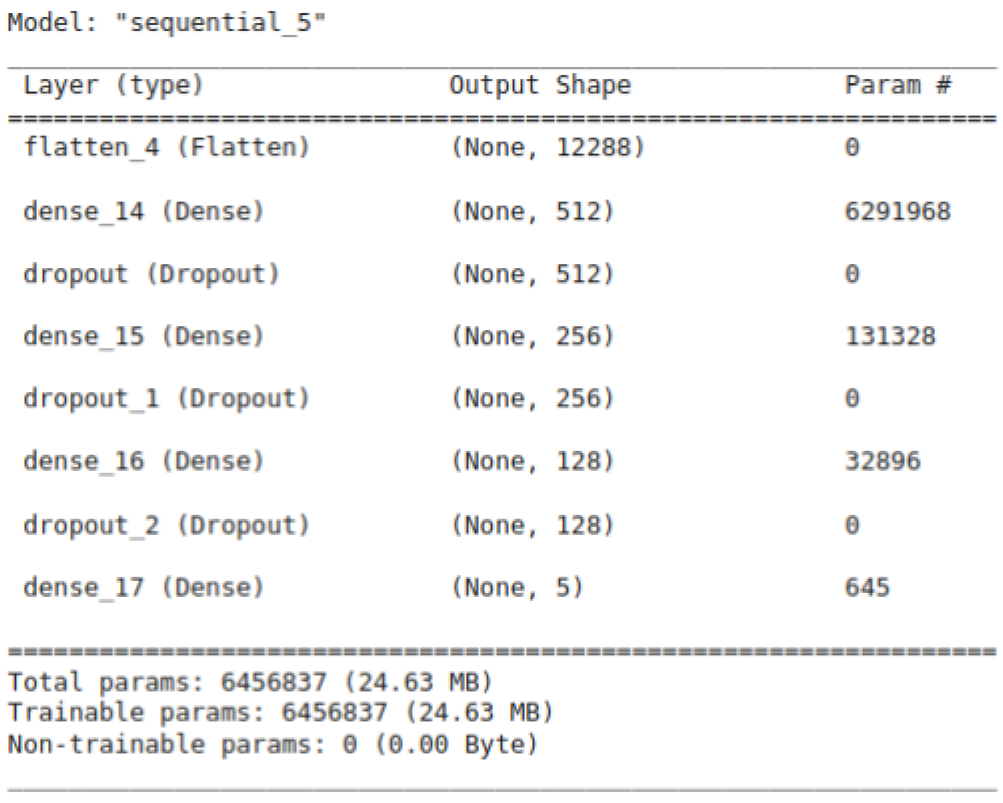


Fig 1: Dataset Images

## Image Classification with Fully Connected Neural Network

### Model Summary:

For the fully connected we have the flatten layer which is our input layer. This flatten the out of 2D shape into 1D vector. Then we have the dense layer with the number of neurons 512,256 and 128 along with the relu as a activation function. LIkewise we have a dropout layer after the dense player in order to prevent the overfitting with the dropout rate of 0.3 which means that almost 30% neurons will be dropped in random while doing training iteration. Then we have a final dense layer which consists of 5 neurons for 5 classes in the classification for our images.



```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_4 (Flatten)         (None, 12288)             0

 dense_14 (Dense)            (None, 512)               6291968

 dropout (Dropout)           (None, 512)               0

 dense_15 (Dense)            (None, 256)               131328

 dropout_1 (Dropout)         (None, 256)               0

 dense_16 (Dense)            (None, 128)               32896

 dropout_2 (Dropout)         (None, 128)               0

 dense_17 (Dense)            (None, 5)                 645

=================================================================
Total params: 6456837 (24.63 MB)
Trainable params: 6456837 (24.63 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Fig 2: FCNN Model Summary

## Training of the Model:

Finally the model is compiled by using the adam optimizer and with learning rate 0.001. For the loss function we have categorical cross-entropy. The model is trained for the 15 epochs. Likewise the training behavior shows that both the training and the validation accuracy is improving gradually.

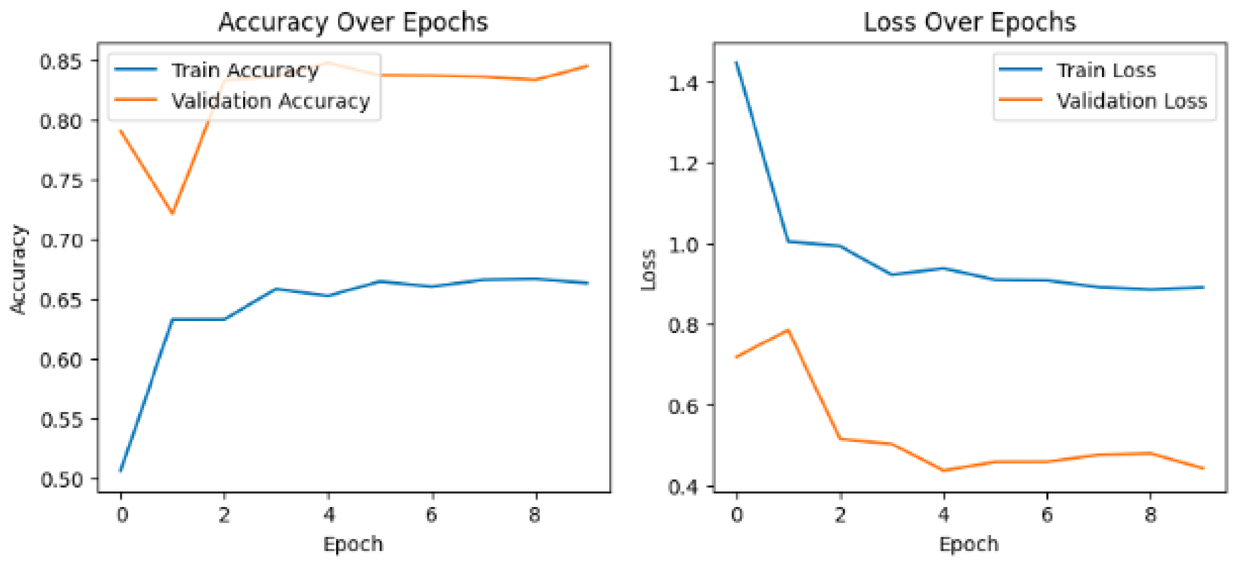
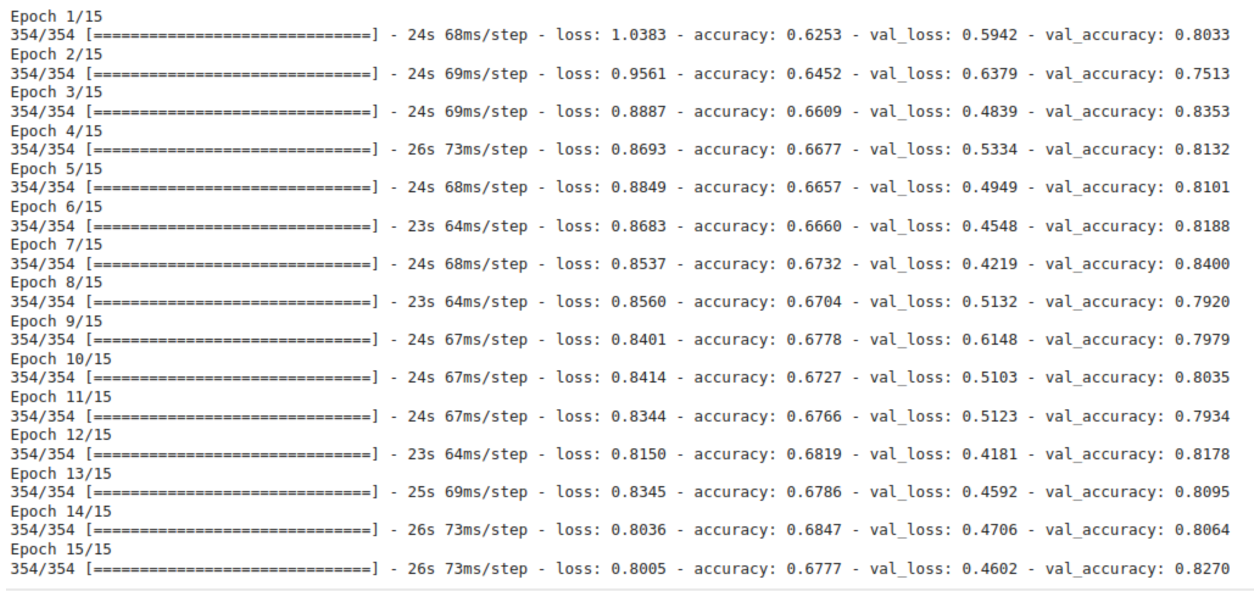


Fig 3: Validation and Train Accuracy and Loss



```
Epoch 1/15
354/354 [==============================] - 24s 68ms/step - loss: 1.0383 - accuracy: 0.6253 - val_loss: 0.5942 - val_accuracy: 0.8033
Epoch 2/15
354/354 [==============================] - 24s 69ms/step - loss: 0.9561 - accuracy: 0.6452 - val_loss: 0.6379 - val_accuracy: 0.7513
Epoch 3/15
354/354 [==============================] - 24s 69ms/step - loss: 0.8887 - accuracy: 0.6609 - val_loss: 0.4839 - val_accuracy: 0.8353
Epoch 4/15
354/354 [==============================] - 26s 73ms/step - loss: 0.8693 - accuracy: 0.6677 - val_loss: 0.5334 - val_accuracy: 0.8132
Epoch 5/15
354/354 [==============================] - 24s 68ms/step - loss: 0.8849 - accuracy: 0.6657 - val_loss: 0.4949 - val_accuracy: 0.8101
Epoch 6/15
354/354 [==============================] - 23s 64ms/step - loss: 0.8683 - accuracy: 0.6660 - val_loss: 0.4548 - val_accuracy: 0.8188
Epoch 7/15
354/354 [==============================] - 24s 68ms/step - loss: 0.8537 - accuracy: 0.6732 - val_loss: 0.4219 - val_accuracy: 0.8400
Epoch 8/15
354/354 [==============================] - 23s 64ms/step - loss: 0.8560 - accuracy: 0.6704 - val_loss: 0.5132 - val_accuracy: 0.7920
Epoch 9/15
354/354 [==============================] - 24s 67ms/step - loss: 0.8401 - accuracy: 0.6778 - val_loss: 0.6148 - val_accuracy: 0.7979
Epoch 10/15
354/354 [==============================] - 24s 67ms/step - loss: 0.8414 - accuracy: 0.6727 - val_loss: 0.5103 - val_accuracy: 0.8035
Epoch 11/15
354/354 [==============================] - 24s 67ms/step - loss: 0.8344 - accuracy: 0.6766 - val_loss: 0.5123 - val_accuracy: 0.7934
Epoch 12/15
354/354 [==============================] - 23s 64ms/step - loss: 0.8150 - accuracy: 0.6819 - val_loss: 0.4181 - val_accuracy: 0.8178
Epoch 13/15
354/354 [==============================] - 25s 69ms/step - loss: 0.8345 - accuracy: 0.6786 - val_loss: 0.4592 - val_accuracy: 0.8095
Epoch 14/15
354/354 [==============================] - 26s 73ms/step - loss: 0.8036 - accuracy: 0.6847 - val_loss: 0.4706 - val_accuracy: 0.8064
Epoch 15/15
354/354 [==============================] - 26s 73ms/step - loss: 0.8005 - accuracy: 0.6777 - val_loss: 0.4602 - val_accuracy: 0.8270
```

Fig 4 : Training FCNN Model

## Findings and Discussion:

After training the data we can conclude that the validation accuracy was 0.8452 whereas validation loss was 0.4421. We can see the roc curve for our model.

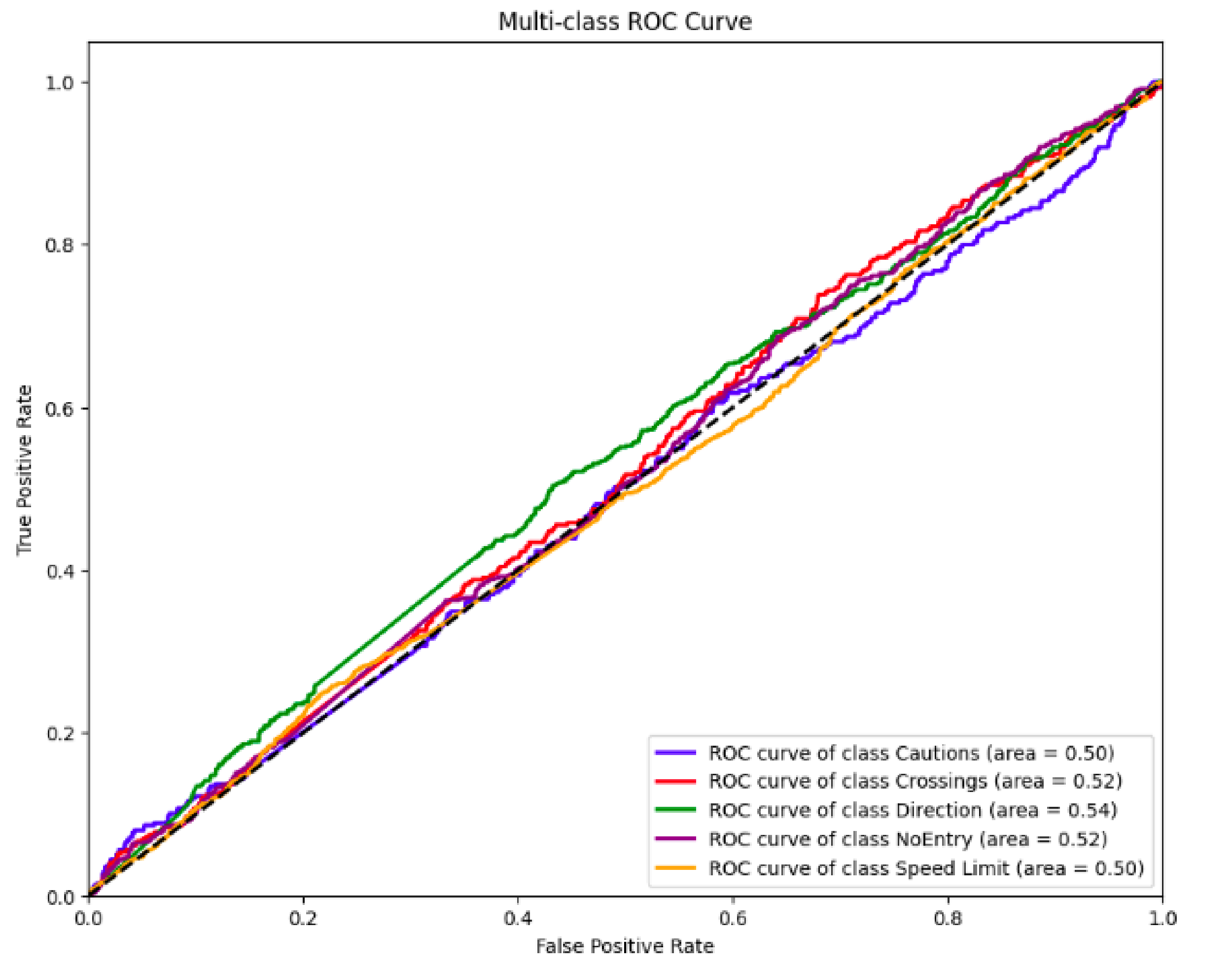


Fig 5 : ROC Curve for FCNN

For this FCNN model when the evaluation is done the test accuracy seems to be 0.6399 and loss is 25.44186

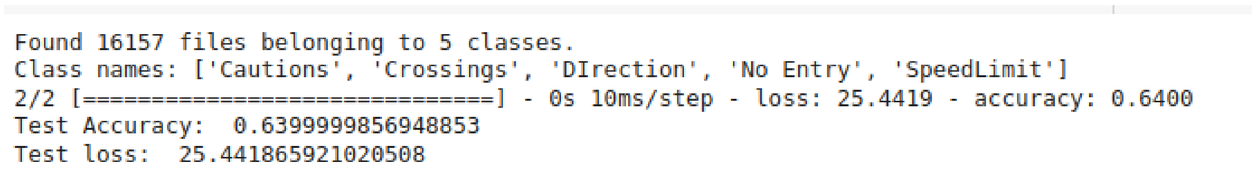

```
Found 16157 files belonging to 5 classes.
Class names: ['Cautions', 'Crossings', 'DIrection', 'No Entry', 'SpeedLimit']
2/2 [==============================] - 0s 10ms/step - loss: 25.4419 - accuracy: 0.6400
Test Accuracy:  0.6399999856948853
Test loss:  25.441865921020508
```

Fig 6: Test accuracy and loss

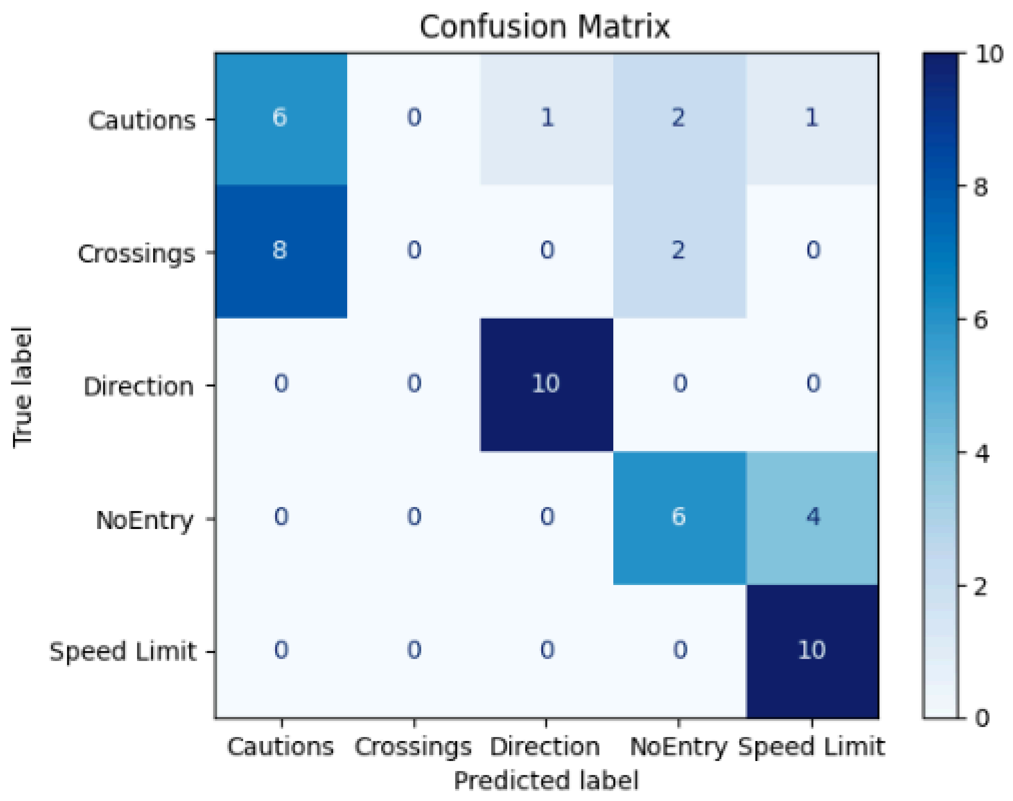
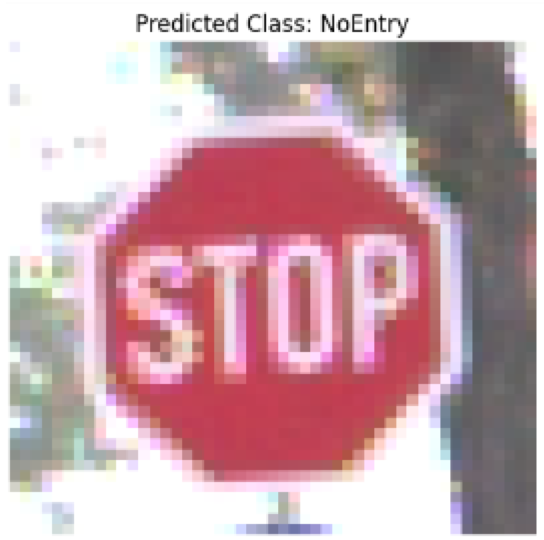


Fig 7 : Confusion Matrix for FCNN

The images are stored in the directory and are predicted using the model which is shown in the image below which showed us that this model worked fairly for our data.



```
/1 [==============================] - 0s 47ms/step
mage: 00008_00013_00018.png, Predicted Class: Speed Limit
```

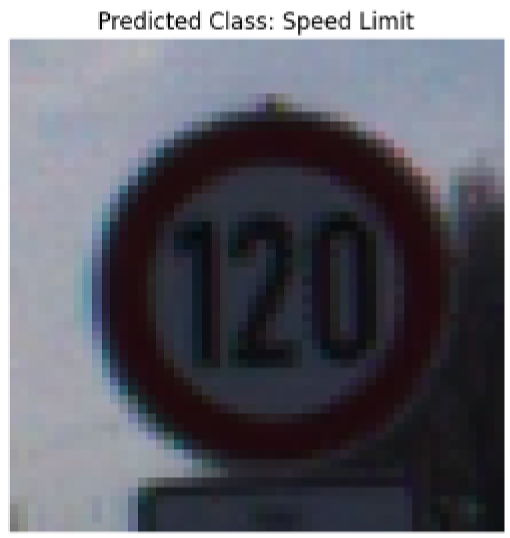
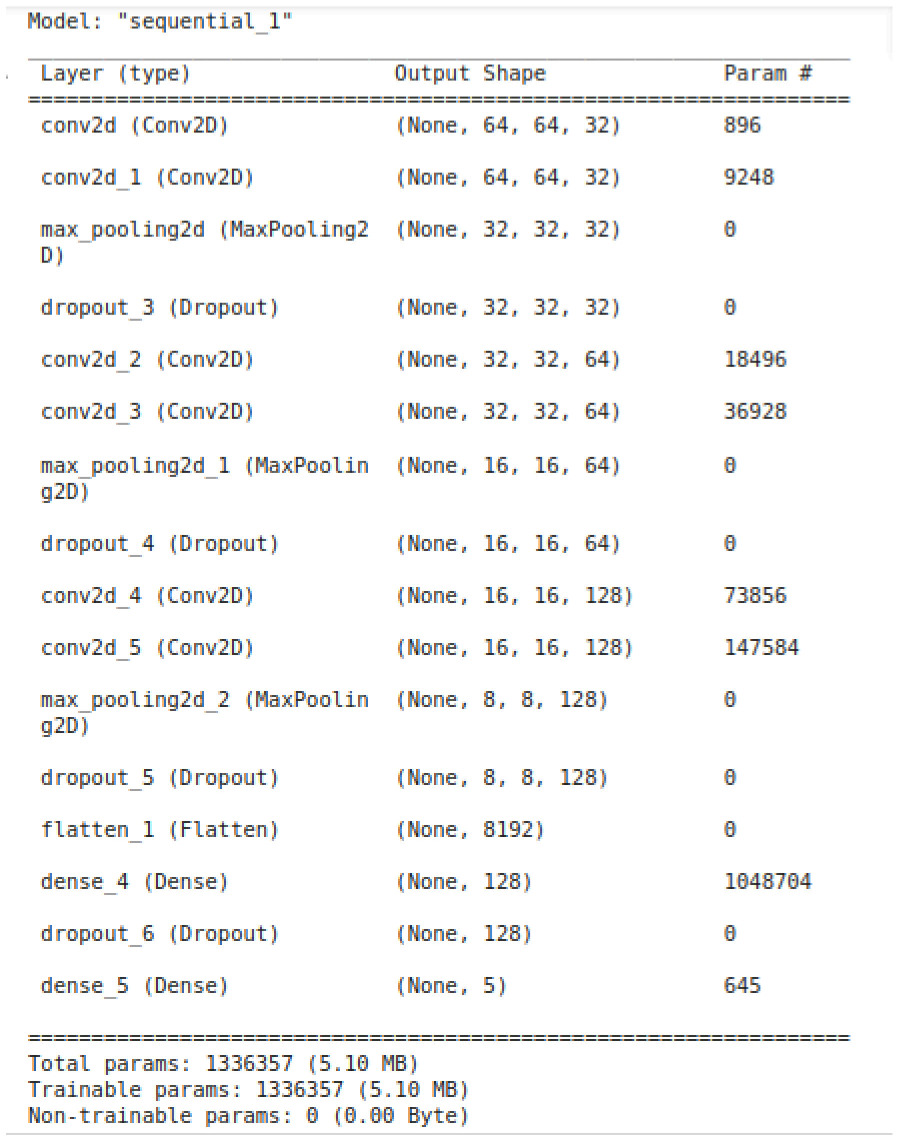


Fig 8 : Predicted Image by FCNN

## Image Classification with Convolutional Neural Network

### Model Summary:

For our model we have 3 convolutional layers which is followed by the max-pooling layer and the dropout layer. Thera is a convolutional layer which performs the convolution operations in order to extract the features from the input images. Similarly we have MaxPooling2D layers and dropout layers to prevent the overfitting. We also have the flatten layer to flatten the output from the convolutional layers into a 1D vector. For the filter size we have have 3*3 and the number of filters in each of the convolutional layers is 32,64 and 128. The number of fractions of neurons to be dropped out during training are 0.25 and 0.4. Finally we have ReLU activation function in convolution and dense layers and finally softmax activation function for output layer.

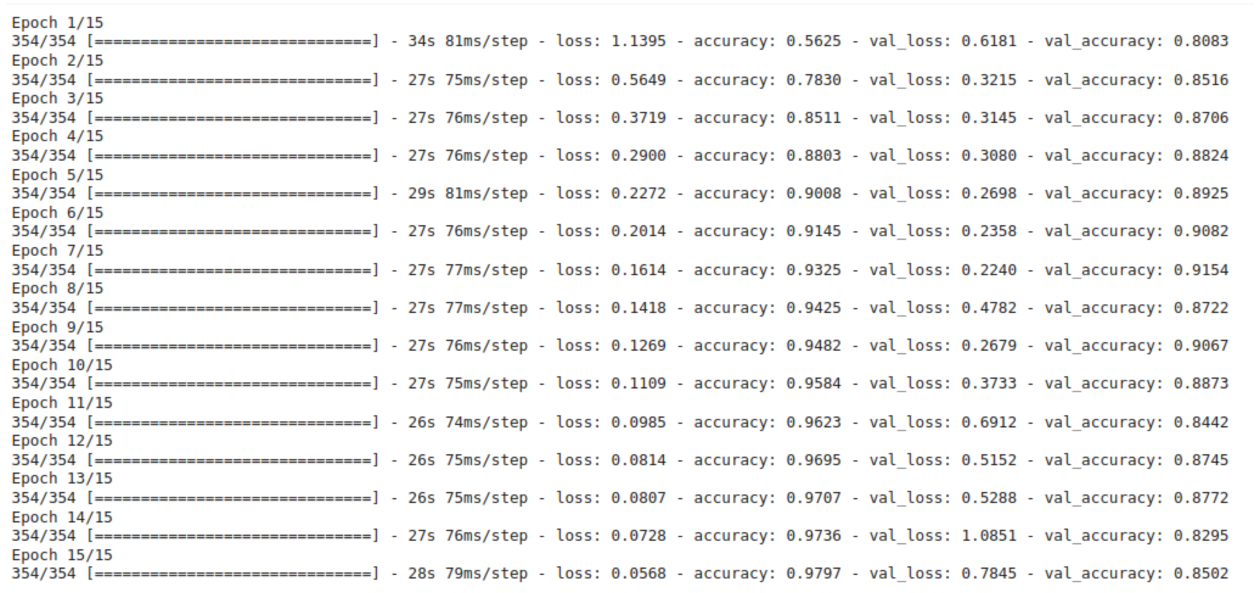


```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 64, 64, 32)        896

 conv2d_1 (Conv2D)           (None, 64, 64, 32)        9248

 max_pooling2d (MaxPooling2  (None, 32, 32, 32)        0
 D)

 dropout_3 (Dropout)         (None, 32, 32, 32)        0

 conv2d_2 (Conv2D)           (None, 32, 32, 64)        18496

 conv2d_3 (Conv2D)           (None, 32, 32, 64)        36928

 max_pooling2d_1 (MaxPoolin  (None, 16, 16, 64)        0
 g2D)

 dropout_4 (Dropout)         (None, 16, 16, 64)        0

 conv2d_4 (Conv2D)           (None, 16, 16, 128)       73856

 conv2d_5 (Conv2D)           (None, 16, 16, 128)       147584

 max_pooling2d_2 (MaxPoolin  (None, 8, 8, 128)         0
 g2D)

 dropout_5 (Dropout)         (None, 8, 8, 128)         0

 flatten_1 (Flatten)         (None, 8192)              0

 dense_4 (Dense)             (None, 128)               1048704

 dropout_6 (Dropout)         (None, 128)               0

 dense_5 (Dense)             (None, 5)                 645

=================================================================
Total params: 1336357 (5.10 MB)
Trainable params: 1336357 (5.10 MB)
Non-trainable params: 0 (0.00 Byte)
```

Fig 9 : Model Summary for CNN

## Training of the Model:

Categorical Crossentropy is used as the loss function used to train the model that is suitable for multi-class classification. Adam is used with a learning rate of 0.0001 as the optimizer. The model was trained for the 15 epochs.



```
Epoch 1/15
354/354 [==============================] - 34s 81ms/step - loss: 1.1395 - accuracy: 0.5625 - val_loss: 0.6181 - val_accuracy: 0.8083
Epoch 2/15
354/354 [==============================] - 27s 75ms/step - loss: 0.5649 - accuracy: 0.7830 - val_loss: 0.3215 - val_accuracy: 0.8516
Epoch 3/15
354/354 [==============================] - 27s 76ms/step - loss: 0.3719 - accuracy: 0.8511 - val_loss: 0.3145 - val_accuracy: 0.8706
Epoch 4/15
354/354 [==============================] - 27s 76ms/step - loss: 0.2900 - accuracy: 0.8803 - val_loss: 0.3080 - val_accuracy: 0.8824
Epoch 5/15
354/354 [==============================] - 29s 81ms/step - loss: 0.2272 - accuracy: 0.9008 - val_loss: 0.2698 - val_accuracy: 0.8925
Epoch 6/15
354/354 [==============================] - 27s 76ms/step - loss: 0.2014 - accuracy: 0.9145 - val_loss: 0.2358 - val_accuracy: 0.9082
Epoch 7/15
354/354 [==============================] - 27s 77ms/step - loss: 0.1614 - accuracy: 0.9325 - val_loss: 0.2240 - val_accuracy: 0.9154
Epoch 8/15
354/354 [==============================] - 27s 77ms/step - loss: 0.1418 - accuracy: 0.9425 - val_loss: 0.4782 - val_accuracy: 0.8722
Epoch 9/15
354/354 [==============================] - 27s 76ms/step - loss: 0.1269 - accuracy: 0.9482 - val_loss: 0.2679 - val_accuracy: 0.9067
Epoch 10/15
354/354 [==============================] - 27s 75ms/step - loss: 0.1109 - accuracy: 0.9584 - val_loss: 0.3733 - val_accuracy: 0.8873
Epoch 11/15
354/354 [==============================] - 26s 74ms/step - loss: 0.0985 - accuracy: 0.9623 - val_loss: 0.6912 - val_accuracy: 0.8442
Epoch 12/15
354/354 [==============================] - 26s 75ms/step - loss: 0.0814 - accuracy: 0.9695 - val_loss: 0.5152 - val_accuracy: 0.8745
Epoch 13/15
354/354 [==============================] - 26s 75ms/step - loss: 0.0807 - accuracy: 0.9707 - val_loss: 0.5288 - val_accuracy: 0.8772
Epoch 14/15
354/354 [==============================] - 27s 76ms/step - loss: 0.0728 - accuracy: 0.9736 - val_loss: 1.0851 - val_accuracy: 0.8295
Epoch 15/15
354/354 [==============================] - 28s 79ms/step - loss: 0.0568 - accuracy: 0.9797 - val_loss: 0.7845 - val_accuracy: 0.8502
```

Fig 10 : Training CNN Model

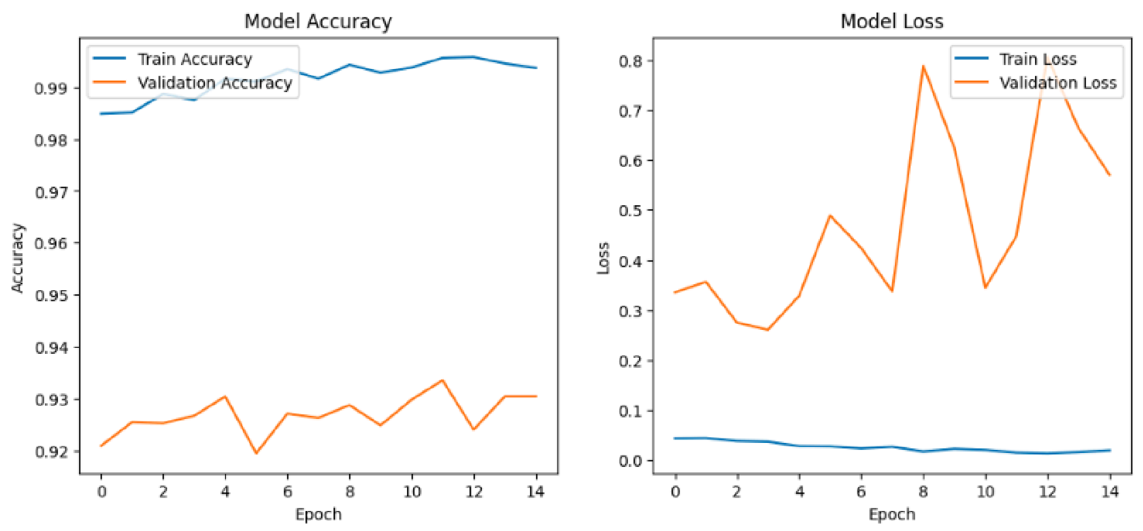


Fig 11: Validation and Train Accuracy and Loss

## Findings and Discussions:

We can see that the validation loss for this model is 0.785 and validation accuracy is 0.8502. Here is our ROC Curve for our -CNN model.

```
152/152 [==============================] - 3s 21ms/step
```

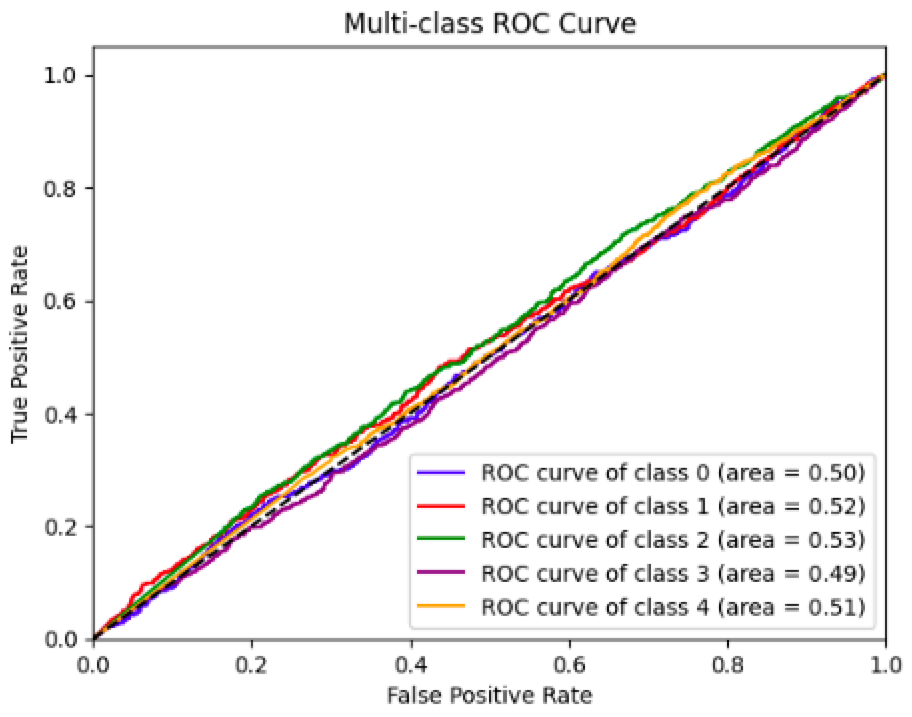
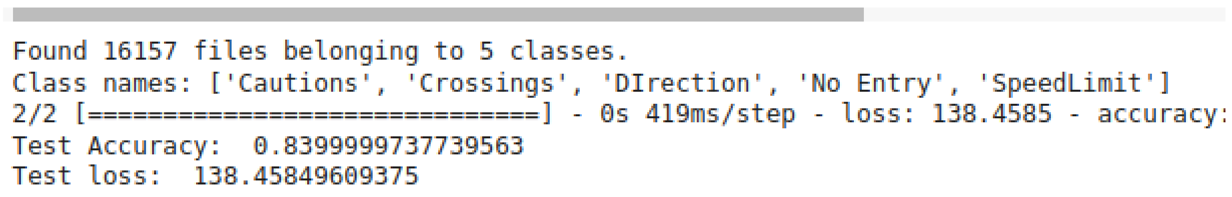


Fig 12:: ROC Curve for CNN

For the test accuracy and loss we have 0.83999 and 134.4584 respectively.



```
Found 16157 files belonging to 5 classes.
Class names: ['Cautions', 'Crossings', 'DIrection', 'No Entry', 'SpeedLimit']
2/2 [==============================] - 0s 419ms/step - loss: 138.4585 - accuracy:
Test Accuracy:  0.8399999737739563
Test loss:  138.45849609375
```

Fig 13: Test accuracy and loss

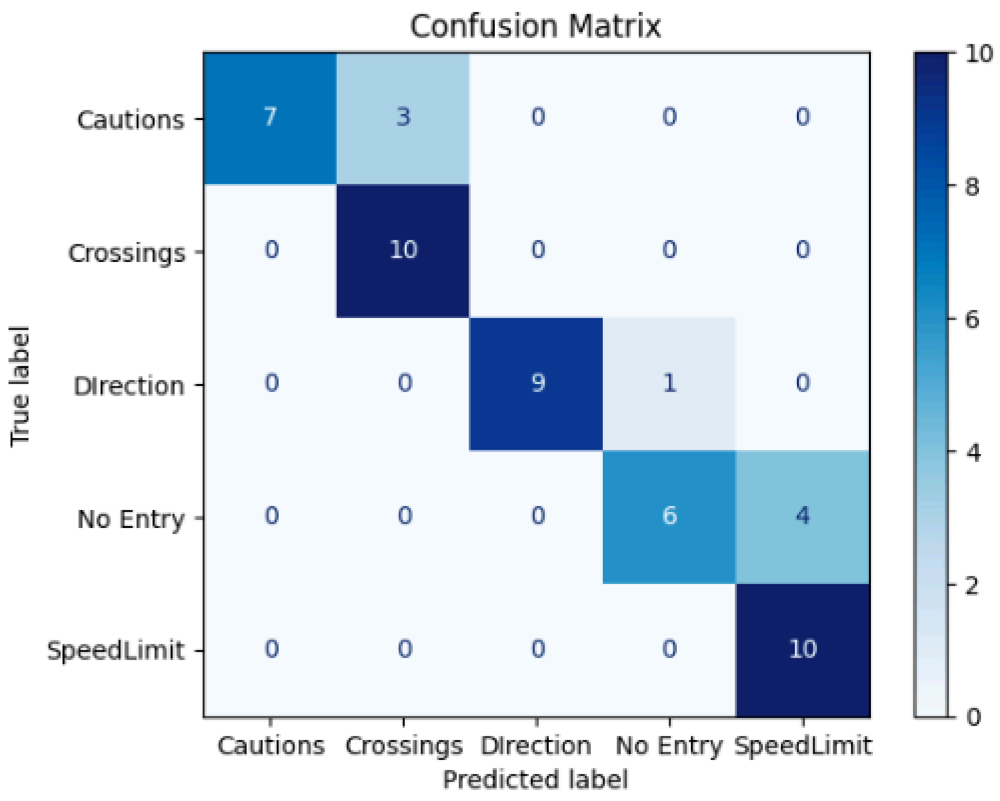
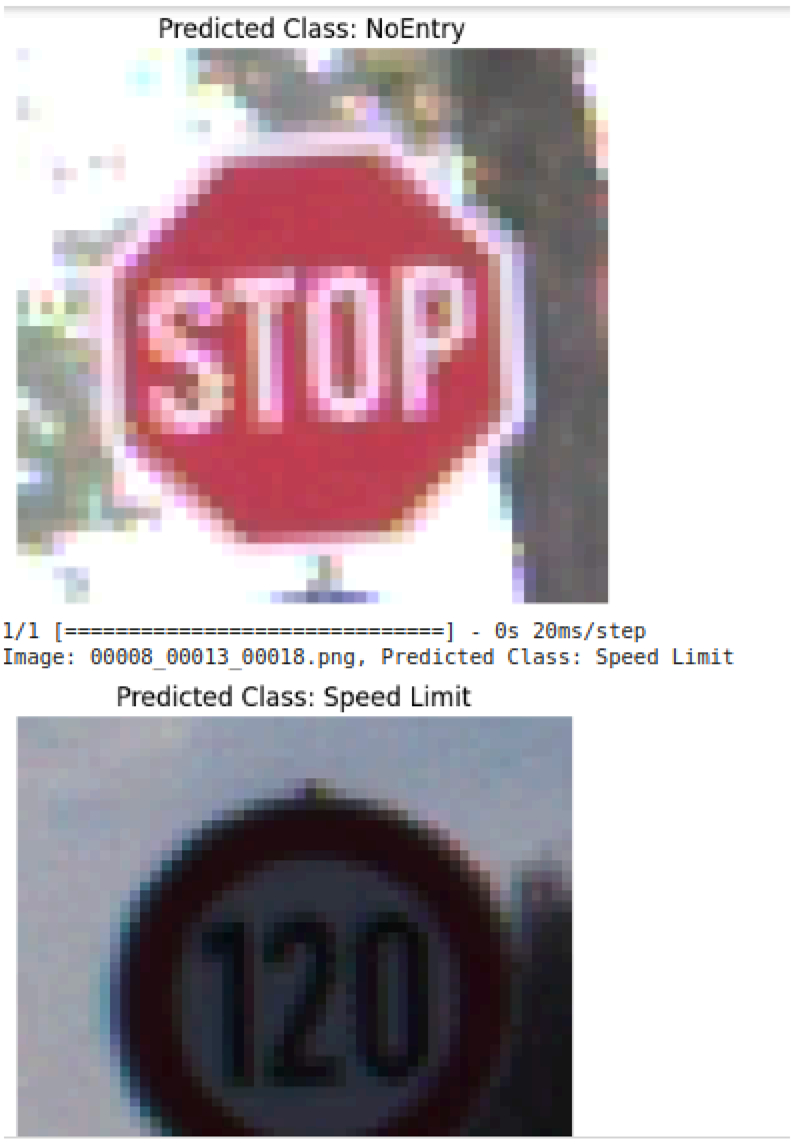


Fig 14 : Confusion Matrix for CNN

The image which are predicted are shown in the image below:



Predicted Class: NoEntry

```
1/1 [==============================] - 0s 20ms/step
Image: 00008_00013_00018.png, Predicted Class: Speed Limit
```

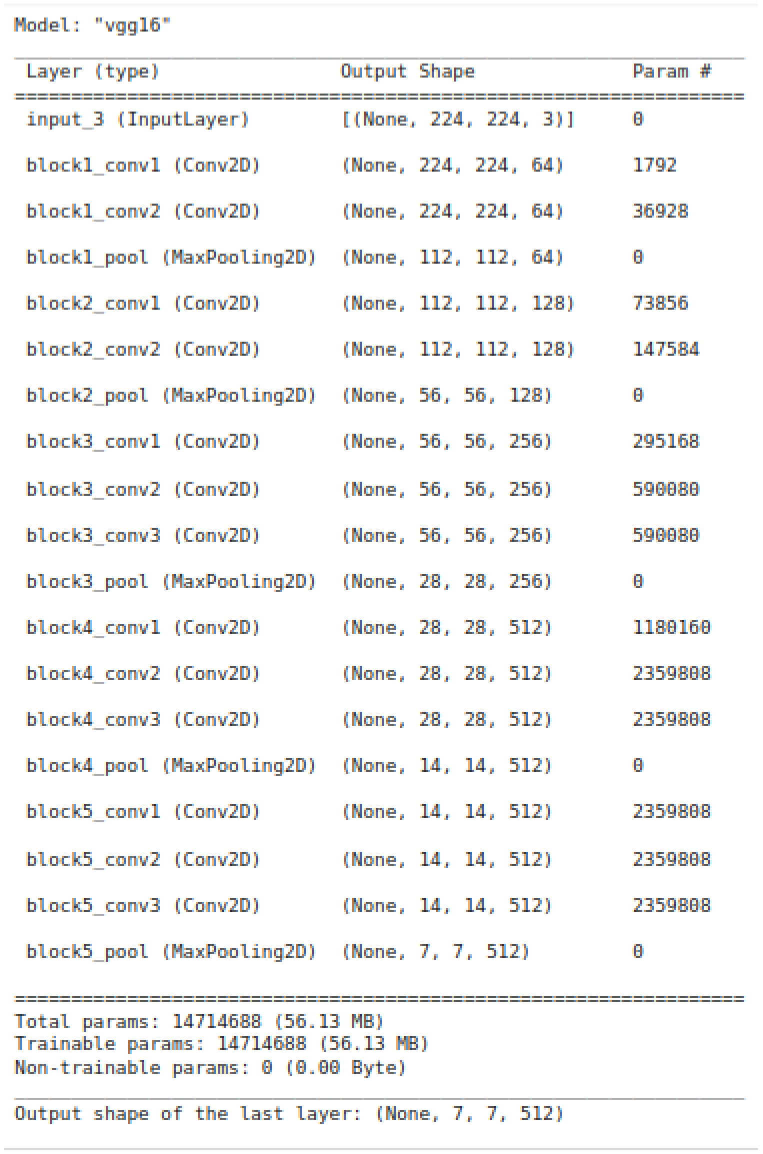
Predicted Class: Speed Limit

Fig 15 : Predicted image by CNN

## Image Classification with Transfer Learning

### Model Summary:

The model we are using for Transfer Learning is a pre-trained VGG16 model which is available and provided by the tensorflow library and the weights that are loaded in the model are from the 'imagenet' weights.It has 16 layers with simple architecture having 13 convolutional layers and 3 dense layers(fully connected layers) where each of them look upon the different features of the images. Talking about the model hyperparameters our images are supposed to have a size of 224*224 pixels with 3 color channels with the activation function ReLU and drop out rate of 0.5. Then we have added transfer learning.Lastly we have a custom top layer for the model in order to perform classification tasks. Flatten is used to prepare the fully connected layers which have a dense layer of 512 units and an activation function called ReLU. We have added the dropout rate of 0.5 with softmax as the activation function and a dense layer of 5 units.

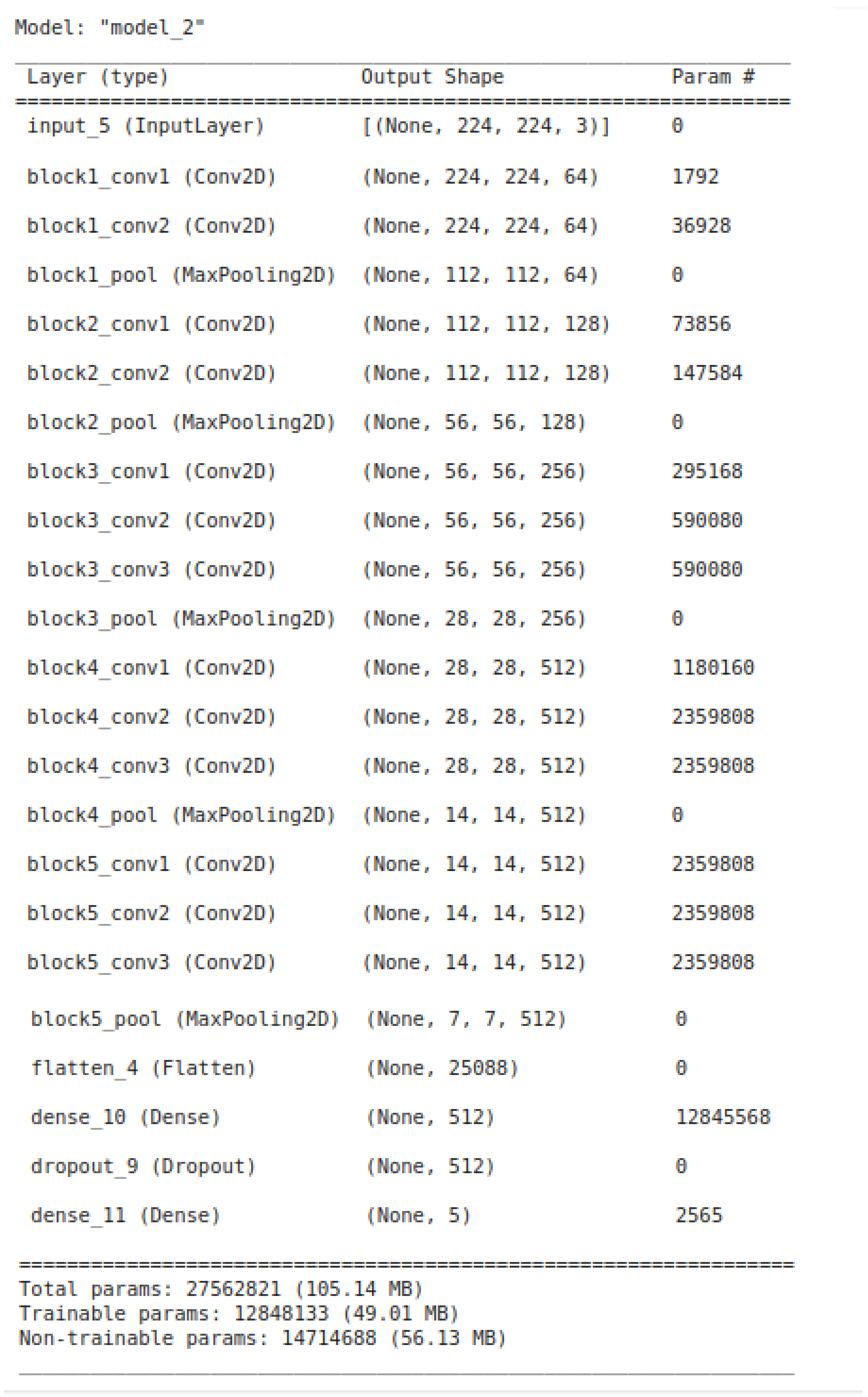


```
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_3 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

=================================================================
Total params: 14714688 (56.13 MB)
Trainable params: 14714688 (56.13 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Output shape of the last layer: (None, 7, 7, 512)
```

Fig 16 : Model Summary of Transfer Learning

## Training of Model:

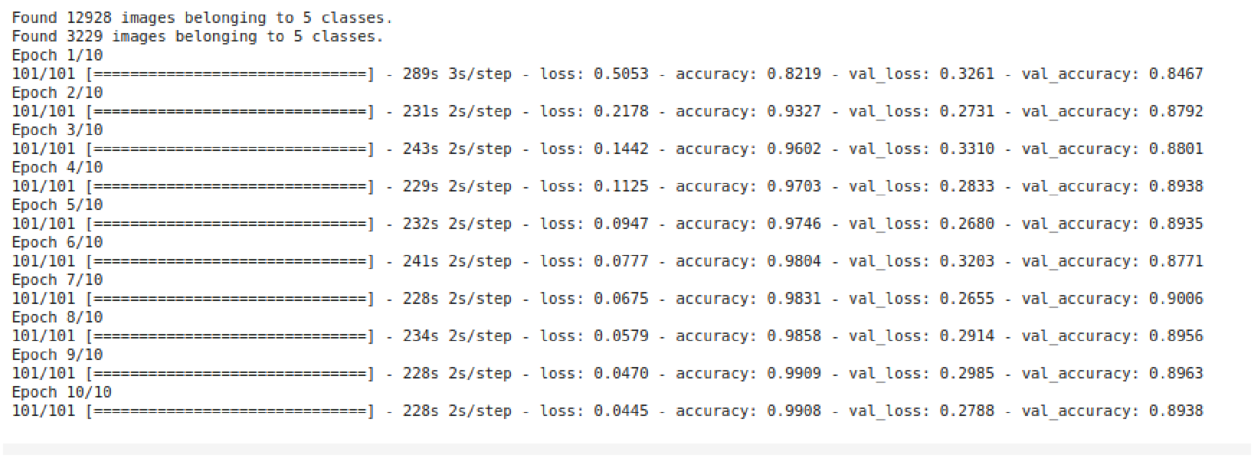
Here we have frozen the layers which ensures that their weights are not updated during the training process.These frozen layers allow the  models to utilize the pre-trained features.



```
Model: "model_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 input_5 (InputLayer)        [(None, 224, 224, 3)]     0         

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792      

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928     

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0         

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856     

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584    

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0         

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168    

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080    

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080    

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0         

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160   

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808   

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808   

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0         

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808   

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808   

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808   

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0         

 flatten_4 (Flatten)         (None, 25088)             0         

 dense_10 (Dense)            (None, 512)               12845568  

 dropout_9 (Dropout)         (None, 512)               0         

 dense_11 (Dense)            (None, 5)                 2565      

=================================================================
Total params: 27562821 (105.14 MB)
Trainable params: 12848133 (49.01 MB)
Non-trainable params: 14714688 (56.13 MB)
_________________________________________________________________
```

Fig 18 : Model Summary

We have used 10 epochs for our training process.



```
Found 12928 images belonging to 5 classes.
Found 3229 images belonging to 5 classes.
Epoch 1/10
101/101 [==============================] - 289s 3s/step - loss: 0.5053 - accuracy: 0.8219 - val_loss: 0.3261 - val_accuracy: 0.8467
Epoch 2/10
101/101 [==============================] - 231s 2s/step - loss: 0.2178 - accuracy: 0.9327 - val_loss: 0.2731 - val_accuracy: 0.8792
Epoch 3/10
101/101 [==============================] - 243s 2s/step - loss: 0.1442 - accuracy: 0.9602 - val_loss: 0.3310 - val_accuracy: 0.8801
Epoch 4/10
101/101 [==============================] - 229s 2s/step - loss: 0.1125 - accuracy: 0.9703 - val_loss: 0.2833 - val_accuracy: 0.8938
Epoch 5/10
101/101 [==============================] - 232s 2s/step - loss: 0.0947 - accuracy: 0.9746 - val_loss: 0.2680 - val_accuracy: 0.8935
Epoch 6/10
101/101 [==============================] - 241s 2s/step - loss: 0.0777 - accuracy: 0.9804 - val_loss: 0.3203 - val_accuracy: 0.8771
Epoch 7/10
101/101 [==============================] - 228s 2s/step - loss: 0.0675 - accuracy: 0.9831 - val_loss: 0.2655 - val_accuracy: 0.9006
Epoch 8/10
101/101 [==============================] - 234s 2s/step - loss: 0.0579 - accuracy: 0.9858 - val_loss: 0.2914 - val_accuracy: 0.8956
Epoch 9/10
101/101 [==============================] - 228s 2s/step - loss: 0.0470 - accuracy: 0.9909 - val_loss: 0.2985 - val_accuracy: 0.8963
Epoch 10/10
101/101 [==============================] - 228s 2s/step - loss: 0.0445 - accuracy: 0.9908 - val_loss: 0.2788 - val_accuracy: 0.8938
```

Fig 20 : Training process for Transfer Learning

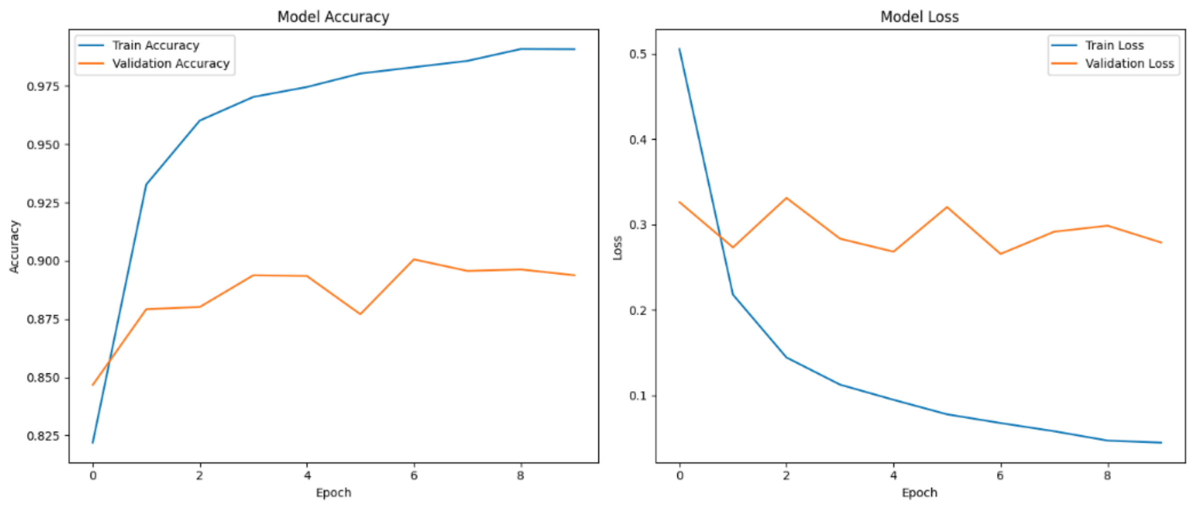


Fig 21: Validation and Train Accuracy and Loss

## Findings and Discussion:

Finally we had the validation loss and validation accuracy of 0.2788 and 0.8938 respectively.

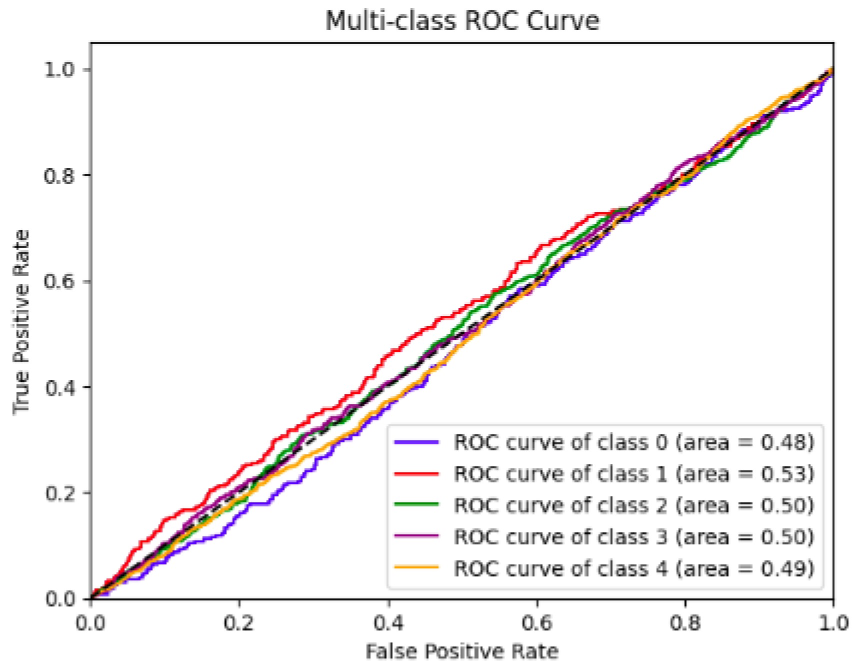
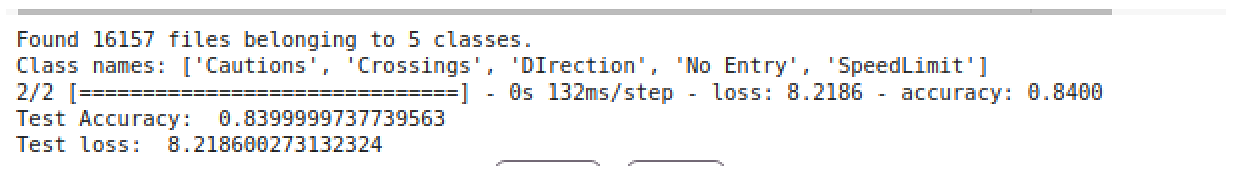


Fig 22: ROC Curve for Transfer Learning



```
Found 16157 files belonging to 5 classes.
Class names: ['Cautions', 'Crossings', 'DIrection', 'No Entry', 'SpeedLimit']
2/2 [==============================] - 0s 132ms/step - loss: 8.2186 - accuracy: 0.8400
Test Accuracy:  0.8399999737739563
Test loss:  8.218600273132324
```

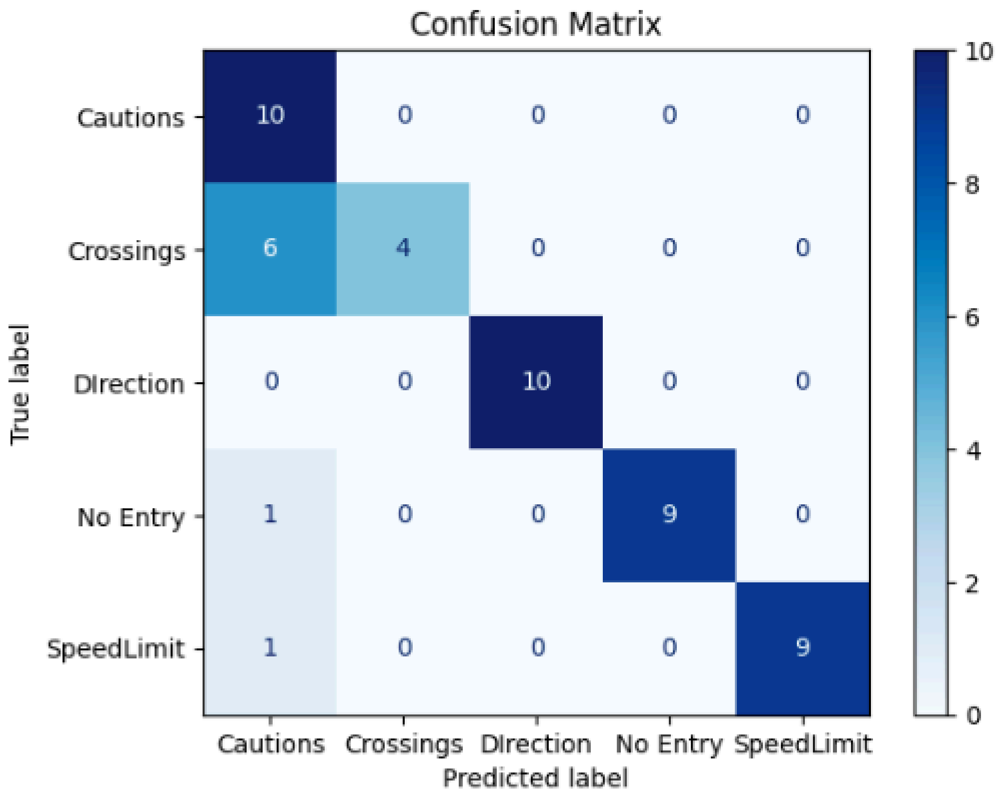
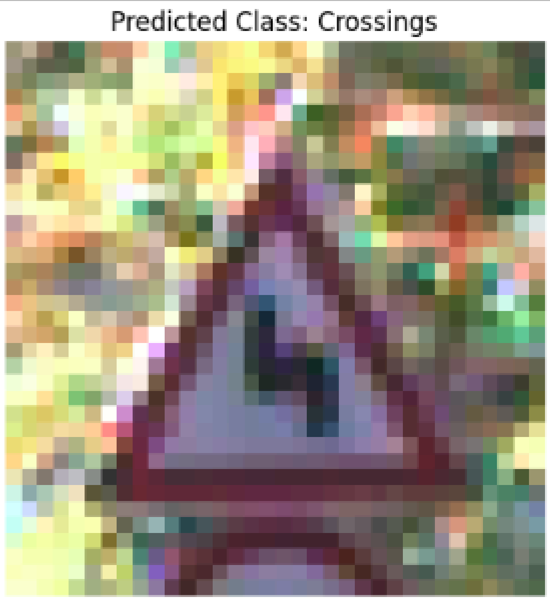


Fig 24: Confusion Matrix



```
1/1 [==============================] - 0s 28ms/step
Image: 00027_00000_00002.png, Predicted Class: Cautions
```

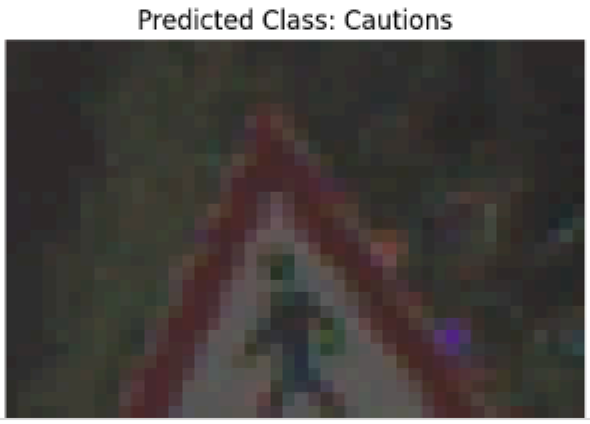


Fig 24: Predicted Image

## Final Discussion

When we compare our overall results of the model, the pre-trained model i.e., transfer learning gives us the most accurate result in comparison with others . After that CNN somehow works good but FCNN comparatively looks worse.