**Introduction:**

In this assignment, I conducted classification of non-vehicle and vehicle images. I used state of the art methods including Gray-Level Co-occurrence Matrix (GLCM) for capturing texture patterns, NumPy for extracting statistical features, FAST algorithm keypoints for identifying important points, and Canny edge detection to detect edges. To enhance the feature set, I considered the correlation of texture and statistical features with the target variable, and for classification, a Fully Connected Neural Network (FCNN) was employed.

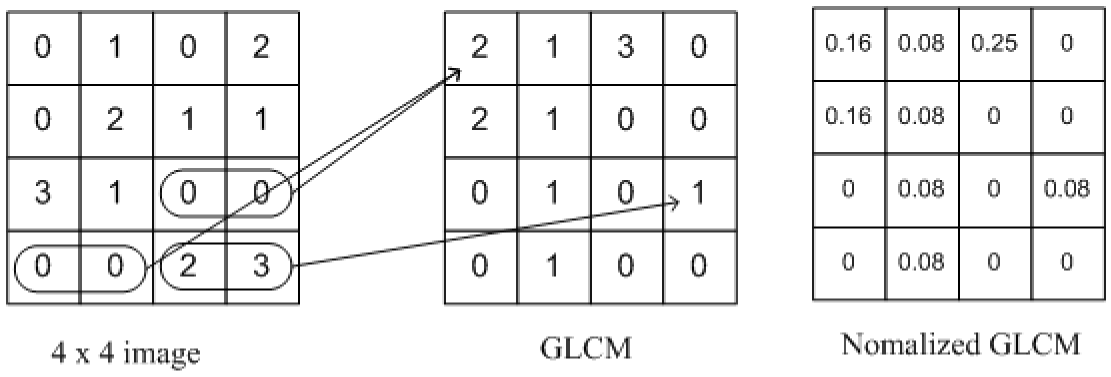
**Data Understanding:**

The dataset for this task consists of 17,760 images. One thing sets apart these 64x64x3 pixel images: they have been properly cropped to highlight only the most crucial visual elements, removing any extra background or unnecessary information. There are 8,968 images that are of non-vehicles and 8,792 images that are of vehicles.

**Feature Extraction:**

**GLCM:**

I examined texture in images using GLCM, or Gray-Level Co-occurrence Matrix. It provides information on the frequency of pairs of pixel values found in the image at particular distances and angles. I used a horizontal angle with distance one, which aids in the interpretation of textures and patterns.



| | | | |
|---|---|---|---|
| 0 | 1 | 0 | 2 |
| 0 | 2 | 1 | 1 |
| 3 | 1 | 0 | 0 |
| 0 | 0 | 2 | 3 |

| | | | |
|---|---|---|---|
| 2 | 1 | 3 | 0 |
| 2 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 |

| | | | |
|---|---|---|---|
| 0.16 | 0.08 | 0.25 | 0 |
| 0.16 | 0.08 | 0 | 0 |
| 0 | 0.08 | 0 | 0.08 |
| 0 | 0.08 | 0 | 0 |

**Numpy statistics:**

**Mean Intensity**: Usually related to overall light. mean intensity is the average brightness of each pixel in the image.

**Image contrast** is indicated by the standard deviation, which measures how much pixel values deviate from the mean.

**Root Mean Square Error**, or RMSE, measures how similar the image's pixel values are overall.

**GLCM Features:**

**Contrast**: Measures the difference between light and dark areas in the image.Higher contrast values indicate more contrast.

**Correlation:** The linear relationship between pixel values is described by correlation; higher values indicate a stronger linear relationship.

**Dissimilarity:** Calculates the variation in pixel values; regions with high values are considered dissimilar.

**Homogeneity**: Gives the closeness of pixel value pairs in the image.

**Energy:** Reflects how uniform the image texture is, with 1 indicating maximum uniformity.

**Other Features:**

**Aspect Ratio**: This function calculates the width to height ratio of an image. An image that is wider is indicated by a value larger than 1.

**Feature selection of these Features using Correlation:**

The five most important features that show the strongest linear relationships with the target variable were found using correlation analysis. Improving the model's understanding and predicted performance was the reason for choosing these important features.

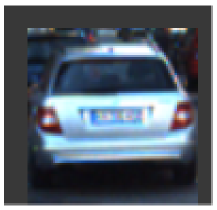
| **Feature** | **Correlation Value** |
|---|---|
| Mean Intensity | -0.175189 |
| Standard Deviation | 0.512027 |
| Contrast | 0.407801 |
| Energy | -0.231785 |
| Correlation | -0.161981 |
| Homogeneity | -0.303135 |
| Dissimilarity | 0.472436 |
| RMSE | -0.030552 |

|  |  |
|---|---|
| Aspect Ratio | NaN |
| Elongation | NaN |

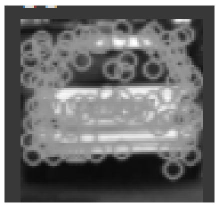
**Features from Accelerated Segment Test:**

In order to efficiently and rapidly extract image features, I employed a technique called FAST (Features from Accelerated Segment Test). The two most important pieces of information from the FAST keypoints that I used were the x and y coordinates. These coordinates were used to create, apply, and initialize with zeros a mask. By keeping only the crucial edge data for further processing and setting the values at the x and y coordinates to 255, we were then able to improve image classification.

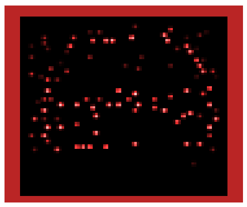
**Original Image:**



**FAST Keypoints Drawing**



**Keypoint Mask**



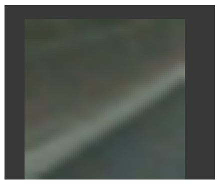
**Limitation of FAST:**

Fast keypoints may not be as effective in a variety of image conditions when applied to textures that are uniform or have low contrast, and they are not able to handle changes in scale or rotation.

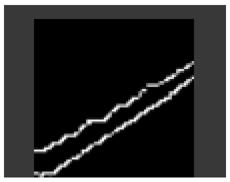
**Canny Edge Detection:**

Due to this limitation of Fast algorithm I used canny edge detection for feature extraction of some images that are not handled with Fast algorithm . Different thresholds for canny are defined due to the issue that some images are blurred or some images have no objects in it like images of sky or some images with only light colors . Therefore on high thresholds if no edge is found in these types of images that are discussed above then low thresholds are applied so that if any image has some edge it may be detected .
Also the same threshold is not applied directly to all images because it will then become sensitive to noise , as some images may have many edges so that we cannot consider all of them because it will then pick the surrounding points as noise.

**Original Image**



**Canny Edge**

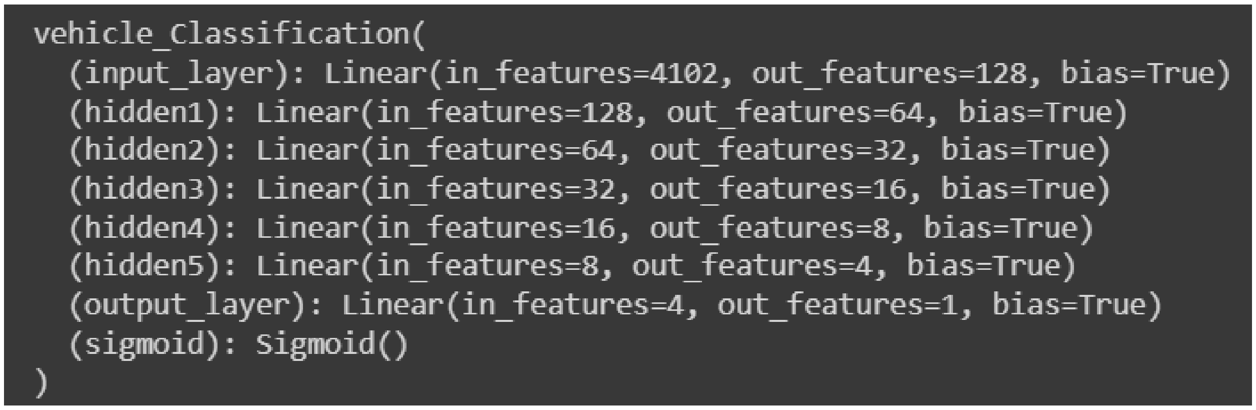


**Combined Features:**

I utilized my dataset of 17,760 images by using 4,098 features from the FAST and Canny edge detection techniques. To improve the accuracy of image classification, five statistical and texture features (a total of 4,104) were added to these original features.

**Neural Network:**

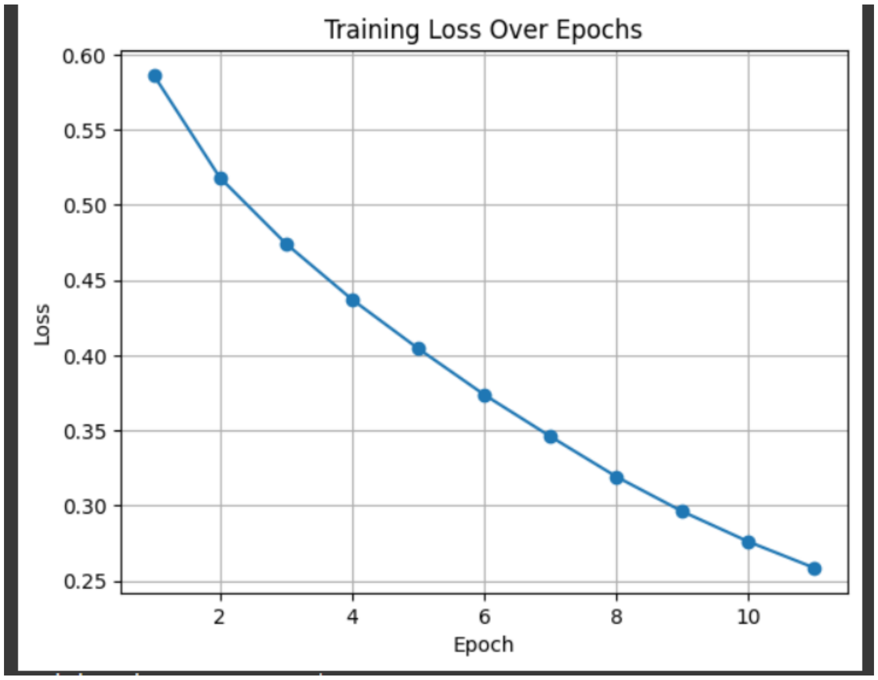
- ➢ The first hidden layer contains 128 neurons, and the input layer receives the input features.
- ➢ A deep neural network is formed by the 64, 32, 16, 8, and 4 neurons found in the ensuing hidden layers.
- ➢ One neuron with sigmoid activation makes up the output layer; this neuron is usually utilized for binary classification .
- ➢ Binary cross-entropy (BCE) loss, a popular option for binary classification tasks, was used to train the network.
- ➢ Using this architecture, the dataset was processed with a 64-batch size to maximize the model's accuracy in image classification.



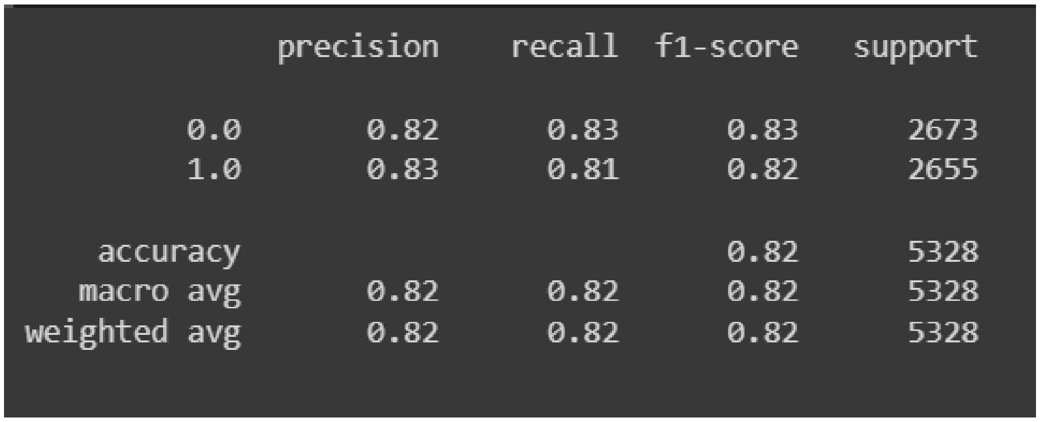
```
vehicle_Classification(
  (input_layer): Linear(in_features=4102, out_features=128, bias=True)
  (hidden1): Linear(in_features=128, out_features=64, bias=True)
  (hidden2): Linear(in_features=64, out_features=32, bias=True)
  (hidden3): Linear(in_features=32, out_features=16, bias=True)
  (hidden4): Linear(in_features=16, out_features=8, bias=True)
  (hidden5): Linear(in_features=8, out_features=4, bias=True)
  (output_layer): Linear(in_features=4, out_features=1, bias=True)
  (sigmoid): Sigmoid()
)
```

**Training:**

The training process was completed in 18.54 seconds over 11 epochs. During this training, I observed that while the loss continued to decrease with more epochs, the model's performance on unseen test data remained consistent after the 11th epoch. This phenomenon occurs because, with more training, the model becomes overly sensitive to noise and doesn't improve its generalization to the test data any further. As a result, I made the decision to stop training after 11 epochs to maintain the model's optimal performance.



**With Confusion Matrix Evaluation:**

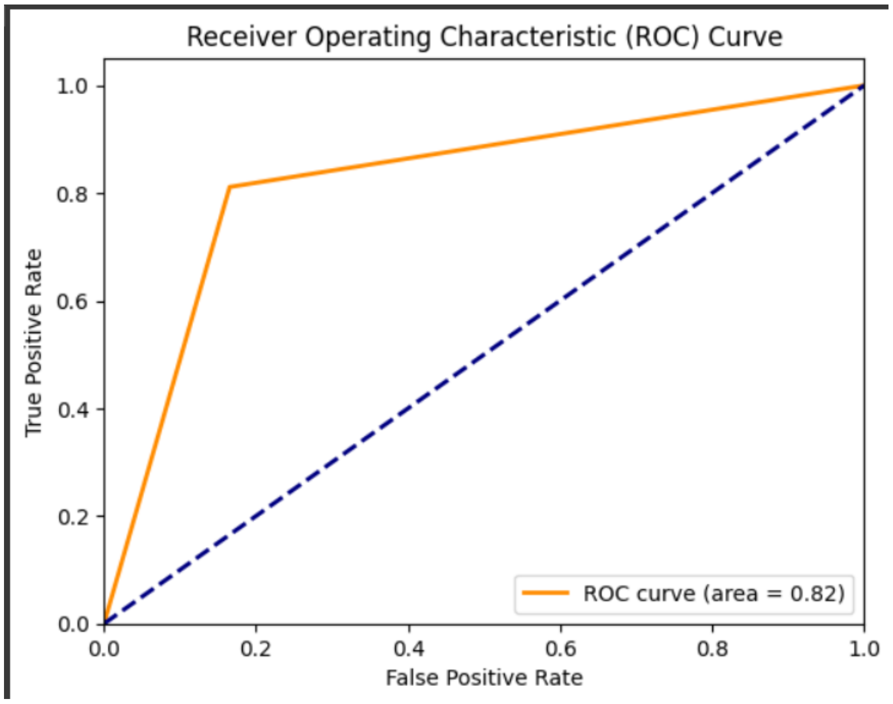

```
              precision    recall  f1-score   support

         0.0       0.82      0.83      0.83      2673
         1.0       0.83      0.81      0.82      2655

    accuracy                           0.82      5328
   macro avg       0.82      0.82      0.82      5328
weighted avg       0.82      0.82      0.82      5328
```

**ROC / AUC Evaluation:**

Notably, the dark orange curve, which stands well above the diagonal line, demonstrates the classifier's excellent ability to differentiate between classes, with an AUC (Area Under the Curve) value of 0.82, indicating strong performance.


Receiver Operating Characteristic (ROC) Curve
True Positive Rate
False Positive Rate
ROC curve (area = 0.82)