# An Academia Recommendation System using Community Detection based on GEAM in Multiplex Networks

A Project Report Submitted

in Partial Fulfilment of the Requirements

for the Degree of

## BACHELOR OF TECHNOLOGY

in

## COMPUTER SCIENCE AND ENGINEERING

*by*

**Neeraj P Yetheendran - 2021BCS0106**

**Korlepara Sai Saranya - 2021BCS0045**

**Baggam Naga Jayadeep - 2021BCS0179**

**Abhay Kumar - 2021BCS0087**


*to*

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**INDIAN INSTITUTE OF INFORMATION TECHNOLOGY**

**KOTTAYAM-686635, INDIA**

# DECLARATION

We, **Neeraj P Yetheendran** (**Roll No: 2021BCS0106**), **Korlepara Sai Saranya** (**Roll No: 2021BCS0045**), **Baggam Naga Jayadeep** (**Roll No: 2021BCS0179**), **Abhay Kumar** (**Roll No: 2021BCS0087**) hereby declare that this report entitled **"An Academia Recommendation System using Community Detection based on GEAM in Multiplex Networks"** submitted to Indian Institute of Information Technology Kottayam towards partial requirement of **Bachelor of Technology** in **Computer Science Engineering** is an original work carried out by us under the supervision of **Dr. Divya Sindhu Lekha** and has not formed the basis for the award of any degree or diploma, in this or any other institution or university. We have sincerely tried to uphold the academic ethics and honesty. Whenever an external information or statement or result is used then, that have been duly acknowledged and cited.

Kottayam-686635 **Neeraj P Yetheendran - Roll No: 2021BCS0106**

November 2024 **Korlepara Sai Saranya - Roll No: 2021BCS0045**

**Baggam Naga Jayadeep - Roll No: 2021BCS0179**

**Abhay Kumar - Roll No: 2021BCS0087**

# CERTIFICATE

This is to certify that the work contained in this project report entitled **"An Academia Recommendation System using Community Detection based on GEAM in Multiplex Networks"** submitted by **Neeraj P Yetheendran** (**Roll No: 2021BCS0106**), **Korlepara Sai Saranya** (**Roll No: 2021BCS0045**), **Baggam Naga Jayadeep** (**Roll No: 2021BCS0179**), **Abhay Kumar** (**Roll No: 2021BCS0087**) to the Indian Institute of Information Technology Kottayam towards partial requirement of **Bachelor of Technology** in **Indian Institute of Information Technology, Kottayam** has been carried out by them under my supervision and that it has not been submitted elsewhere for the award of any degree.

Kottayam-686635
November 2024

(Dr. Divya Sindhu Lekha)
Project Supervisor

# ABSTRACT

The paper introduces an AI-Driven driver drowsiness detection system that aims to utilize two base models — (i) one that utilizes a shallow convolutional neural network (CNN) with novel transfer learning-based features which combines the strengths of the Visual Geometry Group (VGG-16) and Light Gradient-Boosting Machine (LGBM) methods (ii) other that uses a multi-loss convolutional neural network. These models are ensembled through stacked generalization to enhance the drowsiness detection.

The first base model predicts driver drowsiness using eye movement behaviour, and opening and closing of the eye lids. First, a 68-point face landmark identification approach is used to identify faces and extract the eye areas. Then, these are passed through a lightweight CNN model, powered by VGG which generates salient transfer features using LGBM. A K-Neighbour classifier is used to detect the drowsiness using the features extracted.

The second base model utilizes a multi-loss convolutional neural network pre-trained on 300W-LP, a large synthetically expanded dataset, to predict the Euler angles of head — pitch, roll, and yaw directly from the images, through joint binned pose classification and regression. Then, excessive deviation of driver's head posture is detected using an empirical relation among the Euler angles, to detect driver drowsiness.

Ensemble techniques like stacked generalization is utilized to improve overall predictive performance by leveraging the strengths of each model variant.

Overall, the system utilizes a lightweight CNN powered by VGG and LGBM methods to detect drowsiness from facial features, and aims to en-

semble it with a multi-loss CNN to detect drowsiness from head posture, aiming to provide a robust system to detect drowsiness in driver, so as to alert the drivers to lower the number of accidents caused by weariness.

# Contents

# List of Figures

# Chapter 1

# Introduction

## 1.1 Background

In today's fast-paced world, where everyone is constantly juggling around work, social commitments, and personal responsibilities, getting enough sleep has become a luxury. This has taken a significant toll on several people's mental health, so much that the weariness can have extreme consequences when they get behind the wheel. Driving when drowsy significantly impairs reaction times, reduces alertness, and increases the risk of accidents.

As per approximations furnished by the National Highway Traffic Safety Administration (NHTSA), driving while fatigued caused $12.5 billion in economic loss, 71,000 injuries, and 1,550 fatalities. Statistics (NHTSA) [1] show that in 2017, accidents involving drowsy drivers resulted in 50,000 injuries and 795 fatalities. This calls for the need of a system in the vehicle, which can scan and alert the driver when signs of fatigue are recognized.

One of the most accurate ways of detecting fatigue in driver is through

physiological signals like heart rate, Electroencephalography (EEG), Electrocardiography (ECG or EKG), Respiration Rate, Body Temperature, etc. However, these methods are not really feasible, since they require expensive equipment, and a continuous direct contact with driver's body. Hence, we require a system that can detect drowsiness in driver through behavioral signals like eyes movement, opening and closing of mouth, head position, etc.

To address these challenges, We propose a lightweight novel transfer learning-based features generation which combines the strengths of the Visual Geometry Group (VGG-16) and Light Gradient-Boosting Machine (LGBM) methods. Experimental evaluations reveal that the k-neighbors classifier on the extracted features of eye lids outperformed the state-of-the-art approach with a high-performance accuracy of 99%. The computational complexity analysis shows that the proposed approach detects driver drowsiness in 0.00829 seconds [2].

However, predicting drowsiness just through facial features and triggering an alarm would seem some what a too bit far-fetched, and might result in certain False Positives. Hence, we aim to top it off by ensembling another multi-loss CNN to it, which estimates the Euler angles (pitch, roll, and yaw) of the driver's head, and use an empirical relation to determine excessive deflection of head. Through this interdisciplinary approach the system aims to combine the accuracies of individual models, and predict driver's fatigue with a greater accuracy.

## 1.2 Goal

The main objectives of the work are:

- Develop an integrated system comprising of an LGBM and a VGG based CNN model to detect driver drowsiness from facial features, coupled using ensemble techniques with another multi-loss CNN model which predicts extreme deflection in driver's head for enhanced accuracy.
- Make the model as lightweight as possible, to make it robust and efficient so that it can be incorporated in day-to-day lives.
- Incorporate a mechanism for the system to continuously learn and adapt to new data
- Fine tune the ensembled model for greater accuracies.

## 1.3 Motivation

In today's fast-paced world, with everyone is constantly juggling around work, social commitments, and personal responsibilities it becomes absolutely important to have a system to detect and alert a driver when he is drowsy, to prevent accidents. Physiological signals are not really feasible, since they require expensive equipment, and a continuous direct contact with driver's body. By leveraging advanced machine learning techniques like a novel transfer learning-based features generation with VGG and LGBM

in a lightweight convolutional neural network, coupled with ensemble methods, we aim to enhance prediction accuracy and provide accurate signals for drowsiness detection. Making the model lightweight, robust and efficient to consume less energy makes it more usable in real-world scenarios.

## 1.4 Problem Statement

Develop and evaluate an advanced driver drowsiness detection system that integrates multiple detection techniques to enhance its accuracy and robustness. The system should combine various measures, such as physiological, behavioural, and vehicle-based indicators, to identify drowsiness at different stages. This research will explore and compare different technologies and algorithms, including those based on eye activity, to assess their effectiveness in real-world conditions. The aim is to improve early drowsiness detection, minimize false positives and negatives, and provide insights into integrating these systems into broader driver assistance technologies to enhance overall road safety.

# Chapter 2

# Literature Survey

The paper [**?**] presents a new method for fatigued driving using a deep learning model that combines the lightweight nature of YOLOv5s with 3D facial key feature extraction. This study shows that the improved YOLOv5 architecture, equipped with ShuffleNetV2_BD and maxpool cross-scale feature aggregation (M-CFAM), improves the detection accuracy while reducing the computational cost. Vital signs such as Early Response Rate (EAR) and Marginal Response Rate (MAR) are evaluated for important fatigue indicators such as eyes closed and yawning. The paper notes that PERCLOS (percentage of eyes closed) is useful in detecting fatigue in the short term. However, the long-term monitoring or intervention effects of different factors on the working model are also discussed. In addition, this model primarily focuses on immediate detection and does not consider the influence of different lighting conditions or environmental factors that may affect the accuracy in real-world use. The focus is on additional measures and other factors that may affect the reliability of driving fatigue intensification. The paper [**?**]

proposes a hybrid model combining MTCNN for spatial feature extraction and LSTM and TSC-LSTM for temporal sequence analysis (drowsiness detection), enhancing detection accuracy with less noise. The cost computation is compared with original VGG11 architecture to find the fastest and most accurate CNN.

A systematic literature review was conducted to identify studies on eye-activity-based driver drowsiness detection (DDD) systems [**?**]. Relevant studies were selected from four databases, focusing on drowsiness, eye activity, and performance measures. Key data on ocular parameters, technologies, and performance metrics were extracted and analyzed, with a classification of eye activity measures. Future directions for DDD systems were also discussed.

The study [**?**] proposed a VGLG approach that combines VGG-16 for feature extraction and LGBM for transfer feature generation. Four machine learning models and two deep neural networks were validated using k-fold cross-validation and hyperparameter tuning. The approach utilized a standard dataset comprising 4,103 eye images of drivers, meticulously labeled as either open or closed eye movements.

The paper [**?**] suggested a combination of machine learning and deep learning models using HOG features, PCA, and a custom 30-layer CNN architecture. YawDD Video Dataset, with videos of drivers showing normal, talking, and yawning behaviors.

The paper [**?**] proposed a shallow CNN architecture using limited training data for drowsiness detection based on visual cues like eyelid closure.

Head Pose Estimation Through Euler Angles and Deep Learning Ruiz et

al. [?] introduce a convolutional neural network (CNN) model that predicts head pose angles—specifically pitch, roll, and yaw—directly from image intensities. This approach bypasses traditional keypoint-based methods, which rely on aligning 2D-3D face models and can be error-prone when landmarks are obscured or inaccurately detected. In Ruiz's study, the CNN model's multi-loss function simultaneously classifies and regresses each head pose angle, a method shown to significantly improve accuracy. Using a large, synthetically expanded dataset for training, the model achieves state-of-the-art results on benchmarks such as the AFLW2000 and BIWI datasets. However, certain limitations are acknowledged, such as a reduction in performance under extreme head poses and on low-resolution images, which points to an ongoing need for robustness improvements, especially for applications like surveillance and driver monitoring. GRU-Based Temporal Prediction Model for Stock Indices The study by [?] proposes StockNet, a GRU-based (Gated Recurrent Unit) model designed to predict stock indices by integrating injection and investigation modules that reduce overfitting through data augmentation. StockNet utilizes temporal dependencies within financial time series data, showing that GRU architectures can effectively capture sequential relationships. Despite achieving strong predictive accuracy, StockNet's reliance on specific data distributions and the need for further generalization to diverse market conditions remain areas for development. The study illustrates the GRU model's capacity to handle complex, time-sensitive financial forecasting tasks, while also suggesting that continuous model adaptation is essential for dynamic financial environments.

The study in [?] described an empirical relation between different Euler

angles — pitch, roll and yaw and extreme deflection of driver's head. As per the study, when the yaw angle is greater than 20°, the pitch angle is greater than 21°, and the roll angle is greater than 20.5°, the driver's head position could be concluded to be deviated excessively. We can use this in empirical relation after obtaining the Euler's angles of the head from [**?**], and detect driver's drowsiness using head position.

# Chapter 3

# Chapter-3 Methodology

## 3.1 Data Preparation

The dataset used in this study consists of images categorized into four classes: Closed, Open, No Yawn, and Yawn. The images are preprocessed and transformed using `torchvision.transforms` to ensure consistency and improve model performance. Data augmentation techniques, such as resizing, cropping, and normalization, are applied to enhance the dataset's diversity.

## 3.2 WorkFlow

This section describes the overall workflow of the methodology, including data preprocessing, model training, and evaluation steps.

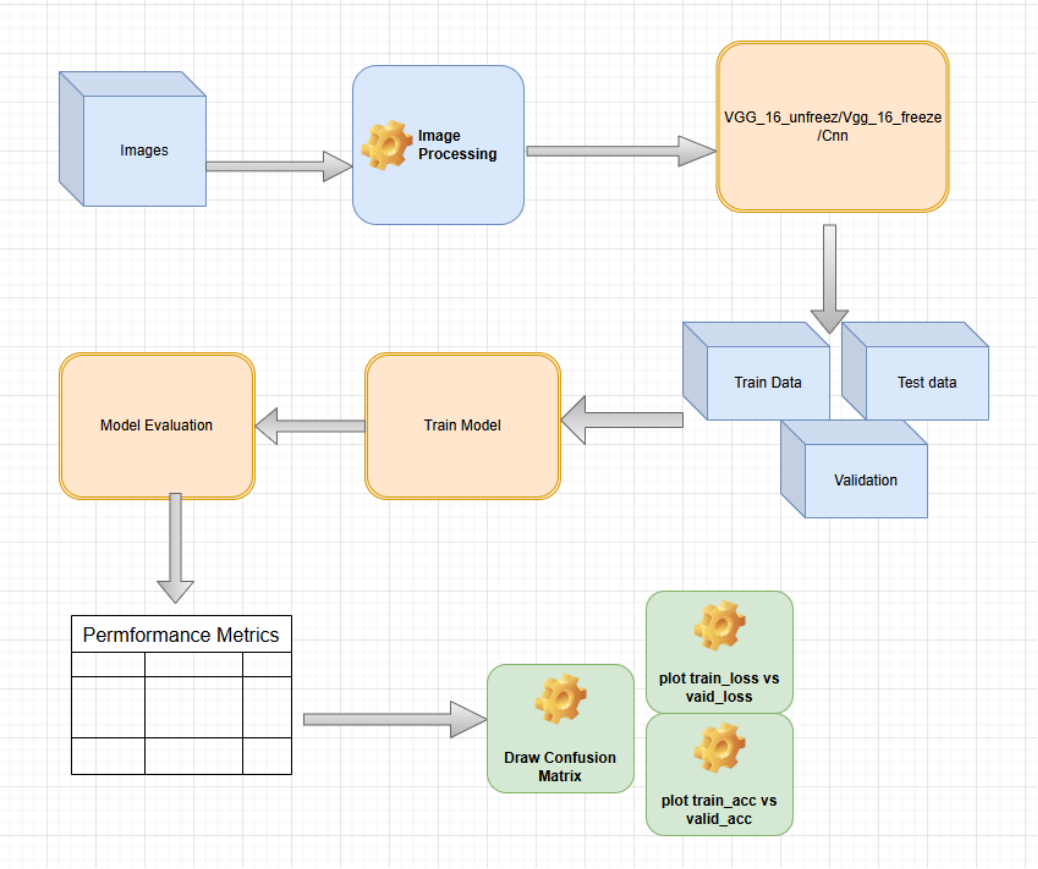


Figure 3.1: Workflow Step

### 3.2.1 Step 1: Data Collection

The first step involves collecting the dataset, which consists of images categorized into four classes: Closed, Open, No Yawn, and Yawn. This dataset is crucial for training and evaluating the models.

### 3.2.2 Step 2: Data Preprocessing

In this step, the collected images are preprocessed to ensure consistency and improve model performance. Preprocessing includes resizing, cropping, and normalization using `torchvision.transforms`. Data augmentation techniques are also applied to enhance the dataset's diversity.

### 3.2.3 Step 3: Model Training

The preprocessed data is then used to train the models. Various model architectures, such as VGG-16 with Batch Normalization, VGG-16 with Unfreezing and Learning Rate Scheduler, and a Lightweight CNN, are employed. Each model is fine-tuned and optimized for the task.

### 3.2.4 Step 4: Model Evaluation

After training, the models are evaluated using validation data. The performance metrics, such as accuracy and loss, are tracked and analyzed to determine the effectiveness of each model.

## 3.3 Model Architectures

### 3.3.1 Model 1: VGG-16 with Batch Normalization

The first model is based on the VGG-16 architecture with batch normalization. The model is pre-trained on ImageNet and fine-tuned for the DDD task. The last three convolutional layers are unfrozen to allow fine-tuning.

```
vgg16 = models.vgg16_bn(weights=models.VGG16_BN_Weights.DEFAULT)  # Use pretrained

# Freeze training for all layers in the feature extractor
for param in vgg16.features.parameters():
    param.requires_grad = False

# Unfreeze the last three convolutional layers
conv_layers_to_unfreeze = []
for i in range(len(vgg16.features) - 1, -1, -1):  # Traverse in reverse
    if isinstance(vgg16.features[i], nn.Conv2d):
        conv_layers_to_unfreeze.append(i)
    if len(conv_layers_to_unfreeze) == 3:
        break

# Unfreeze these layers
for i in conv_layers_to_unfreeze:
    for param in vgg16.features[i].parameters():
        param.requires_grad = True
```

```
# Modify the classifier to adapt to the number of classes
num_features = vgg16.classifier[6].in_features  # Get input features of the last l
features = list(vgg16.classifier.children())[:-1]  # Remove last layer
features.append(nn.Linear(num_features, 4))  # Add a new layer for 4 classes
vgg16.classifier = nn.Sequential(*features)  # Replace the classifier
vgg16 = vgg16.to(device)
```

### 3.3.2 Model 2: VGG-16 with Unfreezing and Learning Rate Scheduler

The second model is similar to the first but includes a learning rate scheduler to adjust the learning rate based on the validation loss.

```
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(vgg16.parameters(), lr=0.0001, weight_decay=0.01)
exp_lr_scheduler = lr_scheduler.ReduceLROnPlateau(optimizer, mode='min', factor=0.
```

### 3.3.3 Model 3: Lightweight CNN

The third model is a lightweight CNN designed from scratch to balance computational efficiency and accuracy.

```
class LightWeightCNN(nn.Module):
    def __init__(self):
        super(LightWeightCNN, self).__init__()
        self.conv1 = nn.Conv2d(in_channels=3, out_channels=32, kernel_size=3, stri
        self.conv2 = nn.Conv2d(in_channels=32, out_channels=64, kernel_size=3, str
```

```
        self.conv3 = nn.Conv2d(in_channels=64, out_channels=128, kernel_size=3, st
        self.conv4 = nn.Conv2d(in_channels=128, out_channels=128, kernel_size=3, s
        self.conv5 = nn.Conv2d(in_channels=128, out_channels=128, kernel_size=3, s
        self.pool = nn.MaxPool2d(kernel_size=2, stride=2)
        self.fc1 = nn.Linear(in_features=128*7*7, out_features=1024)
        self.fc2 = nn.Linear(in_features=1024, out_features=4)
        self.dropout = nn.Dropout2d(0.2)
        self.dropout1 = nn.Dropout(0.2)

    def forward(self, x):
        x = self.dropout(self.pool(F.relu(self.conv1(x))))
        x = self.dropout(self.pool(F.relu(self.conv2(x))))
        x = self.dropout(self.pool(F.relu(self.conv3(x))))
        x = self.pool(F.relu(self.conv4(x)))
        x = self.pool(F.relu(self.conv5(x)))
        x = x.view(-1, 128*7*7)
        x = self.dropout1(x)
        x = F.relu(self.fc1(x))
        x = self.dropout1(x)
        x = self.fc2(x)
        return x
```

## 3.4 Training

The models are trained using the following parameters:

- **Optimizer:** SGD with learning rate 0.01 for the lightweight CNN, and Adam with learning rate 0.0001 for the VGG-16 models.
- **Loss Function:** CrossEntropyLoss
- **Epochs:** 50

Training and validation losses and accuracies are tracked and printed for each epoch. The models are saved if the validation loss decreases. report booktabs caption float graphicx

# Chapter 4

# Results

## 4.1 Evaluation Metrics

The models' performance is evaluated using the following metrics:

- **Accuracy:** Measures the proportion of correctly classified instances out of the total instances.
- **Precision:** Measures the proportion of true positive instances out of the total predicted positive instances.
- **Recall:** Measures the proportion of true positive instances out of the total actual positive instances.
- **F1 Score:** The harmonic mean of precision and recall, providing a single metric that balances both.

The table below summarizes the evaluation metrics for the models:

## 4.2 VGG-16 Fine-tuned

### 4.2.1 Training and Validation Loss

### 4.2.2 Training and Validation Accuracy

## 4.3 VGG16 with all layer convolutional layer frozen

### 4.3.1 Training and Validation Loss

### 4.3.2 Training and Validation Accuracy

## 4.4 Lightweight CNN

### 4.4.1 Training and Validation Loss

### 4.4.2 Training and Validation Accuracy

### 4.4.3 Confusion Matrix

## 4.5 Conclusion

The lightweight CNN model for detecting driver drowsiness and distraction demonstrates satisfactory performance metrics, including accuracy, precision, recall, and F1 score. The model's architecture balances computational efficiency and accuracy, making it suitable for real-time applications in driver monitoring systems. Future work will focus on implementing additional data

Table 4.1: Evaluation Metrics for the Models

| **Model** | **Accuracy** | **Precision** | **Recall** | **F1 Score** |
|---|---|---|---|---|
| VGG-16 Fine-tuned | 0.997691 | 0.997712 | 0.997691 | 0.997691 |
| VGG-16 | 0.993072 | 0.993093 | 0.993072 | 0.993072 |
| CNN | 0.969977 | 0.969997 | 0.969977 | 0.969977 |

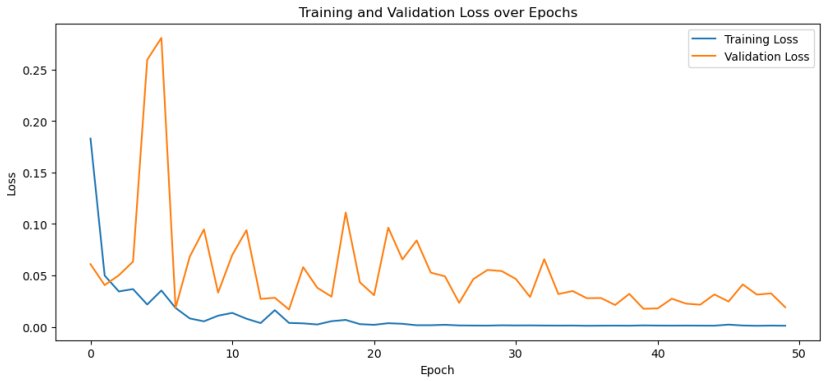


Figure 4.1: Training and Validation Loss for VGG16 fine-tuned (last 3 layer)

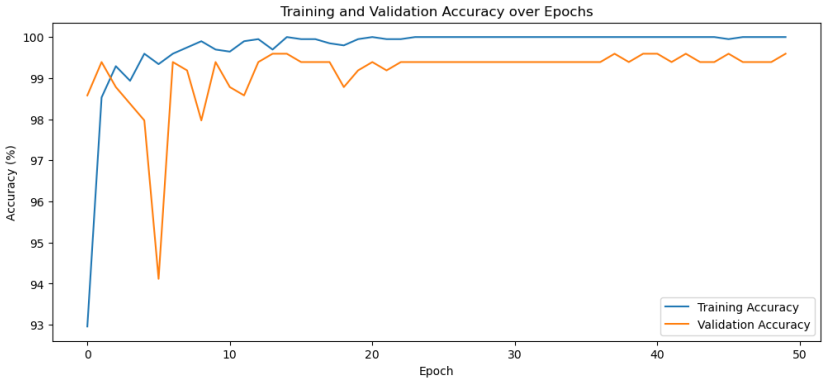


Figure 4.2: Training and Validation Accuracy for VGG16 fine-tuned (last 3 layer)


Figure 4.3: Training and Validation Loss for VGG16 with Learning Rate Scheduler

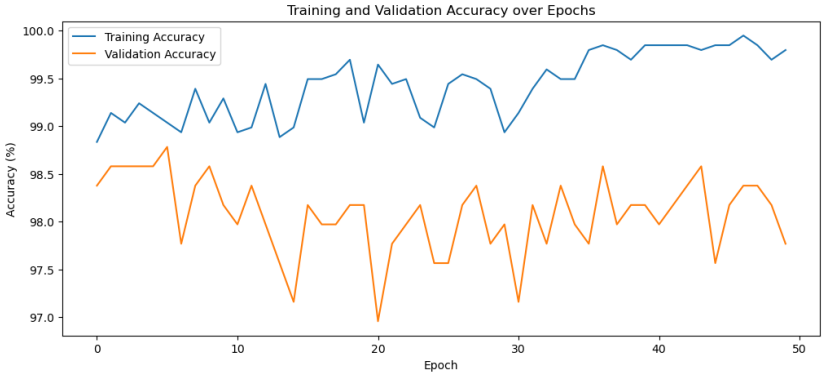


Figure 4.4: Training and Validation Accuracy for VGG16 with Learning Rate Scheduler

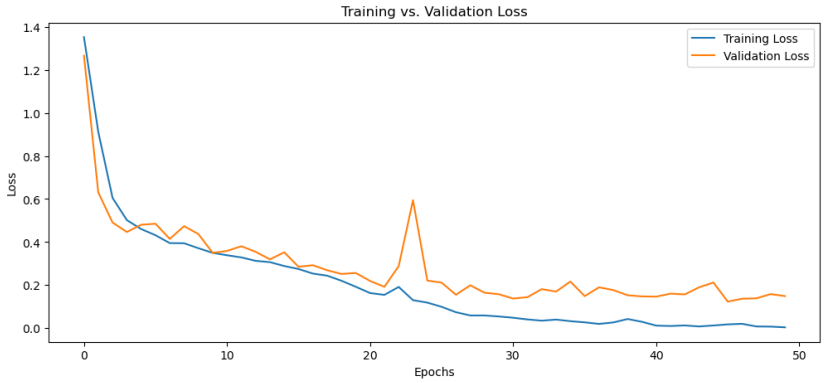


Figure 4.5: Training and Validation Loss for Lightweight CNN

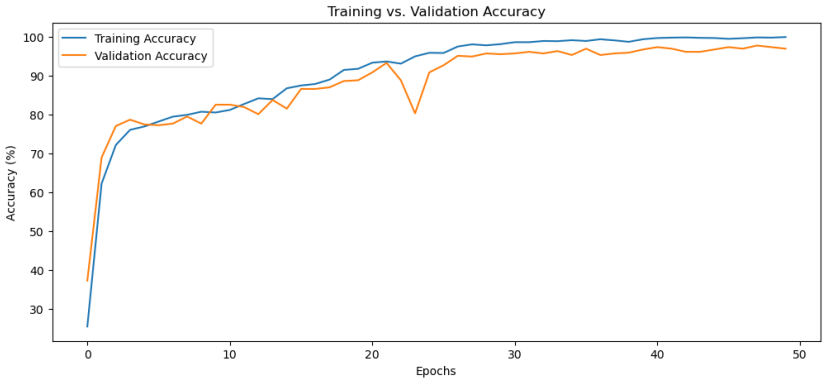


Figure 4.6: Training and Validation Accuracy for Lightweight CNN

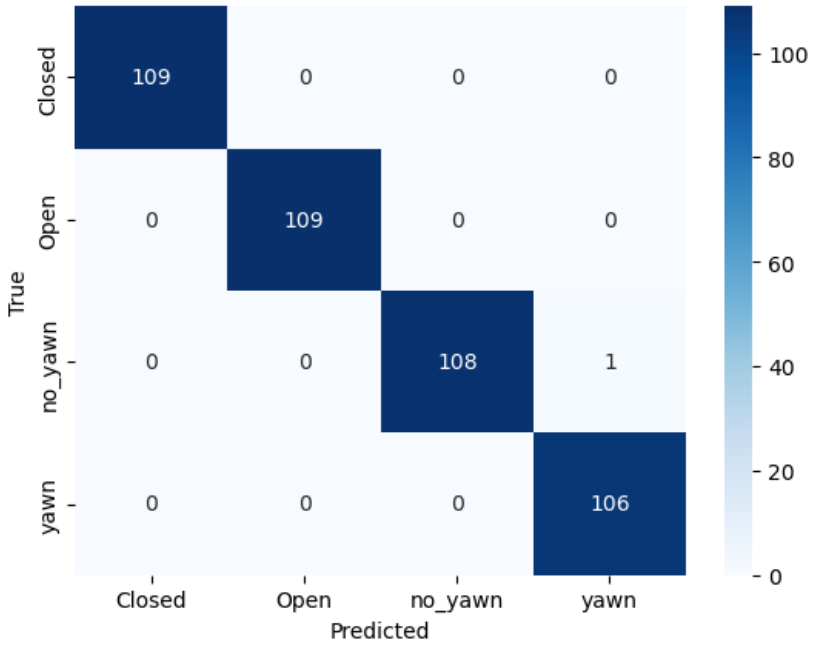


Figure 4.7: Confusion Matrix VGG16 with last 3 Convolutional Layers unfreezed

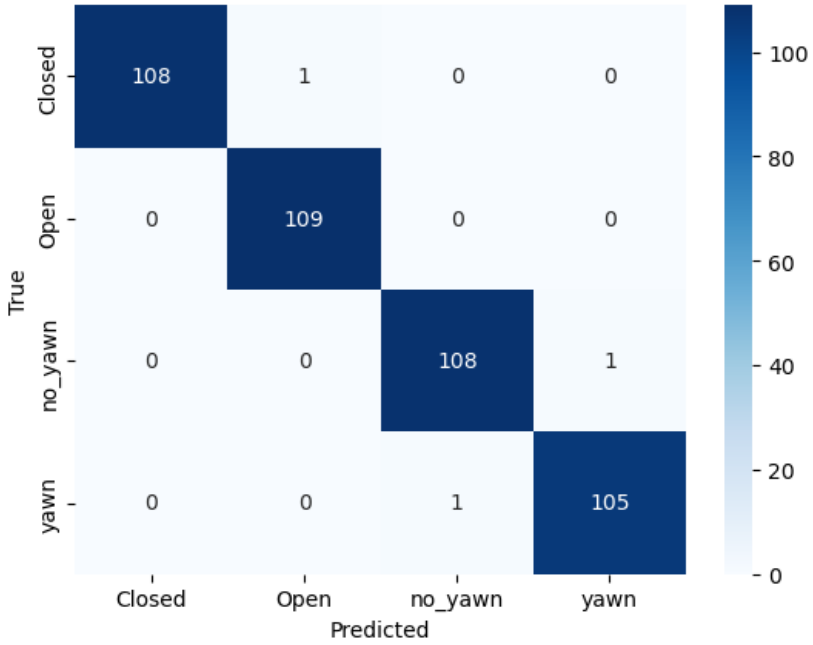


Figure 4.8: Confusion Matrix for VGG16

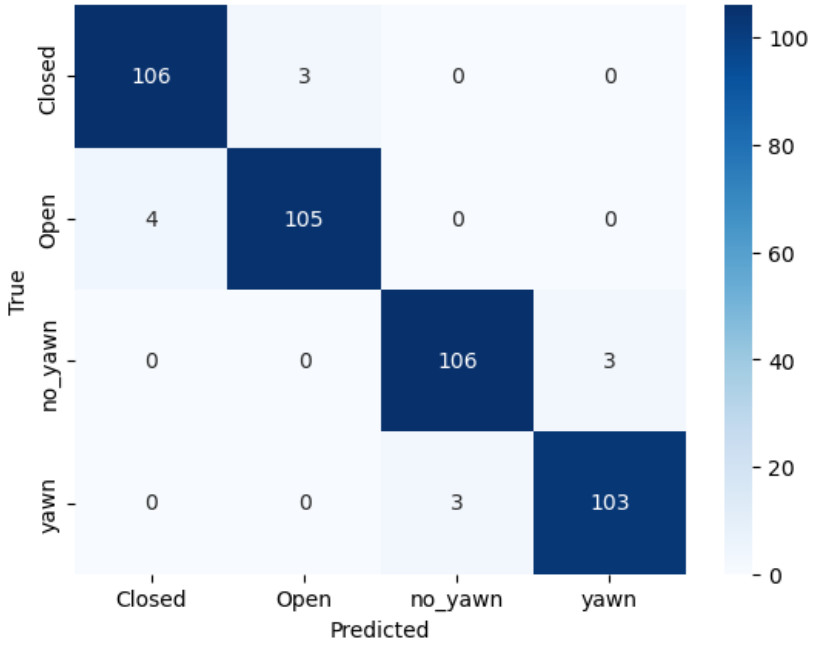


Figure 4.9: Confusion Matrix for Lightweight CNN

augmentation techniques, hyperparameter tuning, and exploring more complex architectures or pre-trained models to further improve performance.

# Bibliography

[1] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. Advances in neural information processing systems, 25.

[2] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[3] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).