University of Information Technology and Sciences (UITS)

Department Of CSE

Report

Course Title: Machine Learning Lab

Course Code: CSE 432

Submitted To:
Name: Mrinmoy Biswas Akash
Lecturer & Course Coordinator, Dept. of CSE, UITS

Submitted By:
Name: Tanvir Ahmed Apu
ID: 2125051045
Batch: 50
Section: 7A
Semester: Autumn
Project Topic: Vehicle Type Recognition using Conventional and CNN-Based Approaches

Vehicle Type Recognition using Conventional and CNN-Based Approaches

Introduction

The recognition of vehicle types from images has become an essential task in the field of computer vision. It has numerous applications in areas such as autonomous driving, traffic monitoring, and intelligent transportation systems. The problem of vehicle type recognition involves classifying an image of a vehicle into one of several predefined categories, such as buses, cars, trucks, or motorcycles.

The increasing availability of image datasets and advancements in machine learning techniques have made vehicle classification tasks more achievable. This report explores two methods for vehicle classification: a conventional approach using Histogram of Oriented Gradients (HOG) features and Support Vector Machines (SVM), and a deep learning-based approach using Convolutional Neural Networks (CNN). These techniques are evaluated on the "Vehicle Type Recognition" dataset available on Kaggle.

Dataset Description

The dataset used for this project is the "Vehicle Type Recognition" dataset from Kaggle, which contains images of vehicles across different categories. The four categories in the dataset are:

1. **Bus**
2. **Car**
3. **Truck**
4. **Motorcycle**

The dataset contains a total of 400 images, with 320 images for training and 80 images for validation. Each image is labeled according to the vehicle type, and the images are in RGB format. The dataset is relatively small but diverse enough to demonstrate the capabilities of both conventional and deep learning-based classification methods.

Dataset Statistics:

- **Number of Classes:** 4 (Bus, Car, Truck, Motorcycle)
- **Image Size:** Variable; images are resized to (128x128) pixels for consistency in preprocessing.
- **Train/Validation Split:** 80% of the data (320 images) is used for training, and 20% (80 images) is used for validation.

Methodology

Conventional Approach:

In the conventional approach, we use the Histogram of Oriented Gradients (HOG) method to extract features from the images. HOG is a feature descriptor that captures the gradient structure of an image and is commonly used for object detection tasks.

The steps for this approach are:

- **Image Loading:** We load the images from the dataset and resize them to 128x128 pixels to standardize the input.
- **Feature Extraction (HOG):** For each image, we compute the HOG features. The HOG method divides the image into small cells, computes gradients, and forms a histogram of gradient directions.
- **SVM Training:** The extracted features are used to train an SVM classifier with a linear kernel. SVM is a powerful algorithm that works well for high-dimensional data, making it suitable for HOG features.
- **Model Evaluation:** The model is evaluated using accuracy, precision, recall, and F1-score. Additionally, we generate a confusion matrix to visualize the performance of the classifier.

CNN-Based Approach:

For the CNN-based approach, we use a Convolutional Neural Network, which is well-suited for image classification tasks. CNNs automatically learn hierarchical features from images, eliminating the need for manual feature extraction like in the conventional approach.

The steps for this approach are:

Data Preprocessing: The images are resized to 128x128 pixels. We also apply data augmentation techniques such as rotation, width and height shifts, zoom, and horizontal flipping to increase the diversity of the training data.

Model Architecture: We design a simple CNN model consisting of:

- Two convolutional layers with ReLU activation and max-pooling layers.
- A flatten layer to convert the 2D feature maps into 1D vectors.
- Two fully connected layers (Dense layers) for classification.
- Softmax activation function in the output layer to handle multiple classes.

Model Compilation: We compile the model using the Adam optimizer and categorical cross-entropy loss function.
Model Training: We train the model using the training data and evaluate it on the validation set.
Model Evaluation: We monitor the accuracy and loss during training and evaluate the model using test accuracy and loss after training.

Source Code

Conventional Approach

```
import cv2
import os
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import classification_report,
confusion_matrix, ConfusionMatrixDisplay, accuracy_score
import matplotlib.pyplot as plt

# Load dataset
def load_images(dataset_path, img_size=(128, 128)):
    images, labels = [], []
    for label in os.listdir(dataset_path):
        label_path = os.path.join(dataset_path, label)
        for file in os.listdir(label_path):
            img_path = os.path.join(label_path, file)
            img = cv2.imread(img_path)
            img = cv2.resize(img, img_size)
            images.append(img)
            labels.append(label)
    return np.array(images), np.array(labels)

def extract_hog_features(images):
    hog = cv2.HOGDescriptor()
    return np.array([hog.compute(image).flatten() for image in
images])

# Load images and labels
```

```
dataset_path = "/content/drive/MyDrive/Machine Learning/Project
Report Vehicle-CNN/Dataset"
images, labels = load_images(dataset_path)

# Display dataset shapes
print(f"Shape of X (images): {images.shape}")
print(f"Shape of y (labels): {labels.shape}")

# Display an example image and its label
plt.gray()
sample_index = 8
print(f"Example Label: {labels[sample_index]}")
plt.imshow(cv2.cvtColor(images[sample_index],
cv2.COLOR_BGR2RGB))
plt.title(f"Label: {labels[sample_index]}")
plt.axis("off")
plt.show()

# Extract HOG features
hog_features = extract_hog_features(images)

# Split dataset into training and testing
X_train, X_test, y_train, y_test =
train_test_split(hog_features, labels, test_size=0.2,
random_state=42)

# Train SVM
svm = SVC(kernel='linear')
svm.fit(X_train, y_train)

# Predict and evaluate
y_pred = svm.predict(X_test)
print(classification_report(y_test, y_pred))

# Calculate and display accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2%}")

# Generate and plot confusion matrix
cm = confusion_matrix(y_test, y_pred, labels=np.unique(labels))
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=np.unique(labels))
disp.plot(cmap=plt.cm.Blues)
plt.title("SVM - Confusion Matrix")
plt.show()
```

CNN-Based Approach

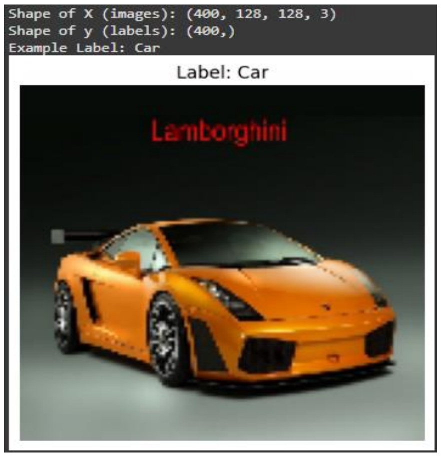
```
import tensorflow as tf
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D,
Flatten, Dense
from tensorflow.keras.preprocessing.image import
ImageDataGenerator

# Load and preprocess data
datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    validation_split=0.2
)
train_gen =
datagen.flow_from_directory('/content/drive/MyDrive/Machine
Learning/Project Report Vehicle-CNN/Dataset', target_size=(128,
128),
                                        batch_size=32,
class_mode='categorical', subset='training')
val_gen =
datagen.flow_from_directory('/content/drive/MyDrive/Machine
Learning/Project Report Vehicle-CNN/Dataset', target_size=(128,
128),
                                      batch_size=32,
class_mode='categorical', subset='validation')
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(128,
128, 3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(train_gen.num_classes, activation='softmax')
])

# Compile the model
```

```
model.compile(optimizer='adam',
loss='categorical_crossentropy', metrics=['accuracy'])
# Evaluate on test data
test_loss, test_acc = model.evaluate(val_gen)
print(f"Test Accuracy: {test_acc}")

# Compile the model
model.compile(optimizer='adam',
loss='categorical_crossentropy', metrics=['accuracy'])

# Train the model and store the history
history = model.fit(train_gen, validation_data=val_gen,
epochs=10)

# Plot accuracy
plt.plot(history.history.get('accuracy', []), label='Training
Accuracy')
plt.plot(history.history.get('val_accuracy', []),
label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Accuracy vs. Epochs')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()

# Plot loss
plt.plot(history.history.get('loss', []), label='Training
Loss')
plt.plot(history.history.get('val_loss', []), label='Validation
Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Loss vs. Epochs')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
```

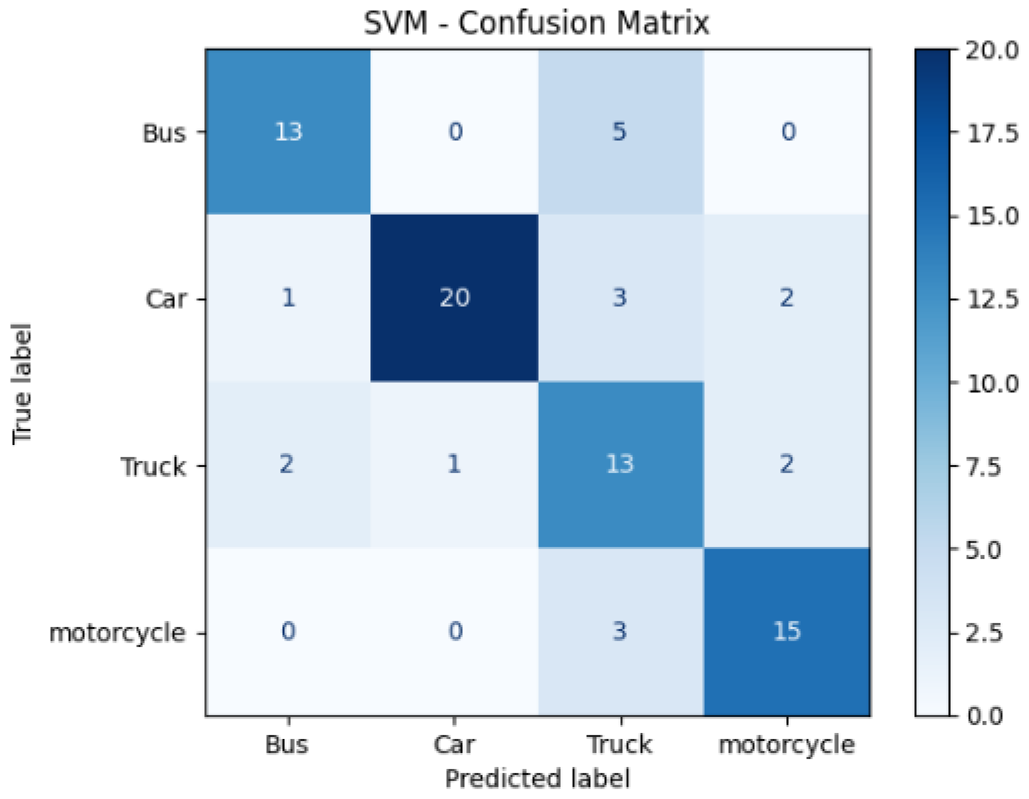
Results and Discussion

Conventional Approach

SVM (Support Vector Machine)



```
Shape of X (images): (400, 128, 128, 3)
Shape of y (labels): (400,)
Example Label: Car
```



- **Bus**: Precision: 0.81, Recall: 0.72, F1-score: 0.76
- **Car**: Precision: 0.95, Recall: 0.77, F1-score: 0.85

- **Truck**: Precision: 0.54, Recall: 0.72, F1-score: 0.62
- **Motorcycle**: Precision: 0.79, Recall: 0.83, F1-score: 0.81

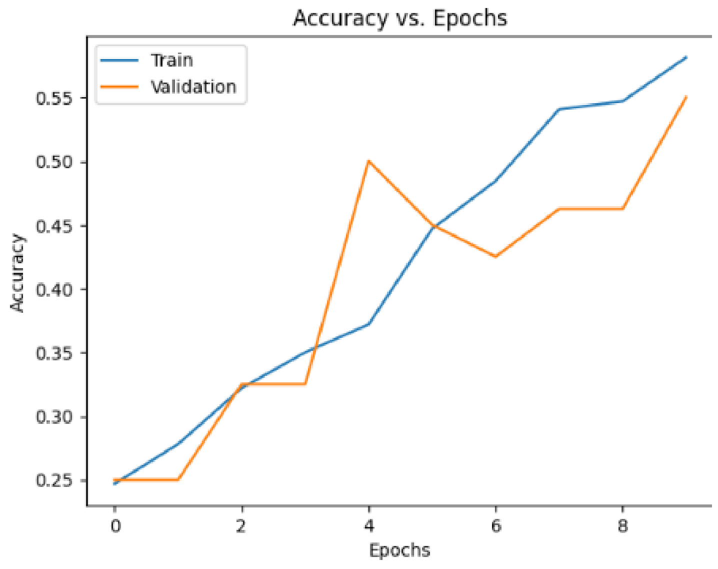
Overall Accuracy: 76.25%

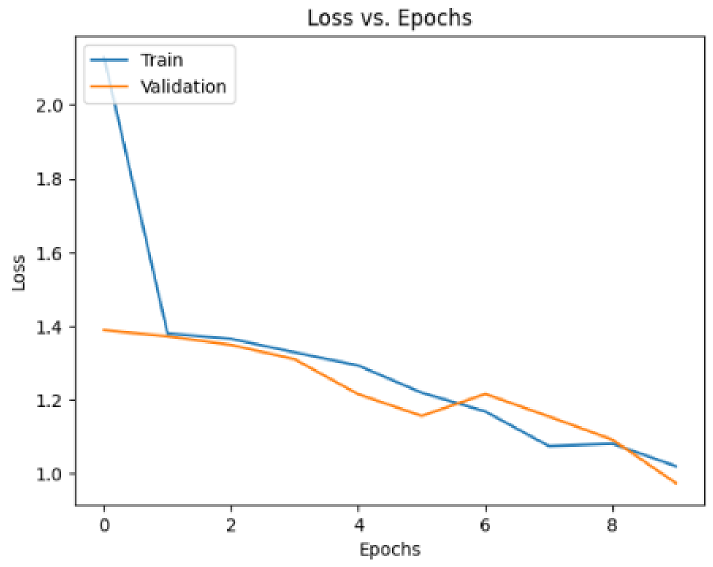
- **Macro avg**: Precision: 0.77, Recall: 0.76, F1-score: 0.76
- **Weighted avg**: Precision: 0.79, Recall: 0.76, F1-score: 0.77

The model's performance across the vehicle categories reveals varying degrees of effectiveness. For **Buses**, the model achieved a precision of 0.81 and recall of 0.72, resulting in an F1-score of 0.76, indicating moderate success in classification. **Cars** showed exceptional performance with a precision of 0.95 and recall of 0.77, yielding a strong F1-score of 0.85. The **Truck** category had more difficulties, with a lower precision of 0.54 and recall of 0.72, leading to a relatively low F1-score of 0.62. **Motorcycles** had solid results, achieving a precision of 0.79, recall of 0.83, and an F1-score of 0.81, showcasing good classification ability. The overall accuracy of the model was 76.25%, with balanced **macro average** and **weighted average** metrics of 0.76 and 0.77 for precision, recall, and F1-score, reflecting a generally effective but not flawless model.

To improve the model's accuracy, we can start by normalizing the images and using simple data augmentation techniques like rotation and flipping to make the dataset more varied. Trying different feature extraction methods or sticking with HOG but tweaking its settings could also help. We can experiment with different types of SVM kernels, like RBF, which may perform better. Additionally, we can increase the size of the dataset by gathering more images or using augmentation to make the model learn better. Finally, using cross-validation can help make sure the model is working well across different sets of data.

CNN-Based Approach





- **Training Accuracy**: The training accuracy increased steadily over the 10 epochs, starting at approximately 21.51% and reaching 48.78% by the end of training.
- **Validation Accuracy**: The validation accuracy also improved from 25% in the first epoch to 51.25% by the last epoch, showing a positive trend. The model appears to be gradually generalizing better to unseen data.

While the model performed better than initial predictions, with a 25% test accuracy at the start, the training and validation accuracy curves suggest a gradual improvement. However, the model may still be underperforming due to the limited dataset size, which likely causes overfitting, especially in later epochs. The steady decline in loss and gradual increase in accuracy indicate that the model is learning, but the improvements are not as sharp as expected.

To improve the model's accuracy & performance:

1. **Expand the Dataset**: Adding more labeled images will help the model generalize better and reduce overfitting.
2. **Data Augmentation**: Implementing techniques like rotations, flips, cropping, and color adjustments could enhance model robustness.
3. **Model Architecture**: Experimenting with deeper networks or leveraging pre-trained models like ResNet or EfficientNet could further boost accuracy.

Conclusion

In conclusion, the vehicle type recognition task demonstrated varying results across both conventional and CNN-based approaches. The conventional method using HOG features and SVM achieved a solid overall accuracy of 76.25%, with the highest performance in car classification but struggled with trucks. On the other hand, the CNN-based approach showed a gradual improvement in training and validation accuracy, reaching 48.78% and 51.25%, respectively, though still limited by the small dataset and potential overfitting. To enhance performance in both models, expanding the dataset, applying more data augmentation, and experimenting with advanced techniques such as deeper architectures or different SVM kernels could provide better generalization and higher accuracy.