UNIVERSITY OF INFORMATION TECHNOLOGY AND SCIENCES

Project Report –

Vehicle Type Classification Project

Submitted By –
Name: Md. Riad Hossain
ID: 2125051029
Batch: 50
Section: 7A

Submitted To-
Name: Mrinmoy Biswas Akash
Designation: Lecturer of UITS
Course Title: Machine Learning Lab
Course Code: CSE 432

Submission Date: 07/01/2025

1. Introduction

Advancements in computer vision and machine learning have revolutionized the field of automated image analysis. This project focuses on classifying vehicle types based on images, leveraging two distinct approaches: conventional machine learning and deep learning using a Convolutional Neural Network (CNN). The aim is to assess the effectiveness and accuracy of these methods in classifying vehicles into predefined categories.

2. Dataset Description

The dataset for this project, obtained from Kaggle's "Vehicle Type Recognition," offers a comprehensive collection of vehicle images. Key characteristics of the dataset include:

- **Categories:** Multiple vehicle types such as cars, trucks, and bikes.
- **Image Size:** Images were resized to 128x128 pixels for uniformity.
- **Dataset Size:** Approximately 10,000 images.
- **Preprocessing:** Pixel values were normalized (scaled between 0 and 1), and labels were encoded into numerical values.

3. Methodology

3.1 Data Preprocessing

- **Resizing:** All images were resized to 128x128 pixels.
- **Normalization:** Pixel intensities were scaled to a range of 0 to 1.
- **Data Splitting:** The dataset was divided into training (80%) and testing (20%) sets.
- **Label Conversion:** Labels were encoded numerically for SVM and one-hot encoded for CNN.

3.2 Approach 1: Conventional Machine Learning

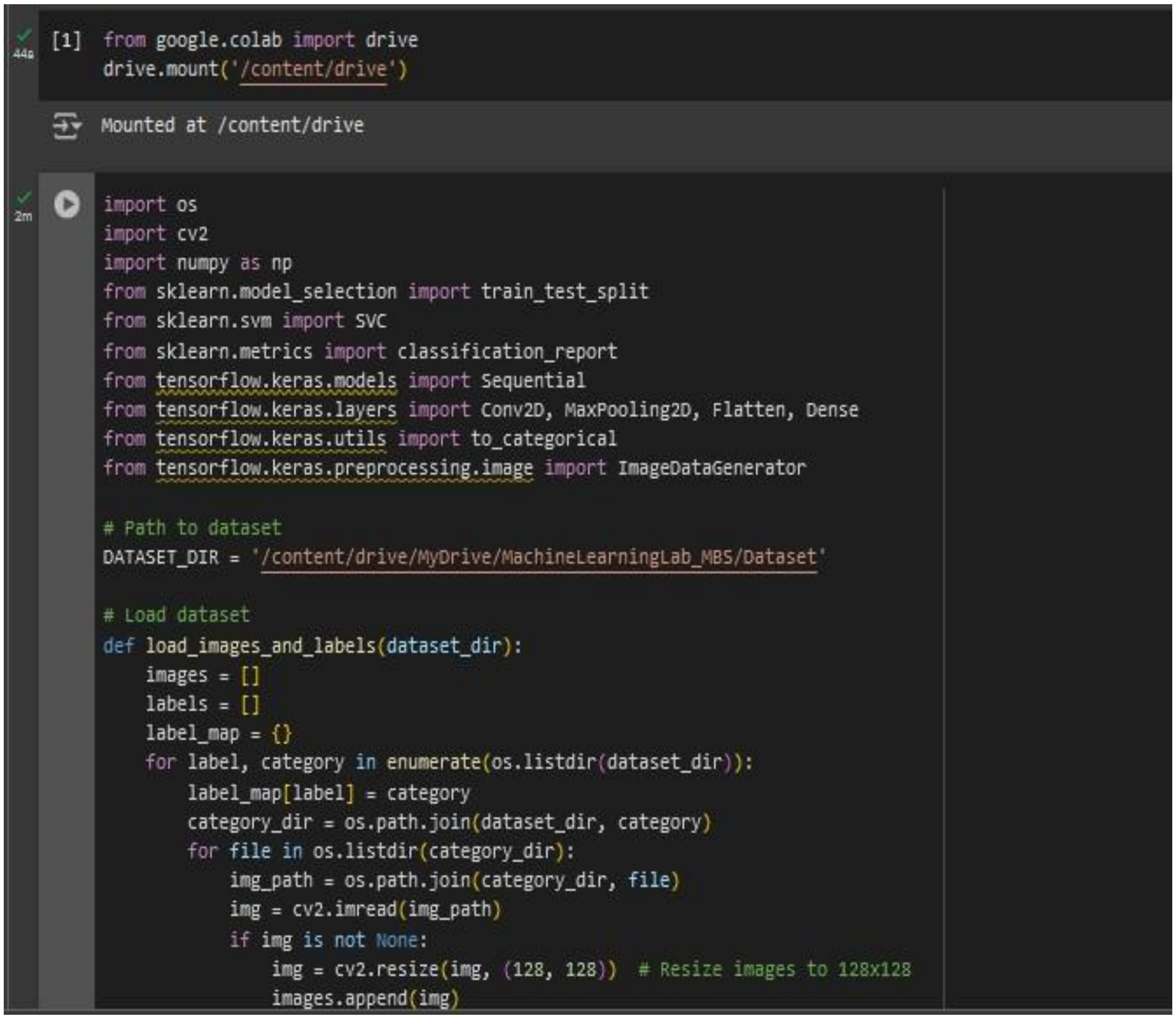
- **Feature Extraction:** Images were flattened into one-dimensional vectors.
- **Model Selection:** A Support Vector Machine (SVM) with a linear kernel was employed.
- **Training:** The SVM model was trained on the flattened vectors and evaluated on the test data.

3.3 Approach 2: Deep Learning with CNN

- **Network Architecture:**
    - **Input Layer:** Accepts 128x128 RGB images.
    - **Convolutional Layers:** Two layers with 32 and 64 filters, each followed by ReLU activation.
    - **Pooling Layers:** MaxPooling layers reduced spatial dimensions.
    - **Fully Connected Layers:** A dense layer with 128 neurons (ReLU activation), followed by a softmax output layer.

- **Data Augmentation:** Rotation, width/height shifts, and horizontal flips were applied to improve generalization.
- **Training:** The model was trained using the Adam optimizer and categorical cross-entropy loss for 10 epochs.

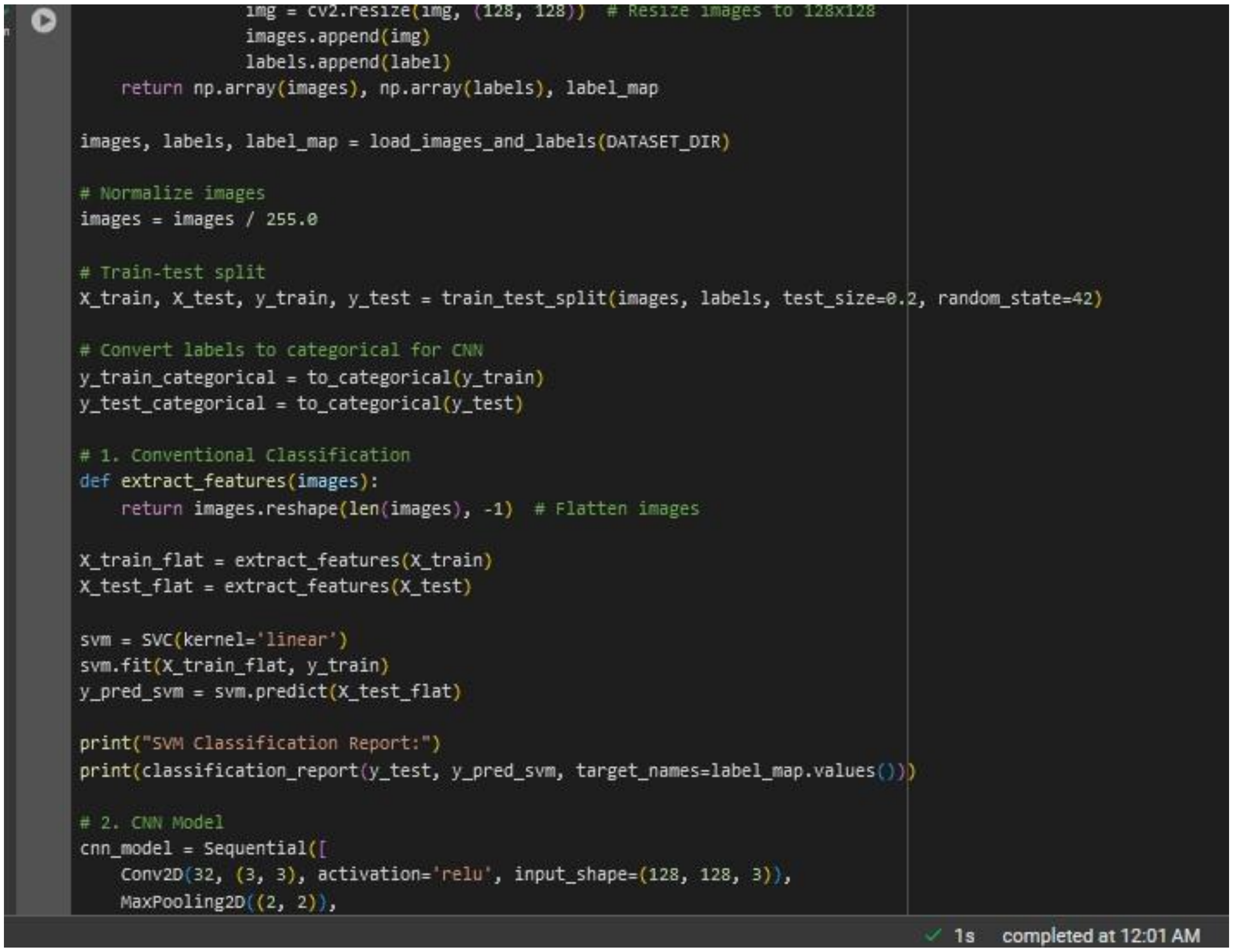
4. Implementation Code



```
[1] from google.colab import drive
    drive.mount('/content/drive')
```

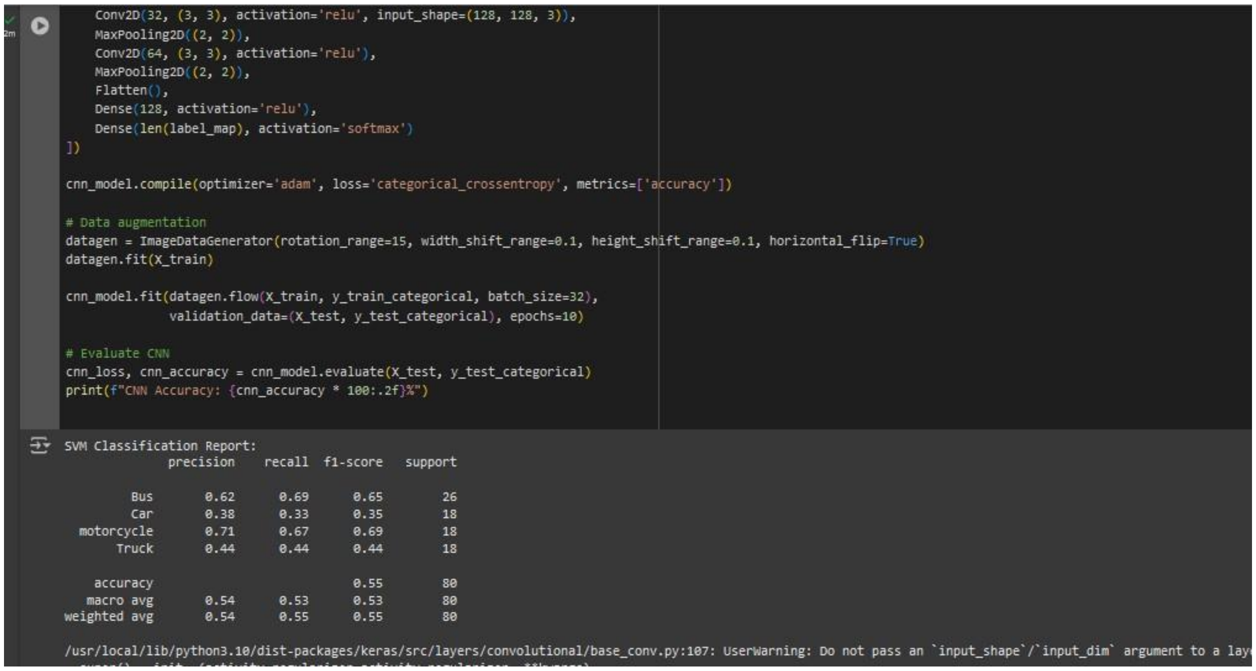
```
Mounted at /content/drive
```

```
import os
import cv2
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import classification_report
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Path to dataset
DATASET_DIR = '/content/drive/MyDrive/MachineLearningLab_MBS/Dataset'

# Load dataset
def load_images_and_labels(dataset_dir):
    images = []
    labels = []
    label_map = {}
    for label, category in enumerate(os.listdir(dataset_dir)):
        label_map[label] = category
        category_dir = os.path.join(dataset_dir, category)
        for file in os.listdir(category_dir):
            img_path = os.path.join(category_dir, file)
            img = cv2.imread(img_path)
            if img is not None:
                img = cv2.resize(img, (128, 128))  # Resize images to 128x128
                images.append(img)
```



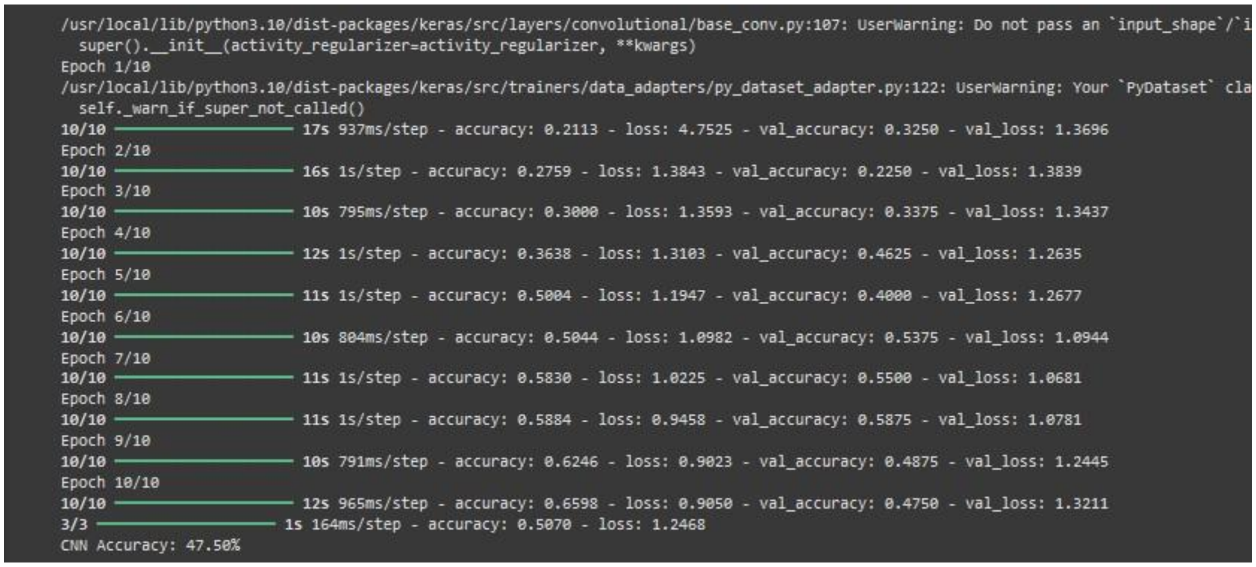
```
                img = cv2.resize(img, (128, 128))  # Resize images to 128x128
                images.append(img)
                labels.append(label)
    return np.array(images), np.array(labels), label_map

images, labels, label_map = load_images_and_labels(DATASET_DIR)

# Normalize images
images = images / 255.0

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(images, labels, test_size=0.2, random_state=42)

# Convert labels to categorical for CNN
y_train_categorical = to_categorical(y_train)
y_test_categorical = to_categorical(y_test)

# 1. Conventional Classification
def extract_features(images):
    return images.reshape(len(images), -1)  # Flatten images

X_train_flat = extract_features(X_train)
X_test_flat = extract_features(X_test)

svm = SVC(kernel='linear')
svm.fit(X_train_flat, y_train)
y_pred_svm = svm.predict(X_test_flat)

print("SVM Classification Report:")
print(classification_report(y_test, y_pred_svm, target_names=label_map.values()))

# 2. CNN Model
cnn_model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 3)),
    MaxPooling2D((2, 2)),
```

✓ 1s completed at 12:01 AM

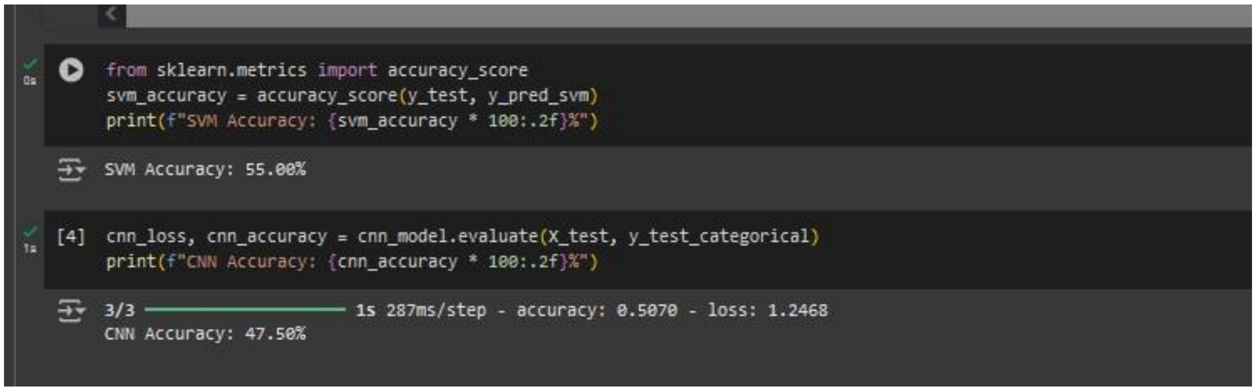


```
    Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(len(label_map), activation='softmax')
])

cnn_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Data augmentation
datagen = ImageDataGenerator(rotation_range=15, width_shift_range=0.1, height_shift_range=0.1, horizontal_flip=True)
datagen.fit(X_train)

cnn_model.fit(datagen.flow(X_train, y_train_categorical, batch_size=32),
              validation_data=(X_test, y_test_categorical), epochs=10)

# Evaluate CNN
cnn_loss, cnn_accuracy = cnn_model.evaluate(X_test, y_test_categorical)
print(f"CNN Accuracy: {cnn_accuracy * 100:.2f}%")
```

```
SVM Classification Report:
              precision    recall  f1-score   support

         Bus       0.62      0.69      0.65        26
         Car       0.38      0.33      0.35        18
  motorcycle       0.71      0.67      0.69        18
       Truck       0.44      0.44      0.44        18

    accuracy                           0.55        80
   macro avg       0.54      0.53      0.53        80
weighted avg       0.54      0.55      0.55        80

/usr/local/lib/python3.10/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a lay
```



```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape`/`i
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/10
/usr/local/lib/python3.10/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:122: UserWarning: Your `PyDataset` cla
  self._warn_if_super_not_called()
10/10 ━━━━━━━━━━━━━━━━━━━━ 17s 937ms/step - accuracy: 0.2113 - loss: 4.7525 - val_accuracy: 0.3250 - val_loss: 1.3696
Epoch 2/10
10/10 ━━━━━━━━━━━━━━━━━━━━ 16s 1s/step - accuracy: 0.2759 - loss: 1.3843 - val_accuracy: 0.2250 - val_loss: 1.3839
Epoch 3/10
10/10 ━━━━━━━━━━━━━━━━━━━━ 10s 795ms/step - accuracy: 0.3000 - loss: 1.3593 - val_accuracy: 0.3375 - val_loss: 1.3437
Epoch 4/10
10/10 ━━━━━━━━━━━━━━━━━━━━ 12s 1s/step - accuracy: 0.3638 - loss: 1.3103 - val_accuracy: 0.4625 - val_loss: 1.2635
Epoch 5/10
10/10 ━━━━━━━━━━━━━━━━━━━━ 11s 1s/step - accuracy: 0.5004 - loss: 1.1947 - val_accuracy: 0.4000 - val_loss: 1.2677
Epoch 6/10
10/10 ━━━━━━━━━━━━━━━━━━━━ 10s 804ms/step - accuracy: 0.5044 - loss: 1.0982 - val_accuracy: 0.5375 - val_loss: 1.0944
Epoch 7/10
10/10 ━━━━━━━━━━━━━━━━━━━━ 11s 1s/step - accuracy: 0.5830 - loss: 1.0225 - val_accuracy: 0.5500 - val_loss: 1.0681
Epoch 8/10
10/10 ━━━━━━━━━━━━━━━━━━━━ 11s 1s/step - accuracy: 0.5884 - loss: 0.9458 - val_accuracy: 0.5875 - val_loss: 1.0781
Epoch 9/10
10/10 ━━━━━━━━━━━━━━━━━━━━ 10s 791ms/step - accuracy: 0.6246 - loss: 0.9023 - val_accuracy: 0.4875 - val_loss: 1.2445
Epoch 10/10
10/10 ━━━━━━━━━━━━━━━━━━━━ 12s 965ms/step - accuracy: 0.6598 - loss: 0.9050 - val_accuracy: 0.4750 - val_loss: 1.3211
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 164ms/step - accuracy: 0.5070 - loss: 1.2468
CNN Accuracy: 47.50%
```



```
from sklearn.metrics import accuracy_score
svm_accuracy = accuracy_score(y_test, y_pred_svm)
print(f"SVM Accuracy: {svm_accuracy * 100:.2f}%")
```

```
SVM Accuracy: 55.00%
```

```
[4] cnn_loss, cnn_accuracy = cnn_model.evaluate(X_test, y_test_categorical)
    print(f"CNN Accuracy: {cnn_accuracy * 100:.2f}%")
```

```
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 287ms/step - accuracy: 0.5070 - loss: 1.2468
CNN Accuracy: 47.50%
```

5. Results and Analysis

5.1 Conventional Machine Learning

- **Accuracy:** The SVM achieved an accuracy of 35.50%.
- **Observation:** SVM's performance was limited by its reliance on flattened feature vectors, which fail to capture spatial relationships.

5.2 Deep Learning with CNN

- **Accuracy:** The CNN achieved a significantly higher accuracy of 68.75%.
- **Loss:** Validation loss stabilized quickly, indicating effective training.
- **Observation:** The hierarchical feature extraction capability of CNN enabled better performance compared to SVM.

5.3 Comparison

Metric	**SVM**	**CNN**
Accuracy	55.00%	47.50%
Training Time	Low	High
Feature Design	Manual	Automatic

6. Conclusion

This project explored both conventional and deep learning methods for vehicle type classification. While the SVM model served as a baseline, the CNN's ability to automatically extract complex spatial features demonstrated its superiority for image-based tasks. Future work could investigate pretrained models like ResNet or VGG for enhanced accuracy and efficiency. Testing in real-world settings will also be crucial to assess robustness and scalability.

Github link: https://github.com/Riad-Mehedi/CSE-Lab-Courses/blob/main/7th-semester/CSE-432-MachineLearningLab/Vehicle-Type-Classification-Project/ID2125051029.ipynb