UITS

Future will be better than thy past

An initiative of PHP Family

University of Information Technology & Sciences

Lab Report

Course Title: Machine Learning

Lab Course Code: CSE432

Project Title: Vehicle Type Classification Project

Submitted To:

Mrinmoy Biswas Akash

Lecturer, Department of CSE, UITS

Submitted by:

Md. Abdullah Al Mamun

ID: 2125051030

Section: 7A

Batch: 50

Vehicle Type Classification Project Report

i. Introduction

The advancement of computer vision and machine learning technologies has enabled the automation of complex visual tasks like vehicle type classification. This project aims to classify vehicles from images into predefined categories, leveraging two distinct approaches: traditional machine learning using Support Vector Machines (SVM) and deep learning through Convolutional Neural Networks (CNNs). The objective is to evaluate the performance of these methods in terms of accuracy, scalability, and efficiency.

ii. Dataset Description

The dataset for this project was sourced from the Kaggle repository, "Vehicle Type Recognition." It consists of diverse vehicle images, organized by type, offering a rich dataset for training and testing classification models.

- **Classes**: The dataset contains images categorized into various vehicle types, such as cars, trucks, and bikes.
- **Image Resolution**: Original image dimensions vary, but all were resized to 128x128 pixels.
- **Size**: The dataset includes approximately 400 images.
- **Preprocessing**:
  - Normalization: Pixel values were scaled to fall between 0 and 1.
  - Label Conversion: Labels were encoded into numerical values for machine learning and one-hot encoded for deep learning models.

iii. Methodology

Data Preprocessing:

1. Resized all images to 128x128 pixels to ensure uniform input dimensions.
2. Normalized pixel intensity values to improve model convergence.
3. Divided the dataset into an 80:20 training-to-testing ratio.
4. Converted labels into numerical or categorical representations for compatibility with respective models.

Traditional Machine Learning (SVM)

- **Feature Extraction**: Flattened images into one-dimensional arrays to serve as input features.
- **Model**: Implemented an SVM classifier with a linear kernel.
- **Training**: The SVM model was trained on flattened feature vectors, optimizing for classification boundaries.

Deep Learning (CNN)

- **CNN Architecture**:
  - **Input Layer**: Accepts resized RGB images of shape 128x128x3.
  - **Convolutional Layers**: Applied 32 and 64 filters in two layers, each followed by ReLU activation.
  - **Pooling Layers**: Used MaxPooling to downsample feature maps and reduce spatial dimensions.
  - **Fully Connected Layers**: Integrated dense layers with 128 neurons and ReLU activation, culminating in a softmax output layer for multi-class classification.
- **Augmentation**: Employed techniques like rotation, shift, and flipping to increase training data diversity.
- **Optimization**: Used Adam optimizer with categorical cross-entropy loss for training over 10 epochs.

iv. 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/7A/Machine Learning Lab/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)
                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
```

```python
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)),
    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

  motorcycle       0.62      0.69      0.65        26
         Car       0.45      0.50      0.47        18
         Bus       0.53      0.44      0.48        18
       Truck       0.50      0.44      0.47        18

    accuracy                           0.54        80
   macro avg       0.53      0.52      0.52        80
weighted avg       0.54      0.54      0.53        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 layer. Wher
  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` class should call `super().__init__(**
  self._warn_if_super_not_called()
10/10 ━━━━━━━━━━━━━━━━━━━━ 13s 829ms/step - accuracy: 0.3256 - loss: 3.1002 - val_accuracy: 0.2250 - val_loss: 1.3960
Epoch 2/10
10/10 ━━━━━━━━━━━━━━━━━━━━ 20s 983ms/step - accuracy: 0.2875 - loss: 1.3769 - val_accuracy: 0.3625 - val_loss: 1.3249
Epoch 3/10
10/10 ━━━━━━━━━━━━━━━━━━━━ 10s 778ms/step - accuracy: 0.3072 - loss: 1.3439 - val_accuracy: 0.4375 - val_loss: 1.1971
Epoch 4/10
10/10 ━━━━━━━━━━━━━━━━━━━━ 11s 1s/step - accuracy: 0.4395 - loss: 1.2019 - val_accuracy: 0.5500 - val_loss: 1.0491
Epoch 5/10
10/10 ━━━━━━━━━━━━━━━━━━━━ 12s 1s/step - accuracy: 0.5530 - loss: 1.0706 - val_accuracy: 0.6125 - val_loss: 0.8640
Epoch 6/10
10/10 ━━━━━━━━━━━━━━━━━━━━ 20s 1s/step - accuracy: 0.5756 - loss: 1.0185 - val_accuracy: 0.6250 - val_loss: 0.9002
Epoch 7/10
10/10 ━━━━━━━━━━━━━━━━━━━━ 19s 766ms/step - accuracy: 0.6016 - loss: 0.9400 - val_accuracy: 0.6375 - val_loss: 0.9038
Epoch 8/10
10/10 ━━━━━━━━━━━━━━━━━━━━ 12s 980ms/step - accuracy: 0.5715 - loss: 0.9543 - val_accuracy: 0.7250 - val_loss: 0.7114
Epoch 9/10
10/10 ━━━━━━━━━━━━━━━━━━━━ 18s 765ms/step - accuracy: 0.6678 - loss: 0.8094 - val_accuracy: 0.7250 - val_loss: 0.7023
Epoch 10/10
10/10 ━━━━━━━━━━━━━━━━━━━━ 13s 908ms/step - accuracy: 0.7180 - loss: 0.7691 - val_accuracy: 0.7250 - val_loss: 0.7560
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 139ms/step - accuracy: 0.7297 - loss: 0.7643
CNN Accuracy: 72.50%
```

```python
[3] 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: 53.75%
```

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

```
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 145ms/step - accuracy: 0.7297 - loss: 0.7643
CNN Accuracy: 72.50%
```

v. Results and Discussion

SVM Results:

- Accuracy: Achieved an accuracy of approximately **53.75%**.
- Analysis: The SVM model's performance was hindered by the flattened representation of image data, which fails to capture spatial relationships critical for classification tasks.

CNN Results:

- Accuracy: Achieved an accuracy of approximately **72.50%**.
- Key Insights:
    - Validation loss stabilized within 10 epochs, reflecting effective training.
    - Hierarchical feature extraction by CNN layers enabled superior classification performance.

Comparison Table:

Metric	SVM	CNN
Accuracy	53.75%	72.50%
Training Time	Low	High
Feature Design	Manual	Automatic

v. Conclusion

This project successfully demonstrated the strengths and limitations of conventional and deep learning techniques in vehicle type classification. While the SVM provided baseline performance, the CNN significantly outperformed it due to its ability to learn complex spatial features from images.

Future enhancements could include employing transfer learning with pretrained models like ResNet or VGG for improved accuracy and efficiency. Additionally, deploying the model in real-world scenarios can help evaluate its reliability and scalability.