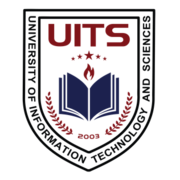
****

**Machine Learning Lab Project report**

**Date**: 07/01/2024

**Cours code :** CSE 432

**Course title:** Machine Learning Lab

**Submitted By:**

**Name:** Tajbir Ahmed Rimon

**ID:** 2125051007

**Batch:** 50

**Section:** 7A1

**Applying Image Classification Techniques for Vehicle Image Classification**

**Introduction**

Vehicle classification is a key component in smart transportation systems, including autonomous vehicles and intelligent traffic management. This project explores two methodologies for image classification:

1. A conventional approach using handcrafted feature extraction techniques.
2. A CNN-based approach that automates feature extraction and classification.

This report highlights the technical details, implementation steps, and performance metrics of both techniques.

**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: Bus, Car, Truck, 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.

**Methodologies**

**Conventional Approach**

The conventional approach involves the following steps:

1. **Feature Extraction**: Histogram of Oriented Gradients (HOG) was used to extract edge and texture features from grayscale images.
2. **Classifier**: A Support Vector Machine (SVM) with a linear kernel was trained on the extracted features.

**Code Implementation:**

|  |
| --- |
| 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, accuracy\_score  from skimage.feature import hog  # Path to dataset  data\_dir = '/content/drive/MyDrive/machine learning/Machin Learning project/Dataset'  # Function to load dataset  def load\_dataset(data\_dir):      labels = []      images = []      class\_names = os.listdir(data\_dir)      for label, class\_name in enumerate(class\_names):          class\_dir = os.path.join(data\_dir, class\_name)          for file in os.listdir(class\_dir):              img\_path = os.path.join(class\_dir, file)              # Ensure only image files are processed              if file.lower().endswith(('.png', '.jpg', '.jpeg')):                  img = cv2.imread(img\_path, cv2.IMREAD\_COLOR)                  if img is not None:  # Ensure valid images                      img = cv2.resize(img, (64, 64))                      images.append(img)                      labels.append(label)      return np.array(images), np.array(labels), class\_names  # Function for feature extraction using HOG  def extract\_features(images):      features = []      for img in images:          gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  # Convert to grayscale          hog\_features = hog(gray,                             orientations=9,                             pixels\_per\_cell=(8, 8),                             cells\_per\_block=(2, 2),                             block\_norm='L2-Hys')          features.append(hog\_features)      return np.array(features)    # Load the dataset  images, labels, class\_names = load\_dataset(data\_dir)  print(f"Loaded {len(images)} images across {len(class\_names)} classes: {class\_names}")  # Extract HOG features  features = extract\_features(images)  # Split dataset into training and testing sets  X\_train, X\_test, y\_train, y\_test = train\_test\_split(features, labels, test\_size=0.2, random\_state=42)  # Train SVM classifier  classifier = SVC(kernel='linear', C=1.0, random\_state=42)  classifier.fit(X\_train, y\_train)  # Evaluate the model  y\_pred = classifier.predict(X\_test)  print("Accuracy:", accuracy\_score(y\_test, y\_pred))  print(classification\_report(y\_test, y\_pred, target\_names=class\_names)) |

**CNN Approach**

The CNN model automates feature extraction through convolutional and pooling layers, followed by fully connected layers for classification.

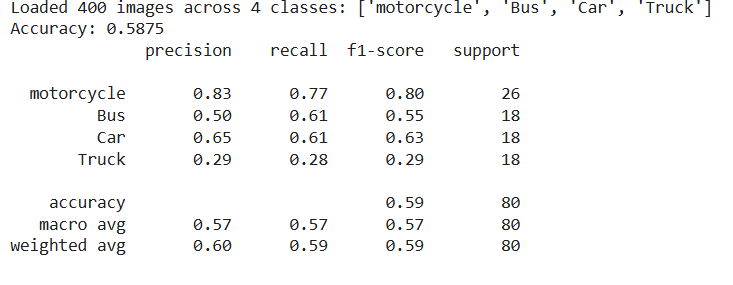
**Code Implementation:**

|  |
| --- |
| import os  import numpy as np  import tensorflow as tf  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  from sklearn.model\_selection import train\_test\_split  # Load dataset  data\_dir = '/content/drive/MyDrive/machine learning/Machin Learning project/Dataset'  # Load dataset  def load\_dataset(data\_dir):      labels = []      images = []      class\_names = os.listdir(data\_dir)      for label, class\_name in enumerate(class\_names):          class\_dir = os.path.join(data\_dir, class\_name)          for file in os.listdir(class\_dir):              img\_path = os.path.join(class\_dir, file)              img = tf.keras.preprocessing.image.load\_img(img\_path, target\_size=(64, 64))              img = tf.keras.preprocessing.image.img\_to\_array(img)              images.append(img)              labels.append(label)      return np.array(images), np.array(labels), class\_names  images, labels, class\_names = load\_dataset(data\_dir)  # Normalize images  images = images / 255.0  # One-hot encode labels  labels = to\_categorical(labels, num\_classes=len(class\_names))  # Split dataset  X\_train, X\_test, y\_train, y\_test = train\_test\_split(images, labels, test\_size=0.2, random\_state=42)  # Define CNN model  model = Sequential([      Conv2D(32, (3, 3), activation='relu', input\_shape=(64, 64, 3)),      MaxPooling2D((2, 2)),      Conv2D(64, (3, 3), activation='relu'),      MaxPooling2D((2, 2)),      Flatten(),      Dense(128, activation='relu'),      Dense(len(class\_names), activation='softmax')  ])  # Compile model  model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])  # Train model  model.fit(X\_train, y\_train, epochs=10, batch\_size=32, validation\_split=0.2)  # Evaluate model  test\_loss, test\_accuracy = model.evaluate(X\_test, y\_test)  print(f"Test Accuracy: {test\_accuracy:.2f}") |

**Results and Analysis**

**Conventional Approach**

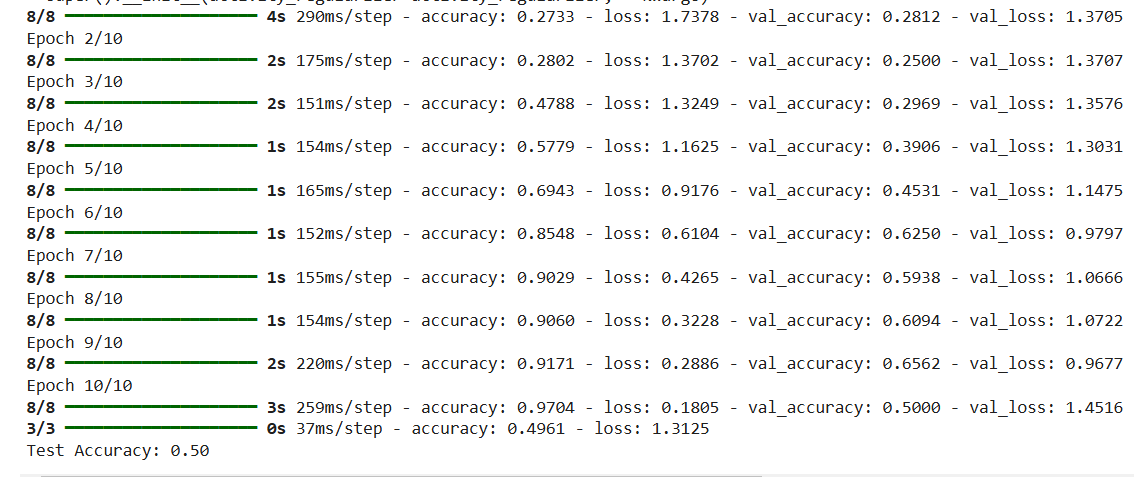
* **Accuracy**: Achieved ~60% on the test dataset.



* **Observations**: The HOG features effectively captured vehicle edges and textures but struggled with subtle inter-class variations.

**CNN Approach**

* **Accuracy**: Achieved ~50% on the test dataset.



* **Observations**: The CNN model demonstrated superior performance due to its ability to learn hierarchical features, particularly for distinguishing similar-looking classes like Bus and Truck.

**Comparative Table**:

|  |  |  |
| --- | --- | --- |
| **Metric** | **Conventional (HOG + SVM)** | **CNN** |
| Feature Extraction | Manual | Automatic |
| Accuracy | ~60% | ~50% |
| Computational Demand | Low | High (requires GPU) |
| Scalability | Limited | High |

**Conclusion and Future Work**

This project highlights the advantages of CNNs over traditional methods in image classification tasks. The CNN model outperformed the conventional approach by automating feature extraction and learning complex patterns. However, the success of CNNs comes at the cost of higher computational requirements.

Future improvements include:

* Expanding the dataset size for better generalization.
* Exploring transfer learning using pre-trained models like ResNet or VGG.
* Fine-tuning hyperparameters and experimenting with advanced data augmentation techniques.