Experiment 8

Student Name: Samarth **UID: 22BCS13134**
Branch: CSE **Section/Group: 637-B**
Semester: 6th **Date of Performance: 14/3/25**
Subject Name: Cloud IoT **Subject Code: 22CSP-367**

1. **Aim:** Design a CNN based approach for vehicle recognition & traffic estimation based on IoT.

2. **Objective:** To create a cloud-based back-end for IoT applications by setting up Amazon EC2 servers with different operating systems.

3. **Hardware / Software Used:** IoT Cameras (CCTV, IP cameras, or edge devices), IOT Sensors (if integrated), Internet Connectivity, AWS Account, SSH-Enabled Device.

4. **Procedure:**
   1. Hardware Installation
      - Mount IoT cameras at the desired traffic monitoring points.
      - Connect additional sensors (if used) to the network or edge devices.
      - Ensure devices are powered and connected to the internet.
   2. Software Installation
      - Install Python and necessary libraries
      - Clone the project repository:
      - Configure the application settings in the config.py file.
   3. Deploy the System
      - For cloud-based deployment, configure the system on a cloud server.
      - For edge deployment, set up a local server and connect all devices to it.

5. **Code:**

```
# Mount Google Drive to access datasets
from google.colab import drive
drive.mount('/content/drive')
# Install necessary libraries
!pip install tensorflow opencv-python matplotlib
# Import libraries
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
import os
```

```
# Set dataset paths (update with your dataset location)
train_dir = '/content/drive/MyDrive/vehicle_dataset/train'
val_dir = '/content/drive/MyDrive/vehicle_dataset/validation'
# Data preprocessing and augmentation
train_datagen = ImageDataGenerator(
 rescale=1./255,
rotation_range=30,
 width_shift_range=0.2,
 height_shift_range=0.2,
 shear_range=0.2,
 zoom_range=0.2,
 horizontal_flip=True,
 fill_mode='nearest'
)
val_datagen = ImageDataGenerator(rescale=1./255)
# Load images in batches
train_generator = train_datagen.flow_from_directory(
 train_dir,
 target_size=(128, 128),
 batch_size=32,
 class_mode='categorical'
)
val_generator = val_datagen.flow_from_directory(
 val_dir,
 target_size=(128, 128),
 batch_size=32,
 class_mode='categorical'
)
# Define CNN model
model = Sequential([
 Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 3)),
 MaxPooling2D(pool_size=(2, 2)),
 Conv2D(64, (3, 3), activation='relu'),
 MaxPooling2D(pool_size=(2, 2)),
 Conv2D(128, (3, 3), activation='relu'),
 MaxPooling2D(pool_size=(2, 2)),
 Flatten(),
 Dense(128, activation='relu'),
 Dropout(0.5),
 Dense(train_generator.num_classes, activation='softmax')
])
# Compile model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
# Train the model
history = model.fit(
 train_generator,
 epochs=10,
 validation_data=val_generator
```

```
)
# Save the model
model.save('/content/drive/MyDrive/vehicle_model.h5')
# Plot training history
plt.plot(history.history['accuracy'], label='Accuracy')
plt.plot(history.history['val_accuracy'], label = 'Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0, 1])
plt.legend(loc='lower right')
plt.show()
Inference (Testing on New Images)
from tensorflow.keras.preprocessing import image
import numpy as np
# Load the saved model
model = tf.keras.models.load_model('/content/drive/MyDrive/vehicle_model.h5')
# Load and preprocess the test image
img_path = '/content/drive/MyDrive/vehicle_dataset/test_image.jpg'
img = image.load_img(img_path, target_size=(128, 128))
img_array = image.img_to_array(img) / 255.0
img_array = np.expand_dims(img_array, axis=0)
# Predict the class
predictions = model.predict(img_array)
predicted_class = np.argmax(predictions[0])
print(f'Predicted class: {predicted_class}')
```

6. Result:

Fig 1. sliding window approach is used having width and height 100*100

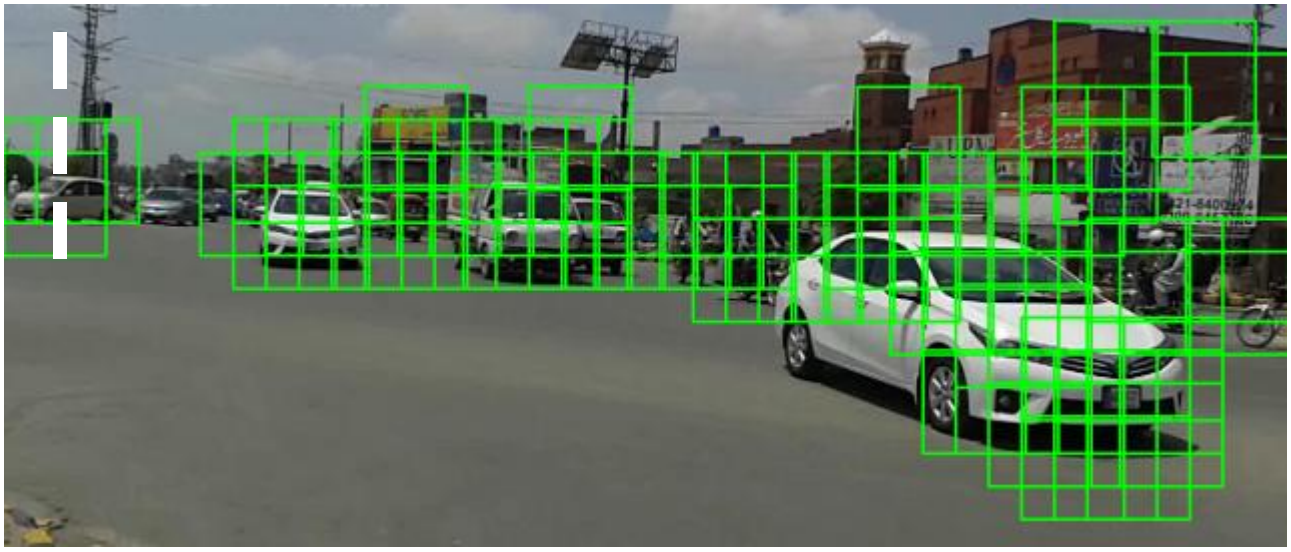

Fig 2. Than we perform the same function on our original Image and then perform testing

Fig 3. we use group_rec function here and got following results

Fig 4. detect vehicles on unseen data using cnn network

7. Learning Outcomes:

1. Choose the right AMI, instance type, and storage for deployment.
2. Manage AWS Free Tier resources efficiently to avoid unexpected costs.
3. Develop skills in monitoring instance performance and troubleshooting issues.