CourseName: Computer Vision Lab **Course Code:** CSP-422

Experiment: 2.3

Aim: Write a program to analyze various object detection algorithms with machine learning.

Software Required: Any Python IDE (e.g.: PyCharm, Jupyter Notebook, GoogleCollab)

Technique used:

Object detection algorithms are a type of computer vision algorithm that aims to identify and locate objects within images or video frames. These algorithms have various applications, including autonomous vehicles, surveillance, image retrieval, and augmented reality. Here are some key object detection algorithms:

- R-CNN (Region-based Convolutional Neural Networks)
- Fast R-CNN
- Faster R-CNN
- YOLO (You Only Look Once)
- SSD (Single Shot MultiBox Detector)
- RetinaNet
- Mask R-CNN
- EfficientDet

Steps:

1. Import necessary libraries and modules for your object detection project, such as OpenCV, NumPy, and OS.
2. Acquire a dataset with labeled images suitable for object detection (e.g., COCO dataset).
3. Resize and normalize the images in the dataset to prepare them for training.
4. Divide the dataset into training and testing sets to assess the performance of the models.
5. Choose and implement object detection algorithms: Select different object detection algorithms like Faster R-CNN, YOLO, or SSD based on your project requirements.
6. Use a machine learning library to train each chosen algorithm on the training set.
7. Assess the performance of each algorithm on the testing set using appropriate metrics, such as precision,

Name: Abhinav Verma **UID:** 20BCS9258

recall, and mean average precision (mAP).

8. Compare and analyze the results obtained from different algorithms to identify their strengths, weaknesses, and trade-offs.
9. Based on the analysis, draw conclusions about the suitability of each algorithm for your specific object detection task.

Implementation:

```
# 1. Import necessary libraries and modules
import cv2
import numpy as np
import os
import tensorflow as tf
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D
from tensorflow.keras.utils import to_categorical

# 2. Load and preprocess the CIFAR-10 dataset
(X_train, y_train), (X_test, y_test) = cifar10.load_data()
X_train = X_train.astype('float32') / 255.0
X_test = X_test.astype('float32') / 255.0
y_train = to_categorical(y_train, num_classes=10)
y_test = to_categorical(y_test, num_classes=10)

# 3. Initialize the model
model = Sequential()

# 4. Add layers to the model
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(32, 32,
3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(10, activation='softmax'))  # Assuming 10 classes in CIFAR-10
```

CourseName: Computer Vision Lab **Course Code:** CSP-422

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

# 6. Train the model
model.fit(X_train, y_train, epochs=10, batch_size=64, validation_data=(X_test,
y_test))

# 7. Evaluate the model
_, accuracy = model.evaluate(X_test, y_test)

# 8. Print the accuracy
print("Accuracy:", accuracy)
```

Output screenshot:

```
# 8. Print the accuracy
print("Accuracy:",accuracy)
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 [==============================] - 3s 0us/step
Epoch 1/10
782/782 [==============================] - 44s 55ms/step - loss: 1.5242 - accuracy: 0.4593 - val_loss: 1.3003 - val_accuracy: 0.5415
Epoch 2/10
782/782 [==============================] - 39s 50ms/step - loss: 1.2109 - accuracy: 0.5789 - val_loss: 1.1474 - val_accuracy: 0.5991
Epoch 3/10
782/782 [==============================] - 38s 49ms/step - loss: 1.0873 - accuracy: 0.6219 - val_loss: 1.0892 - val_accuracy: 0.6152
Epoch 4/10
782/782 [==============================] - 42s 54ms/step - loss: 1.0016 - accuracy: 0.6505 - val_loss: 1.1135 - val_accuracy: 0.6119
Epoch 5/10
782/782 [==============================] - 38s 49ms/step - loss: 0.9368 - accuracy: 0.6761 - val_loss: 1.0569 - val_accuracy: 0.6331
Epoch 6/10
782/782 [==============================] - 38s 48ms/step - loss: 0.8677 - accuracy: 0.6986 - val_loss: 1.0942 - val_accuracy: 0.6189
Epoch 7/10
782/782 [==============================] - 38s 49ms/step - loss: 0.8172 - accuracy: 0.7169 - val_loss: 1.0385 - val_accuracy: 0.6433
Epoch 8/10
782/782 [==============================] - 42s 53ms/step - loss: 0.7662 - accuracy: 0.7330 - val_loss: 1.0147 - val_accuracy: 0.6493
Epoch 9/10
782/782 [==============================] - 38s 48ms/step - loss: 0.7196 - accuracy: 0.7504 - val_loss: 1.0107 - val_accuracy: 0.6578
Epoch 10/10
782/782 [==============================] - 38s 48ms/step - loss: 0.6833 - accuracy: 0.7623 - val_loss: 1.0675 - val_accuracy: 0.6486
313/313 [==============================] - 4s 11ms/step - loss: 1.0675 - accuracy: 0.6486
Accuracy: 0.6485999822616577
```

Output Evaluation

Name: Abhinav Verma **UID:** 20BCS9258