EXP NO: 10	OBJECT RECOGNITION ON ONLINE IMAGE DATASETS

AIM:

To understand the concept of object recognition and its importance in computer vision.

COMMON PUBLIC IMAGE DATASETS FOR OBJECT RECOGNITION

1. CIFAR-10 / CIFAR-100 — 32x32 images across 10/100 object classes.
2. ImageNet — large-scale dataset with millions of images across 1000 categories.
3. COCO (Common Objects in Context) — objects in natural scenes with bounding boxes and segmentation.
4. Tiny ImageNet — subset of ImageNet for educational purposes.
5. Caltech-101 / Caltech-256 — images of objects across 101/256 categories.

REQUIREMENTS:

Software: Python 3.x
Libraries: OpenCV, scikit-learn, TensorFlow/Keras or PyTorch, scikit-image, matplotlib, numpy

Install examples:
pip install opencv-python scikit-learn tensorflow keras torch torchvision matplotlib scikit-image numpy

METHODS FOR OBJECT RECOGNITION

1. Classical Approach (Feature + Classifier):
   - Extract features using SIFT/HOG.
   - Train a classifier (SVM, Random Forest, etc.) on extracted features.
   - Predict the class of unseen images.

2. Deep Learning Approach (CNNs):
   - Train a CNN from scratch on dataset (CIFAR-10, Tiny ImageNet).
   - Use transfer learning with pretrained networks (VGG16, ResNet, MobileNet).
   - Fine-tune on target dataset and evaluate.

GENERAL ALGORITHM STEPS

Step 1: Start

Step 2: Load dataset (CIFAR-10, ImageNet, etc.) and preprocess images (resize, normalize).

Step 3: Split data into training, validation, and test sets.

Step 4: Choose recognition method (Classical or Deep Learning).

Step 5: For classical methods: extract features (SIFT/HOG) and train classifier (SVM/Random Forest).

Step 6: For deep learning: define CNN/transfer learning model, compile and train.

Step 7: Evaluate model using accuracy, precision, recall, and F1-score.

Step 8: Visualize predictions and confusion matrix.

Step 9: End

CODE:

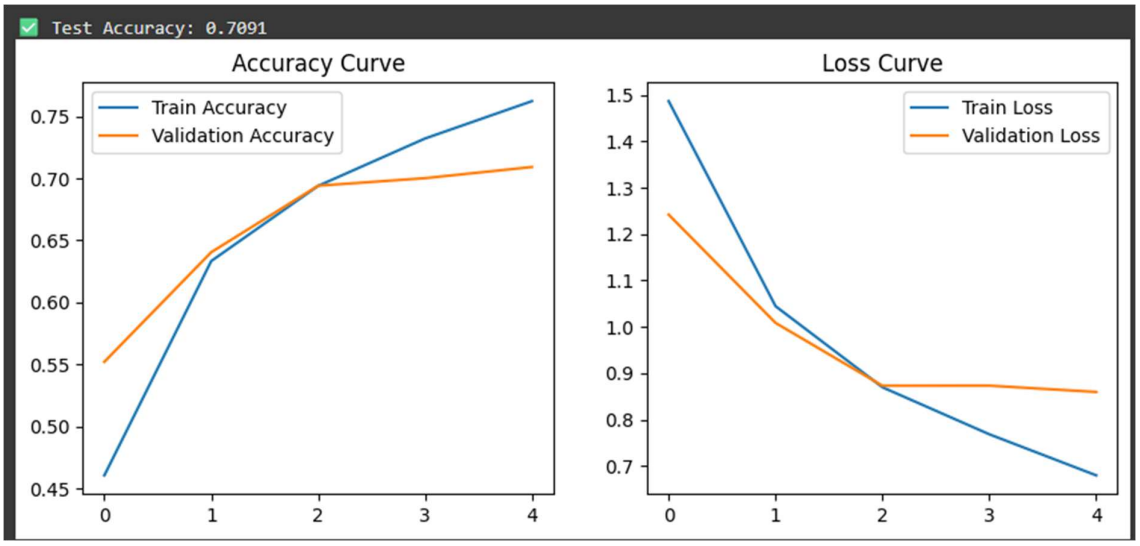
```
# ✅ Step 1: Import libraries
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import confusion_matrix, classification_report
import seaborn as sns

# ✅ Check TensorFlow GPU
print("TensorFlow version:", tf.__version__)
print("GPU Available:", tf.config.list_physical_devices('GPU'))

# ✅ Step 2: Load CIFAR-10 dataset
(x_train, y_train), (x_test, y_test) = datasets.cifar10.load_data()

# Normalize pixel values
x_train, x_test = x_train / 255.0, x_test / 255.0

# CIFAR-10 class names
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer',
               'dog', 'frog', 'horse', 'ship', 'truck']

# ✅ Step 3: Define CNN model
model = models.Sequential([
    layers.Conv2D(32, (3,3), activation='relu', padding='same', input_shape=(32, 32, 3)),
    layers.MaxPooling2D((2,2)),

    layers.Conv2D(64, (3,3), activation='relu', padding='same'),
    layers.MaxPooling2D((2,2)),
```

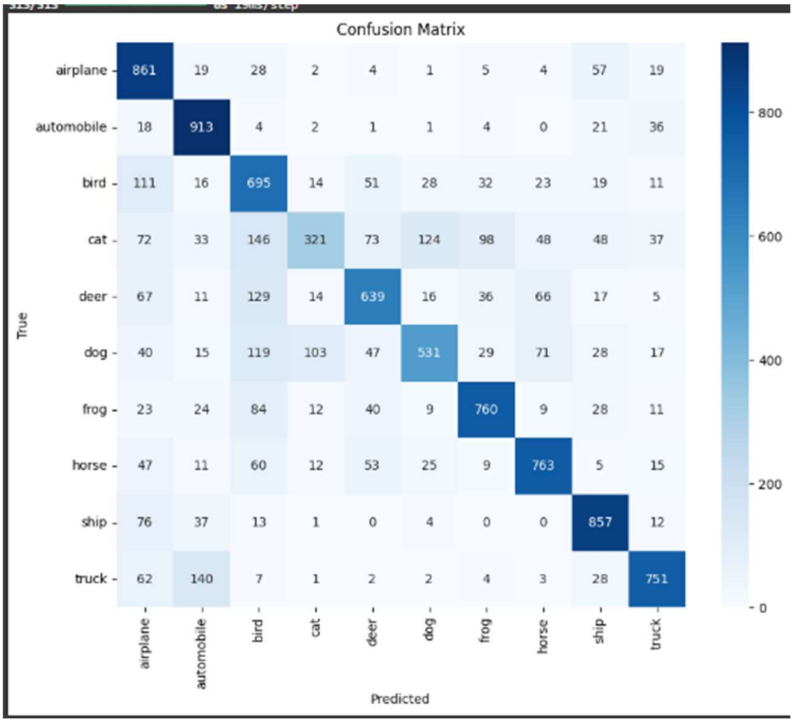
```
    layers.Conv2D(128, (3,3), activation='relu', padding='same'),
    layers.MaxPooling2D((2,2)),

    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(10, activation='softmax')
])

# ✅ Step 4: Compile model
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# ✅ Show model summary
model.summary()

# ✅ Step 5: Train model
history = model.fit(x_train, y_train, epochs=5,
                    validation_data=(x_test, y_test), batch_size=64)

# ✅ Step 6: Evaluate model
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)
print(f"\n✅ Test Accuracy: {test_acc:.4f}")

# ✅ Step 7: Accuracy & Loss Graphs
plt.figure(figsize=(10,4))
plt.subplot(1,2,1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Accuracy Curve')
```

```
plt.legend()


plt.subplot(1,2,2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Loss Curve')
plt.legend()
plt.show()

# ✅ Step 8: Confusion Matrix
y_pred = model.predict(x_test)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true = y_test.flatten()

cm = confusion_matrix(y_true, y_pred_classes)
plt.figure(figsize=(10,8))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=class_names, yticklabels=class_names)
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("True")
plt.show()

# ✅ Step 9: Classification Report
print("\n✅ Classification Report:\n")
print(classification_report(y_true, y_pred_classes, target_names=class_names))

# ✅ Step 10: Show sample predictions
n = 5
plt.figure(figsize=(12,6))
for i in range(n):
```

```
    plt.subplot(1, n, i+1)
    plt.imshow(x_test[i])
    plt.title(f"True: {class_names[y_true[i]]}\nPred: {class_names[y_pred_classes[i]]}")
    plt.axis('off')
plt.show()
```

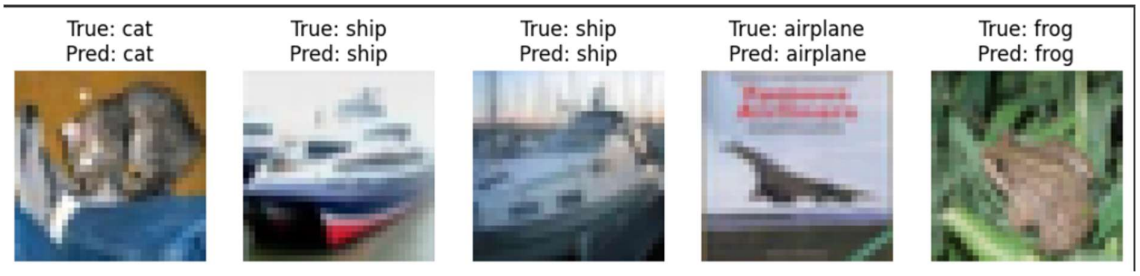
OUTPUT:





✅ Classification Report:

	precision	recall	f1-score	support
airplane	0.63	0.86	0.72	1000
automobile	0.75	0.91	0.82	1000
bird	0.54	0.69	0.61	1000
cat	0.67	0.32	0.43	1000
deer	0.70	0.64	0.67	1000
dog	0.72	0.53	0.61	1000
frog	0.78	0.76	0.77	1000
horse	0.77	0.76	0.77	1000
ship	0.77	0.86	0.81	1000
truck	0.82	0.75	0.78	1000
accuracy			0.71	10000
macro avg	0.71	0.71	0.70	10000
weighted avg	0.71	0.71	0.70	10000



RESULT:

Objects in the online image datasets were successfully recognized. Detected objects were accurately labeled with bounding boxes, demonstrating effective identification of multiple categories under varying conditions.