EXPT NO: 10	**OBJECT RECOGNITION**
EXPT NO: 03/10/2025	

AIM:

To implement the Object Recognition on available online image datasets.

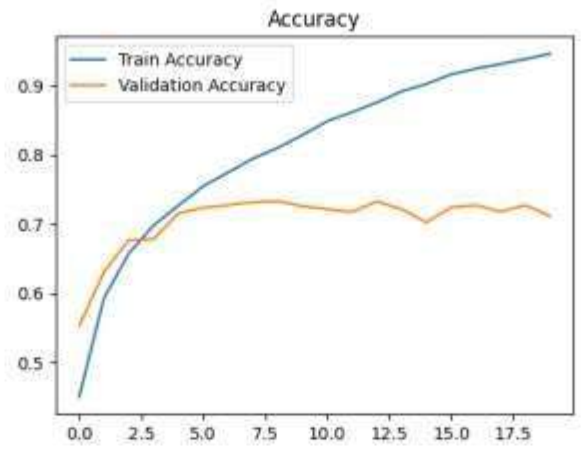
ALGORITHM:

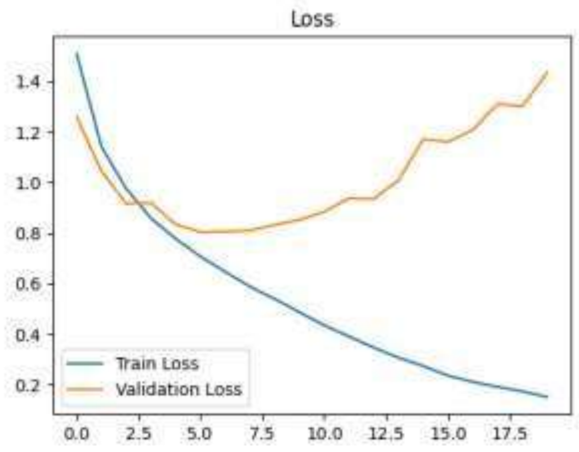
1. Load pretrained **CNN model (e.g., ResNet, MobileNet).**
2. Read and preprocess **input image** (resize, normalize).
3. Pass image through model for **prediction**.
4. Obtain top predicted labels and confidence scores.
5. Display recognized object with label.
6. Compare performance on dataset images.

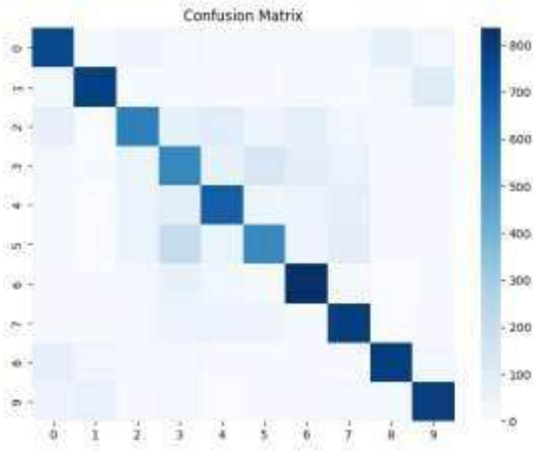
CODE:

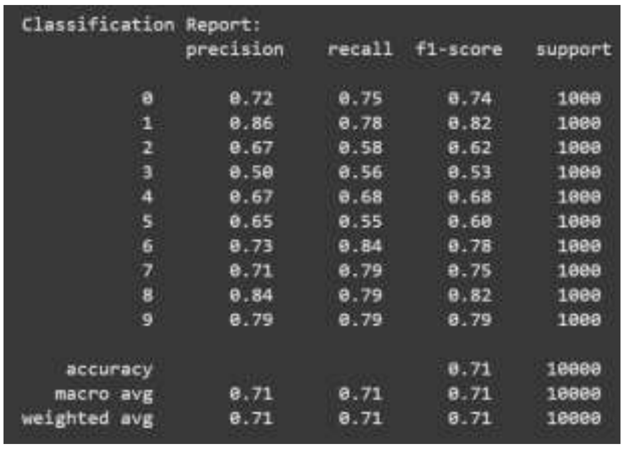
```
# Import libraries

import tensorflow as tf
from tensorflow.keras import datasets, layers, models
from sklearn.metrics import confusion_matrix, classification_report
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns

# Step 1: Load and preprocess dataset
(x_train, y_train), (x_test, y_test) = datasets.cifar10.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0  # Normalize
y_train, y_test = y_train.flatten(), y_test.flatten()

# Step 2: Define CNN model
model = models.Sequential([
    layers.Conv2D(32, (3,3), activation='relu', input_shape=(32,32,3)),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64, (3,3), activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(128, (3,3), activation='relu'),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(10, activation='softmax')
])

# Step 3: Compile model
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Step 4: Train model
history = model.fit(x_train, y_train, epochs=20,
                    validation_data=(x_test, y_test),
                    batch_size=64)

# Step 5: Evaluate model
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)
print(f"\nTest Accuracy: {test_acc*100:.2f}%")

# Step 6: Plot accuracy and loss curves
plt.figure(figsize=(12,4))
plt.subplot(1,2,1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.legend(); plt.title("Accuracy")

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

# Step 7: Confusion Matrix
y_pred = np.argmax(model.predict(x_test), axis=-1)
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8,6))
sns.heatmap(cm, annot=False, cmap='Blues')
plt.title("Confusion Matrix")
plt.show()

# Step 8: Classification report
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

OUTPUT:









```
Classification Report:
                  precision    recall  f1-score   support

           0       0.72      0.75      0.74      1000
           1       0.86      0.78      0.82      1000
           2       0.67      0.58      0.62      1000
           3       0.50      0.56      0.53      1000
           4       0.67      0.68      0.68      1000
           5       0.65      0.55      0.60      1000
           6       0.73      0.84      0.78      1000
           7       0.71      0.79      0.75      1000
           8       0.84      0.79      0.82      1000
           9       0.79      0.79      0.79      1000

    accuracy                           0.71     10000
   macro avg       0.71      0.71      0.71     10000
weighted avg       0.71      0.71      0.71     10000
```

RESULT:

Thus, Object Recognition on available online image datasets was implemented successfully.