**Lab Report: Advanced Computer Vision - Lab 5**

**Student Name:** Anmol Pandey  
**Student ID:** E22CSEU1069  
**Course:** Advanced Computer Vision  
**Lab Number:** 5

## ****1. Objective****

The objective of this lab was to explore Discrete Cosine Transform (DCT) for image compression and deep learning-based classification using Convolutional Neural Networks (CNNs). The tasks focused on frequency domain transformations and supervised learning on image datasets.

## ****2. Introduction****

This lab covers:

* **Block-wise Discrete Cosine Transform (DCT) and Inverse DCT**
* **Image Classification using CNN on MNIST and CIFAR-10**
* **Model Evaluation using Confusion Matrix and ROC Curves**

DCT is widely used in image compression (JPEG) to transform an image into frequency components. CNNs are powerful tools for image classification, leveraging feature extraction through convolutional layers.

## ****3. Implementation Details****

### ****3.1 Libraries Used****

The following Python libraries were used:

* cv2, numpy: For image processing and transformations.
* matplotlib.pyplot, seaborn: For visualization.
* keras: For deep learning model implementation.
* sklearn: For model evaluation metrics.

### ****3.2 Block-wise DCT and Inverse DCT****

import cv2

import numpy as np

def blockwise\_dct(image, block\_size=8):

h, w = image.shape

dct\_image = np.zeros\_like(image, dtype=np.float32)

for i in range(0, h, block\_size):

for j in range(0, w, block\_size):

block = image[i:i + block\_size, j:j + block\_size]

dct\_block = cv2.dct(np.float32(block))

dct\_image[i:i + block\_size, j:j + block\_size] = dct\_block

return dct\_image

**Explanation:**

* Processes the image in 8×8 blocks.
* Applies DCT using OpenCV’s cv2.dct().

### ****3.3 Image Classification using CNN****

#### ****3.3.1 Loading MNIST and CIFAR-10 datasets****

from keras.datasets import mnist, cifar10

(x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data()

**Explanation:**

* Loads the MNIST dataset for handwritten digit recognition.
* Can also load CIFAR-10 for object classification.

#### ****3.3.2 Preprocessing Data****

from keras.utils import to\_categorical

x\_train = x\_train.reshape(-1, 28, 28, 1) / 255.0

x\_test = x\_test.reshape(-1, 28, 28, 1) / 255.0

y\_train = to\_categorical(y\_train)

y\_test = to\_categorical(y\_test)

**Explanation:**

* Normalizes pixel values to [0,1].
* Converts labels to one-hot encoding.

#### ****3.3.3 CNN Model Architecture****

from keras.models import Sequential

from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

model = Sequential([

Conv2D(32, (3,3), activation='relu', input\_shape=(28,28,1)),

MaxPooling2D((2,2)),

Flatten(),

Dense(128, activation='relu'),

Dropout(0.5),

Dense(10, activation='softmax')

])

**Explanation:**

* Uses convolutional layers for feature extraction.
* Fully connected layers classify images.

#### ****3.3.4 Training the Model****

model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

model.fit(x\_train, y\_train, epochs=10, validation\_data=(x\_test, y\_test))

**Explanation:**

* Uses Adam optimizer.
* Trains for 10 epochs.

### ****3.4 Model Evaluation****

#### ****Confusion Matrix****

from sklearn.metrics import confusion\_matrix

import seaborn as sns

y\_pred = model.predict(x\_test).argmax(axis=1)

y\_true = y\_test.argmax(axis=1)

cm = confusion\_matrix(y\_true, y\_pred)

sns.heatmap(cm, annot=True, fmt='d')

plt.show()

**Explanation:**

* Computes confusion matrix to analyze classification performance.

#### ****ROC Curve****

from sklearn.metrics import roc\_curve, auc

fpr, tpr, \_ = roc\_curve(y\_test[:, 1], y\_pred)

plt.plot(fpr, tpr, label='ROC Curve')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.legend()

plt.show()

**Explanation:**

* Plots ROC curve for performance evaluation.

## ****4. Results and Observations****

The implemented code produces the following outputs:

1. **DCT and Inverse DCT Visualizations**: Frequency domain representation of images.
2. **Trained CNN Model**: Achieves high accuracy on MNIST.
3. **Confusion Matrix**: Identifies classification performance per class.
4. **ROC Curve**: Evaluates model discrimination ability.

**Observations:**

* DCT reduces image redundancy and is useful in compression.
* CNN achieves high accuracy in image classification.
* ROC curve analysis confirms model reliability.

## ****5. Conclusion****

This lab introduced key concepts in image transformations and deep learning. The results demonstrated how DCT transforms images into frequency components and how CNNs effectively classify images. These techniques have applications in compression, denoising, and object recognition.

## ****6. Future Scope****

* Apply DCT to color images.
* Train CNNs on other datasets (CIFAR-10, ImageNet).
* Explore different CNN architectures (ResNet, VGG16).
* Implement transfer learning for improved accuracy.

**End of Report**