**Deep Learning Lab**

**Experiment -7**

Name – Adarsh Kumar Singh (24215011111) M.Tech AI

Q.1 "Image Classification Using CIFAR-10 Dataset using simple deep network with 4 hidden layers and 3 dropout layer also apply pruning and quantization to reduce size and report size of model"

1.Train the original model on CIFAR-10.

2.Save the original model (model.h5).

3.Apply pruning manually:

* If a weight is less than 0.01, we set it to 0.

Save the pruned model (pruned\_model.h5).

Apply post-training quantization:

* Converts weights from 32-bit float → 8-bit int.

Save the quantized model (quantized\_model.tflite).

Compare and print the sizes of all three models. **Step 1: Load CIFAR-10 Dataset**

import tensorflow as tf from tensorflow import keras from tensorflow.keras import layers import numpy as np import os import tempfile import struct

# Step 2: Normalize the Image Data

# Load CIFAR-10 dataset

(x\_train, y\_train), (x\_test, y\_test) = keras.datasets.cifar10.load\_data()

# Normalize pixel values to [0,1] x\_train, x\_test = x\_train / 255.0, x\_test / 255.0 # Convert labels to one-hot encoding (Fixing the issue) y\_train = keras.utils.to\_categorical(y\_train, 10) y\_test = keras.utils.to\_categorical(y\_test, 10)

## Step 3: Build and Train a Deep Neural Network

def create\_model():

model = keras.Sequential([ layers.Flatten(input\_shape=(32, 32, 3)), # Flatten input images layers.Dense(512, activation='relu'), layers.Dropout(0.2), # First dropout layers.Dense(256, activation='relu'), layers.Dropout(0.2), # Second dropout layers.Dense(128, activation='relu'), layers.Dense(64, activation='relu'), layers.Dropout(0.2), # Third dropout layers.Dense(10, activation='softmax') # Output layer

])

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

return model # Create model instance model = create\_model() # Train the model model.fit(x\_train, y\_train, epochs=50, validation\_data=(x\_test, y\_test), batch\_size=64)

Epoch 1/50

**782/782** ━━━━━━━━━━━━━━━━━━━━ **9s** 8ms/step - accuracy: 0.1911

- loss: 2.1619 - val\_accuracy: 0.3230 - val\_loss: 1.8654 Epoch 50/50

**782/782** ━━━━━━━━━━━━━━━━━━━━ **3s** 4ms/step - accuracy: 0.4626 - loss: 1.5011 - val\_accuracy: 0.4832 - val\_loss: 1.4592

# Step 4: Save the Model

\_, model\_file = tempfile.mkstemp('.h5') # Create temporary file model.save(model\_file) # Save model original\_size = os.path.getsize(model\_file) / (1024 \* 1024) # Convert to MB print(f"Original Model Size: {original\_size:.2f} MB")

OUTPUT

Original Model Size: 20.03 MB

## Step 5: Apply Model Pruning (Reducing Unimportant Weights)

pruned\_model = tf.keras.models.clone\_model(model) pruned\_model.set\_weights([np.where(np.abs(w) > 0.01, w, 0) for w in model.get\_weights()])

pruned\_model.save("pruned\_model.h5")

## Step 6: Apply Quantization (Reducing Precision of Weights)

# Apply quantization (convert model to TensorFlow Lite format) converter = tf.lite.TFLiteConverter.from\_keras\_model(pruned\_model) converter.optimizations = [tf.lite.Optimize.DEFAULT] quantized\_model = converter.convert() # Save quantized model with open("quantized\_model.tflite", "wb") as f:

f.write(quantized\_model) original\_size = os.path.getsize("model.h5") / 1024 # Convert to KB pruned\_size = os.path.getsize("pruned\_model.h5") / 1024 quantized\_size = os.path.getsize("quantized\_model.tflite") / 1024 print(f"Original Model Size: {original\_size:.2f} KB") print(f"Pruned Model Size: {pruned\_size:.2f} KB") print(f"Quantized Model Size: {quantized\_size:.2f} KB")

## Step 7: Compare Model Size

**Original Model Size: 20512.41 KB**

**Pruned Model Size: 6856.16 KB**

**Quantized Model Size: 1724.73 KB**

**Pruning:Pruning removes unnecessary weights (connections) in a neural network by setting small weights to zero. This reduces model size and speeds up inference while maintaining accuracy. It helps in optimizing storage and computational efficiency.**

**Quantization:Quantization reduces the precision of model weights and activations (e.g., from 32-bit floating-point to 8-bit integers). This significantly decreases the model size and makes it faster, especially for deployment on mobile and edge devices.**

# 