1. Design a Neural Network using ANN algorithm on CIFAR 10 dataset

https://www.cs.toronto.edu/~kriz/cifar.html

In [1]:
```
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.utils import to_categorical

# Load the CIFAR-10 dataset
(train_images, train_labels), (test_images, test_labels) = cifar10.load_data()

# Preprocess the data
train_images, test_images = train_images / 255.0, test_images / 255.0
train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)

# Build the neural network
model = models.Sequential()

# Convolutional Layer 1
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)
model.add(layers.MaxPooling2D((2, 2)))

# Convolutional Layer 2
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))

# Convolutional Layer 3
model.add(layers.Conv2D(64, (3, 3), activation='relu'))

# Flatten the output from the convolutional layers
model.add(layers.Flatten())

# Fully connected layers
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))  # 10 classes in CIFAR-10

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

# Train the model
history = model.fit(train_images, train_labels, epochs=10, validation_data=(te

# Evaluate the model
test_loss, test_acc = model.evaluate(test_images, test_labels)
print(f"Test accuracy: {test_acc}")
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.g
z (https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz)
170498071/170498071 [==============================] - 52s 0us/step
Epoch 1/10
1563/1563 [==============================] - 88s 53ms/step - loss: 1.5268 -
accuracy: 0.4436 - val_loss: 1.2528 - val_accuracy: 0.5459
Epoch 2/10
1563/1563 [==============================] - 82s 53ms/step - loss: 1.1599 -
accuracy: 0.5907 - val_loss: 1.0667 - val_accuracy: 0.6205
Epoch 3/10
1563/1563 [==============================] - 81s 52ms/step - loss: 0.9964 -
accuracy: 0.6519 - val_loss: 0.9673 - val_accuracy: 0.6586
Epoch 4/10
1563/1563 [==============================] - 85s 54ms/step - loss: 0.8935 -
accuracy: 0.6864 - val_loss: 0.9750 - val_accuracy: 0.6615
Epoch 5/10
1563/1563 [==============================] - 91s 58ms/step - loss: 0.8154 -
accuracy: 0.7162 - val_loss: 0.9421 - val_accuracy: 0.6755
Epoch 6/10
1563/1563 [==============================] - 86s 55ms/step - loss: 0.7585 -
accuracy: 0.7335 - val_loss: 0.8620 - val_accuracy: 0.7065
Epoch 7/10
1563/1563 [==============================] - 83s 53ms/step - loss: 0.7119 -
accuracy: 0.7495 - val_loss: 0.8409 - val_accuracy: 0.7121
Epoch 8/10
1563/1563 [==============================] - 84s 53ms/step - loss: 0.6593 -
accuracy: 0.7699 - val_loss: 0.8974 - val_accuracy: 0.7006
Epoch 9/10
1563/1563 [==============================] - 86s 55ms/step - loss: 0.6235 -
accuracy: 0.7823 - val_loss: 0.8570 - val_accuracy: 0.7118
Epoch 10/10
1563/1563 [==============================] - 80s 51ms/step - loss: 0.5885 -
accuracy: 0.7947 - val_loss: 0.9026 - val_accuracy: 0.7111
313/313 [==============================] - 5s 16ms/step - loss: 0.9026 - acc
uracy: 0.7111
Test accuracy: 0.7110999822616577
```

In [ ]: