Batch: HO DL 1 **Experiment Number: 4**

Roll Number: 16010421073 **Name: Keyur Patel**

Title of the Experiment: Transfer Learning with CNN

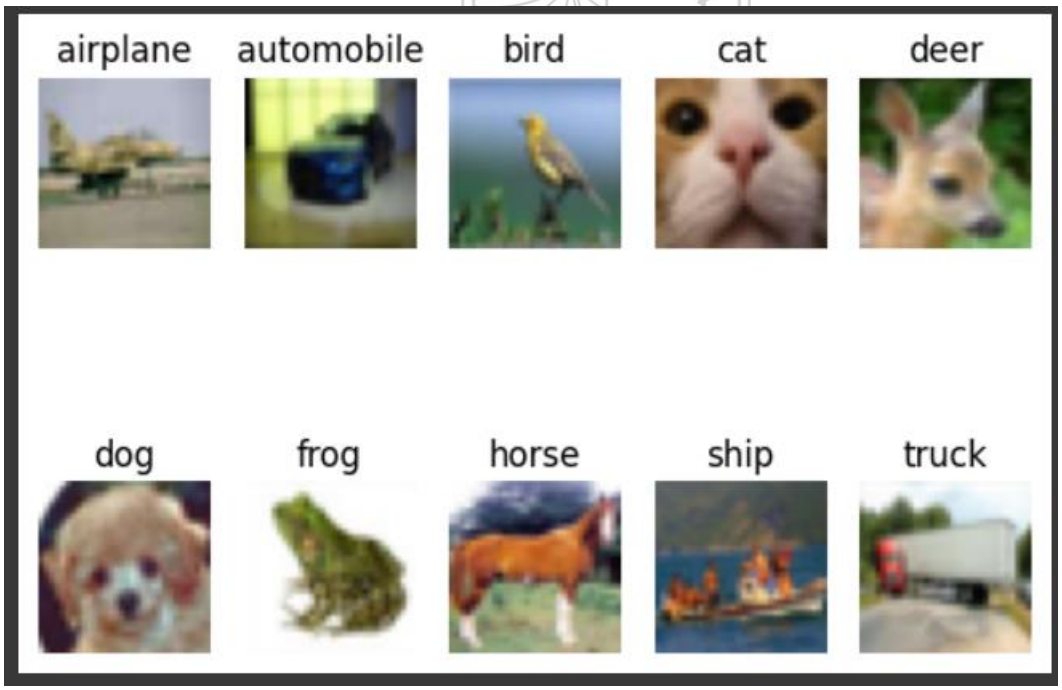
Program:

```
import tensorflow as tf
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten,
Dense, Dropout
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
import matplotlib.pyplot as plt
import numpy as np

(train_images, train_labels), (_, _) = cifar10.load_data()

num_classes = 10
class_labels = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog',
'frog', 'horse', 'ship', 'truck']

class_images = {label: [] for label in class_labels}

for i in range(num_classes):
    class_images[class_labels[i]] = train_images[np.where(train_labels
== i)[0]]

plt.figure(figsize=(12, 8))
for i in range(num_classes):
    plt.subplot(2, 5, i + 1)
    random_image_index = np.random.randint(0,
len(class_images[class_labels[i]]))
    plt.imshow(class_images[class_labels[i]][random_image_index])
    plt.title(class_labels[i])
    plt.axis('off')
plt.show()

model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
```

```
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(num_classes, activation='softmax')
])

model.summary()
tf.keras.utils.plot_model(model, to_file='model.png', show_shapes=True,
show_layer_names=True)

train_images = train_images / 255.0  # Normalize pixel values to [0, 1]

train_labels = to_categorical(train_labels, num_classes)

model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
history = model.fit(train_images, train_labels, epochs=10,
batch_size=64, validation_split=0.2)

train_accuracy = history.history['accuracy'][-1]
val_accuracy = history.history['val_accuracy'][-1]
print(f"Training Accuracy: {train_accuracy}, Validation Accuracy:
{val_accuracy}")

model_regularized = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu',
kernel_regularizer=tf.keras.regularizers.l2(0.001)),
    Dropout(0.5),
    Dense(num_classes, activation='softmax')
])

model_regularized.compile(optimizer='adam',
loss='categorical_crossentropy', metrics=['accuracy'])
history_regularized = model_regularized.fit(train_images, train_labels,
epochs=10, batch_size=64, validation_split=0.2)

plt.plot(history_regularized.history['loss'], label='Training Loss')
plt.plot(history_regularized.history['val_loss'], label='Validation
Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Loss Over Time')
```

```
plt.legend()
plt.show()

plt.plot(history_regularized.history['accuracy'], label='Training
Accuracy')
plt.plot(history_regularized.history['val_accuracy'], label='Validation
Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Accuracy Over Time')
plt.legend()
plt.show()

predictions = model_regularized.predict(train_images)
predicted_classes = np.argmax(predictions, axis=1)
print(classification_report(np.argmax(train_labels, axis=1),
predicted_classes))
```

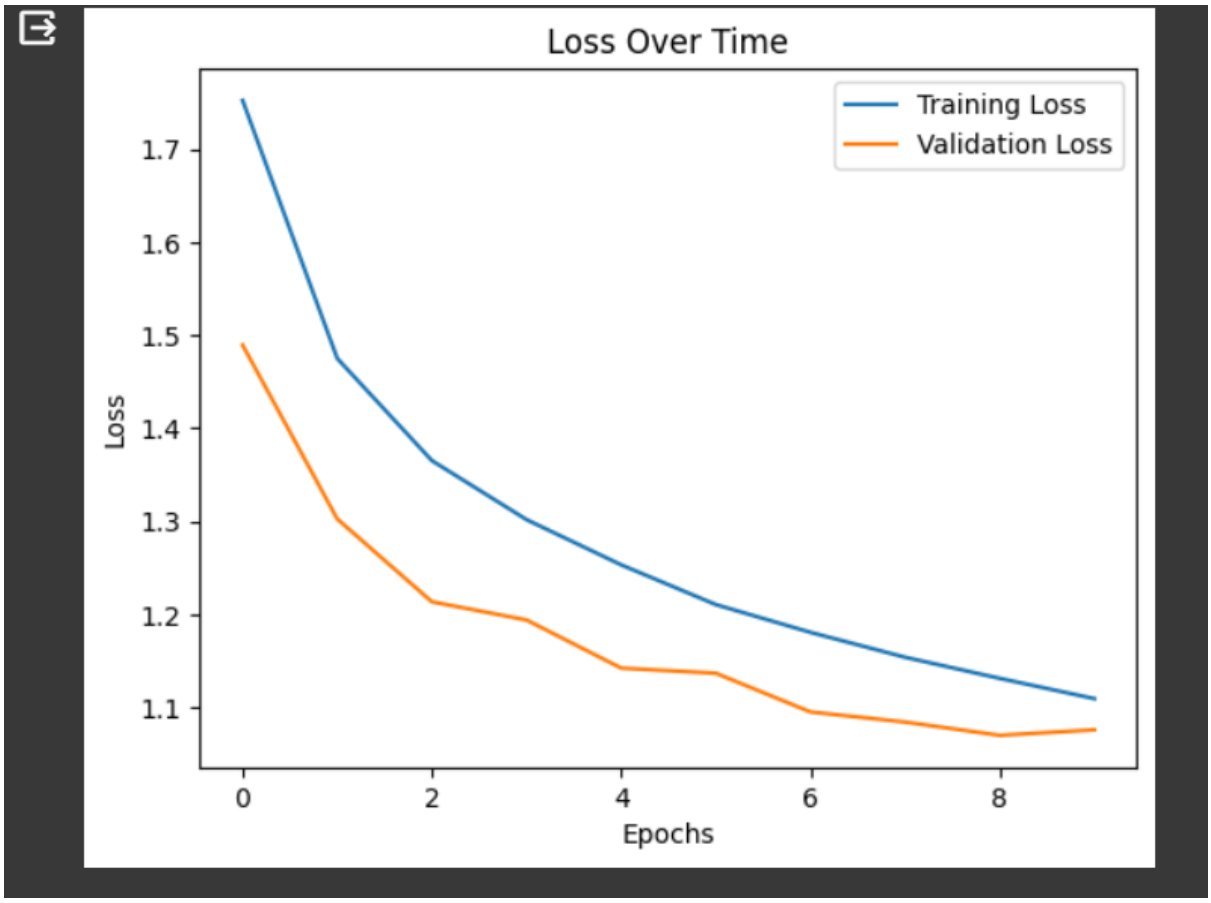
Output:

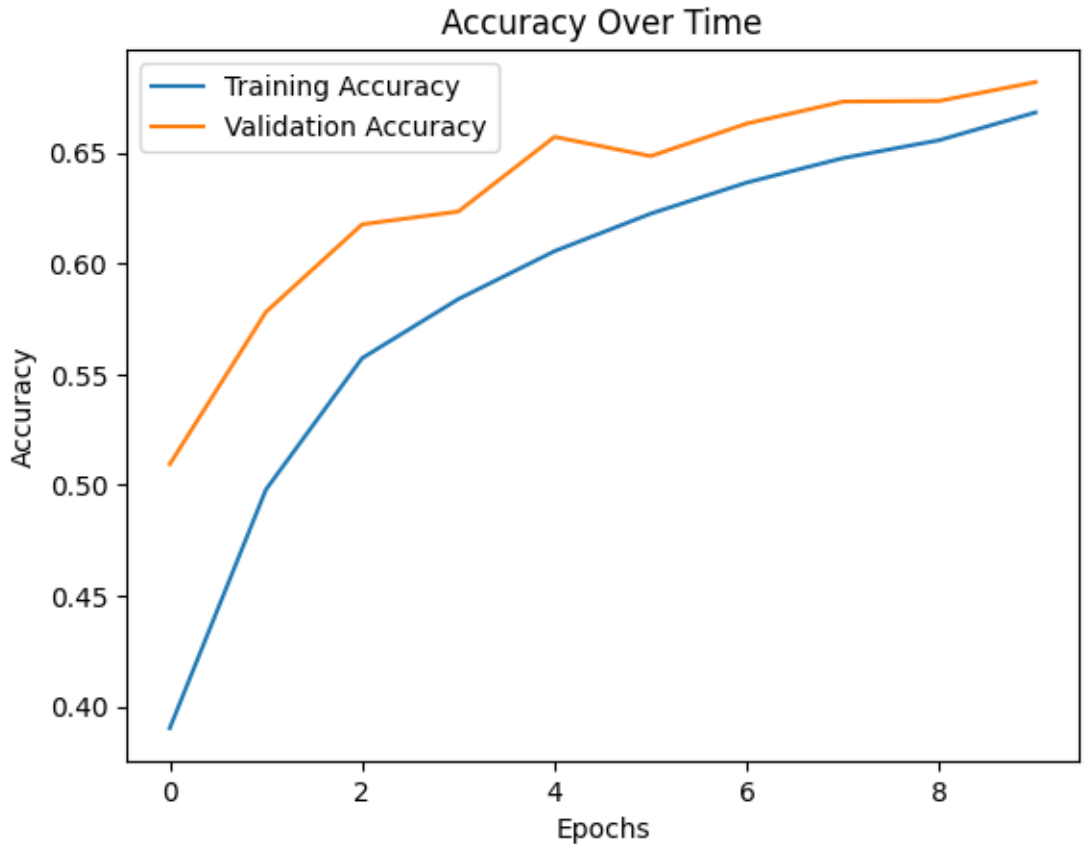


```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 30, 30, 32)        896

 max_pooling2d (MaxPooling2  (None, 15, 15, 32)        0
 D)

 conv2d_1 (Conv2D)           (None, 13, 13, 64)        18496

 max_pooling2d_1 (MaxPoolin  (None, 6, 6, 64)          0
 g2D)

 flatten (Flatten)           (None, 2304)              0

 dense (Dense)               (None, 128)               295040

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 10)                1290

=================================================================
Total params: 315722 (1.20 MB)
Trainable params: 315722 (1.20 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
Epoch 1/10
625/625 [==============================] - 53s 82ms/step - loss: 1.7215 - accuracy: 0.3678 - val_loss: 1.4481 - val_accuracy: 0.4891
Epoch 2/10
625/625 [==============================] - 52s 83ms/step - loss: 1.4020 - accuracy: 0.4972 - val_loss: 1.2242 - val_accuracy: 0.5779
Epoch 3/10
625/625 [==============================] - 52s 84ms/step - loss: 1.2617 - accuracy: 0.5540 - val_loss: 1.1206 - val_accuracy: 0.6010
Epoch 4/10
625/625 [==============================] - 49s 79ms/step - loss: 1.1802 - accuracy: 0.5836 - val_loss: 1.0642 - val_accuracy: 0.6211
Epoch 5/10
625/625 [==============================] - 53s 85ms/step - loss: 1.1165 - accuracy: 0.6040 - val_loss: 1.0149 - val_accuracy: 0.6443
Epoch 6/10
625/625 [==============================] - 52s 83ms/step - loss: 1.0603 - accuracy: 0.6265 - val_loss: 1.0135 - val_accuracy: 0.6456
Epoch 7/10
625/625 [==============================] - 51s 81ms/step - loss: 1.0141 - accuracy: 0.6438 - val_loss: 0.9534 - val_accuracy: 0.6683
Epoch 8/10
625/625 [==============================] - 50s 80ms/step - loss: 0.9762 - accuracy: 0.6547 - val_loss: 0.9365 - val_accuracy: 0.6734
Epoch 9/10
625/625 [==============================] - 51s 82ms/step - loss: 0.9471 - accuracy: 0.6653 - val_loss: 0.9064 - val_accuracy: 0.6817
Epoch 10/10
625/625 [==============================] - 50s 80ms/step - loss: 0.9047 - accuracy: 0.6799 - val_loss: 0.8935 - val_accuracy: 0.6866
Training Accuracy: 0.6799250245094299, Validation Accuracy: 0.6866000294685364
```

```
Epoch 1/10
625/625 [==============================] - 56s 87ms/step - loss: 1.7520 - accuracy: 0.3903 - val_loss: 1.4892 - val_accuracy: 0.5094
Epoch 2/10
625/625 [==============================] - 50s 81ms/step - loss: 1.4748 - accuracy: 0.4979 - val_loss: 1.3024 - val_accuracy: 0.5781
Epoch 3/10
625/625 [==============================] - 52s 83ms/step - loss: 1.3651 - accuracy: 0.5573 - val_loss: 1.2134 - val_accuracy: 0.6176
Epoch 4/10
625/625 [==============================] - 52s 83ms/step - loss: 1.3016 - accuracy: 0.5839 - val_loss: 1.1938 - val_accuracy: 0.6234
Epoch 5/10
625/625 [==============================] - 49s 79ms/step - loss: 1.2531 - accuracy: 0.6055 - val_loss: 1.1421 - val_accuracy: 0.6571
Epoch 6/10
625/625 [==============================] - 52s 83ms/step - loss: 1.2104 - accuracy: 0.6224 - val_loss: 1.1366 - val_accuracy: 0.6484
Epoch 7/10
625/625 [==============================] - 50s 81ms/step - loss: 1.1805 - accuracy: 0.6365 - val_loss: 1.0950 - val_accuracy: 0.6631
Epoch 8/10
625/625 [==============================] - 49s 78ms/step - loss: 1.1538 - accuracy: 0.6475 - val_loss: 1.0842 - val_accuracy: 0.6730
Epoch 9/10
625/625 [==============================] - 50s 80ms/step - loss: 1.1311 - accuracy: 0.6556 - val_loss: 1.0699 - val_accuracy: 0.6733
Epoch 10/10
625/625 [==============================] - 50s 80ms/step - loss: 1.1093 - accuracy: 0.6681 - val_loss: 1.0758 - val_accuracy: 0.6818
```






```
1563/1563 [==============================] - 24s 16ms/step
              precision    recall  f1-score   support

           0       0.79      0.76      0.78      5000
           1       0.88      0.82      0.85      5000
           2       0.75      0.46      0.57      5000
           3       0.60      0.47      0.53      5000
           4       0.57      0.80      0.67      5000
           5       0.74      0.48      0.58      5000
           6       0.62      0.91      0.74      5000
           7       0.77      0.78      0.78      5000
           8       0.76      0.89      0.82      5000
           9       0.82      0.84      0.83      5000

    accuracy                           0.72     50000
   macro avg       0.73      0.72      0.71     50000
weighted avg       0.73      0.72      0.71     50000
```

CO: CO3: Assimilate fundamentals of Convolutional Neural Network.

Conclusion:

In this experiment we learnt about transfer learning with CNN and performed the same. The output of the same has been displayed.