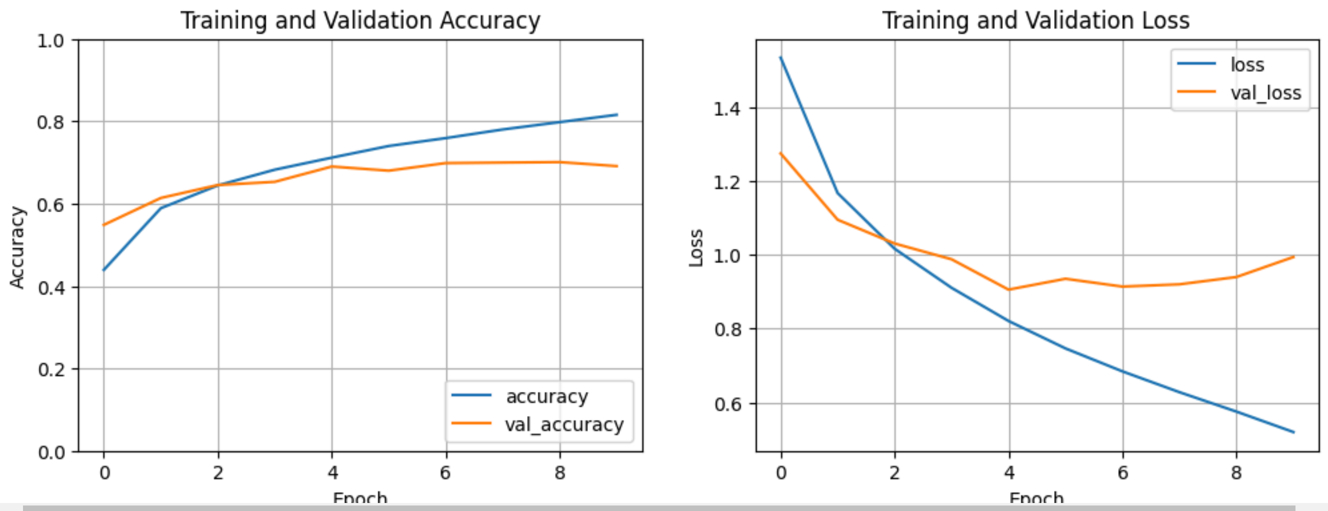
Image Classification Using CNN with the CIFAR-10 Dataset using **basic Hyperparameter Tunning**

```
pip install keras_tuner
```

```
# Step 1: Importing necessary libraries
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
from keras_tuner.tuners import RandomSearch
import matplotlib.pyplot as plt

# Step 2: Loading and preprocessing the CIFAR-10 dataset
(train_images, train_labels), (test_images, test_labels) = datasets.cifar10.load_data()
train_images, test_images = train_images / 255.0, test_images / 255.0

# Step 3: Defining the class names for CIFAR-10
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']

# Step 4: Define a function to build the model.
def build_model(hp):
  model = models.Sequential()
  model.add(layers.Conv2D(hp.Int('conv1_units', min_value=32, max_value=128, step=16), (3,3), activation='relu', input_shape=(32,32,3)))
  model.add(layers.MaxPooling2D((2,2)))
  model.add(layers.Conv2D(hp.Int('conv2_units', min_value=32, max_value=128, step=16), (3,3), activation='relu'))
  model.add(layers.MaxPooling2D((2,2)))
  model.add(layers.Conv2D(hp.Int('conv3_units', min_value=32, max_value=128, step=16), (3,3), activation='relu'))
  model.add(layers.Flatten())
  model.add(layers.Dense(hp.Int('dense_units', min_value=32, max_value=128, step=16), activation='relu'))
  model.add(layers.Dropout(hp.Float('dropout_rate', min_value=0.0, max_value=0.5, step=0.1)))
  model.add(layers.Dense(10))

  # Choose an optimizer and learning rate
  optimizer = tf.keras.optimizers.Adam(learning_rate=hp.Choice('learning_rate', values=[1e-2, 1e-3, 1e-4]))

  model.compile(optimizer=optimizer, loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True), metrics=['accuracy'])

  return model

# Step 5: Define the Tuner
tuner = RandomSearch(
    build_model,
    objective='val_accuracy',
    max_trials=10,
    executions_per_trial=1,
    directory='my_dir',
    project_name='cifar10_tunning'
)

# Step 6: Perform the Hyperparameter search
tuner.search(train_images, train_labels, epochs=5, validation_data=(test_images, test_labels))

# Step 7: Get the best Hyperparameters
best_hps = tuner.get_best_hyperparameters(num_trials=1)[0]

# Step 8: Build the model with the best Hyperparameters and train it
model = tuner.hypermodel.build(best_hps)
history = model.fit(train_images, train_labels, epochs=10, validation_data=(test_images, test_labels))

# Step 9: Plotting training & validation accuracy and loss values
plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0, 1])
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')
plt.grid()

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='loss')
plt.plot(history.history['val_loss'], label = 'val_loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.grid()

plt.show()
```

```
Trial 10 Complete [00h 01m 26s]
val_accuracy: 0.5321000218391418

Best val_accuracy So Far: 0.6790000200271606
Total elapsed time: 00h 12m 50s
Epoch 1/10
1563/1563 [==============================] - 11s 6ms/step - loss: 1.5331 - accuracy: 0.4393 - val_loss: 1.2738 - val_accuracy: 0.548
Epoch 2/10
1563/1563 [==============================] - 9s 6ms/step - loss: 1.1665 - accuracy: 0.5890 - val_loss: 1.0944 - val_accuracy: 0.613
Epoch 3/10
1563/1563 [==============================] - 9s 6ms/step - loss: 1.0160 - accuracy: 0.6442 - val_loss: 1.0301 - val_accuracy: 0.645
Epoch 4/10
1563/1563 [==============================] - 9s 6ms/step - loss: 0.9102 - accuracy: 0.6825 - val_loss: 0.9874 - val_accuracy: 0.653
Epoch 5/10
1563/1563 [==============================] - 9s 6ms/step - loss: 0.8204 - accuracy: 0.7117 - val_loss: 0.9053 - val_accuracy: 0.690
Epoch 6/10
1563/1563 [==============================] - 9s 6ms/step - loss: 0.7461 - accuracy: 0.7400 - val_loss: 0.9347 - val_accuracy: 0.680
Epoch 7/10
1563/1563 [==============================] - 9s 6ms/step - loss: 0.6840 - accuracy: 0.7591 - val_loss: 0.9135 - val_accuracy: 0.698
Epoch 8/10
1563/1563 [==============================] - 9s 6ms/step - loss: 0.6277 - accuracy: 0.7801 - val_loss: 0.9196 - val_accuracy: 0.699
Epoch 9/10
1563/1563 [==============================] - 9s 6ms/step - loss: 0.5752 - accuracy: 0.7979 - val_loss: 0.9394 - val_accuracy: 0.700
Epoch 10/10
1563/1563 [==============================] - 9s 6ms/step - loss: 0.5199 - accuracy: 0.8157 - val_loss: 0.9935 - val_accuracy: 0.691
```


Training and Validation Accuracy
Accuracy
Epoch
accuracy
val_accuracy
Training and Validation Loss
Loss
Epoch
loss
val_loss