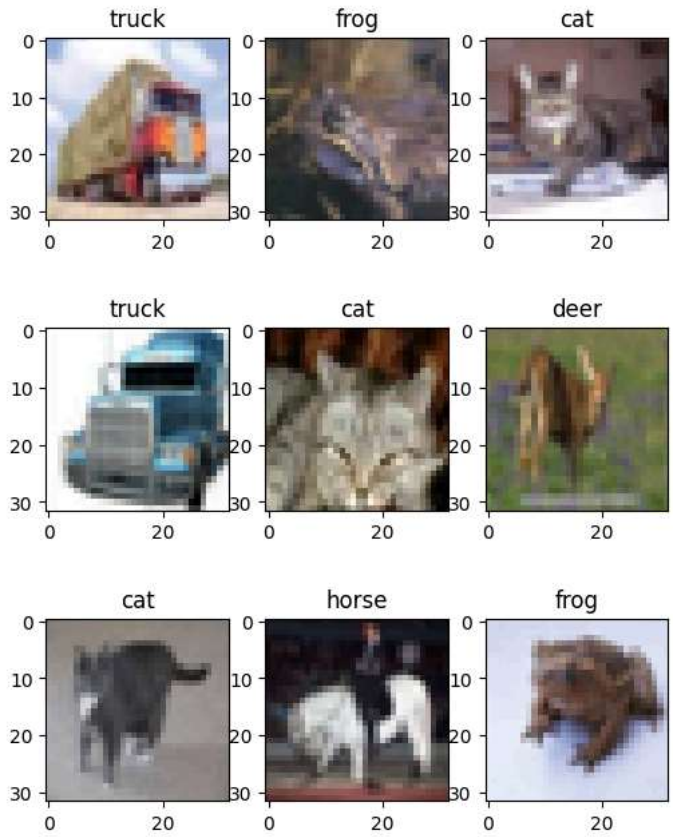
```
import os
import time
import tensorflow as tf
from tensorflow.keras.datasets import cifar10
import tensorflow_datasets as tfds
import tensorflow
from tensorflow.keras.models import Sequential
from tensorflow.keras import (models,layers,)
from matplotlib import pyplot as plt
import numpy as np
import random
from sklearn.metrics import confusion_matrix, classification_report
import matplotlib.pyplot as plt6


cifar_10_labels = {
    0: 'airplane',
    1: 'automobile',
    2: 'bird',
    3: 'cat',
    4: 'deer',
    5: 'dog',
    6: 'frog',
    7: 'horse',
    8: 'ship',
    9: 'truck'
}



model_directory = 'models'

class RandomIntegers:
    def __init__(self):
        pass

    def generate(self, n, length):
        # Generate n unique random integers between 0 and length
        return random.sample(range(length), n)

# Load the CIFAR-10 dataset
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
x_train = x_train / 255.0
x_test = x_test / 255.0
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 ━━━━━━━━━━━━━━━━━━━━ 10s 0us/step
```

```
def display_images():
    random_integers = RandomIntegers().generate(9, len(x_train))
    plt.figure(figsize=(6, 8))
    for counter, i in enumerate(random_integers):
        plt.subplot(3, 3, counter + 1)
        plt.imshow(x_train[i])
        plt.title(cifar_10_labels[y_train[i][0]])
    plt.show()

display_images()

# Define the model
model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(32, 32, 3)),
    tf.keras.layers.Dense(128, activation='relu', kernel_initializer='he_normal'),
    tf.keras.layers.Dense(10, activation='softmax')
])
```




```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/reshaping/flatten.py:37: UserWarning: Do not pass an `input_shape`/`input_dim`
  super().__init__(**kwargs)
```

```
# Compile the model
model.compile(
    optimizer=tf.keras.optimizers.Adam(0.001),
    loss=tf.keras.losses.SparseCategoricalCrossentropy(),
    metrics=[tf.keras.metrics.SparseCategoricalAccuracy()]
)

# Train the model
start_time = time.time()
history = model.fit(
    x_train, y_train,
    epochs=25,
    batch_size=512,
    validation_data=(x_test, y_test)
)
end_time = time.time()

# Print training time
total_time = end_time - start_time
print("Time taken for training: ", total_time, " seconds")

# Plot accuracy
plt.plot(history.history['sparse_categorical_accuracy'])
plt.plot(history.history['val_sparse_categorical_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()

# Plot loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
```

```
plt.show()
```

```
Epoch 1/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 6s 48ms/step - loss: 1.5250 - sparse_categorical_accuracy: 0.4644 - val_loss: 1.5334 - val_sparse_categori
Epoch 2/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 6s 61ms/step - loss: 1.4865 - sparse_categorical_accuracy: 0.4797 - val_loss: 1.5368 - val_sparse_categori
Epoch 3/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 3s 30ms/step - loss: 1.4800 - sparse_categorical_accuracy: 0.4817 - val_loss: 1.5300 - val_sparse_categori
Epoch 4/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 5s 30ms/step - loss: 1.4627 - sparse_categorical_accuracy: 0.4872 - val_loss: 1.5278 - val_sparse_categori
Epoch 5/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 4s 40ms/step - loss: 1.4645 - sparse_categorical_accuracy: 0.4869 - val_loss: 1.5628 - val_sparse_categori
Epoch 6/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 4s 40ms/step - loss: 1.4725 - sparse_categorical_accuracy: 0.4820 - val_loss: 1.5267 - val_sparse_categori
Epoch 7/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 3s 29ms/step - loss: 1.4621 - sparse_categorical_accuracy: 0.4899 - val_loss: 1.5203 - val_sparse_categori
Epoch 8/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 3s 29ms/step - loss: 1.4558 - sparse_categorical_accuracy: 0.4899 - val_loss: 1.5190 - val_sparse_categori
Epoch 9/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 3s 30ms/step - loss: 1.4640 - sparse_categorical_accuracy: 0.4899 - val_loss: 1.5387 - val_sparse_categori
Epoch 10/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 5s 50ms/step - loss: 1.4540 - sparse_categorical_accuracy: 0.4876 - val_loss: 1.5184 - val_sparse_categori
Epoch 11/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 3s 29ms/step - loss: 1.4407 - sparse_categorical_accuracy: 0.4976 - val_loss: 1.5117 - val_sparse_categori
Epoch 12/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 5s 30ms/step - loss: 1.4380 - sparse_categorical_accuracy: 0.4972 - val_loss: 1.5293 - val_sparse_categori
Epoch 13/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 4s 43ms/step - loss: 1.4429 - sparse_categorical_accuracy: 0.4934 - val_loss: 1.5063 - val_sparse_categori
Epoch 14/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 4s 30ms/step - loss: 1.4260 - sparse_categorical_accuracy: 0.5014 - val_loss: 1.5520 - val_sparse_categori
Epoch 15/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 5s 30ms/step - loss: 1.4407 - sparse_categorical_accuracy: 0.4926 - val_loss: 1.5148 - val_sparse_categori
Epoch 16/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 4s 39ms/step - loss: 1.4282 - sparse_categorical_accuracy: 0.5004 - val_loss: 1.5310 - val_sparse_categori
Epoch 17/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 4s 29ms/step - loss: 1.4381 - sparse_categorical_accuracy: 0.4941 - val_loss: 1.5158 - val_sparse_categori
Epoch 18/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 5s 32ms/step - loss: 1.4280 - sparse_categorical_accuracy: 0.5034 - val_loss: 1.5098 - val_sparse_categori
Epoch 19/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 8s 57ms/step - loss: 1.4297 - sparse_categorical_accuracy: 0.4976 - val_loss: 1.4984 - val_sparse_categori
Epoch 20/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 8s 31ms/step - loss: 1.4131 - sparse_categorical_accuracy: 0.5070 - val_loss: 1.5117 - val_sparse_categori
Epoch 21/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 7s 48ms/step - loss: 1.4096 - sparse_categorical_accuracy: 0.5068 - val_loss: 1.5072 - val_sparse_categori
Epoch 22/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 3s 31ms/step - loss: 1.4181 - sparse_categorical_accuracy: 0.5056 - val_loss: 1.4982 - val_sparse_categori
Epoch 23/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 5s 30ms/step - loss: 1.4129 - sparse_categorical_accuracy: 0.5059 - val_loss: 1.5248 - val_sparse_categori
Epoch 24/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 4s 36ms/step - loss: 1.4203 - sparse_categorical_accuracy: 0.5035 - val_loss: 1.5190 - val_sparse_categori
Epoch 25/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 4s 41ms/step - loss: 1.4156 - sparse_categorical_accuracy: 0.5038 - val_loss: 1.5003 - val_sparse_categori
Time taken for training:  117.6912784576416  seconds
```