```
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.optimizers import SGD
import random
```

```
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 ━━━━━━━━━━━━━━━━━━━━ 2s 0us/step
```

```
# Normalize the images to the range [0, 1]
x_train, x_test = x_train / 255.0, x_test / 255.0
```

```
# Convert labels to one-hot encoding
y_train = tf.keras.utils.to_categorical(y_train, 10)
y_test = tf.keras.utils.to_categorical(y_test, 10)
```

```
# Define class names for CIFAR-10
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'sh
```

```
model = Sequential([
    Flatten(input_shape=(32, 32, 3)),  # CIFAR-10 images are 32x32 with 3 channels (RGB)
    Dense(128, activation='relu'),
    Dense(64, activation='relu'),
    Dense(10, activation='softmax')
])
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/reshaping/flatten.py:37: User
  super().__init__(**kwargs)
```

```
model.compile(optimizer=SGD(),
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

```
history = model.fit(x_train, y_train,
                    epochs=20,
                    batch_size=32,
                    validation_data=(x_test, y_test))
```

```
Epoch 1/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 13s 8ms/step - accuracy: 0.2719 - loss: 2.0086 - val_ac
Epoch 2/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 18s 6ms/step - accuracy: 0.3832 - loss: 1.7382 - val_ac
Epoch 3/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 12s 7ms/step - accuracy: 0.4149 - loss: 1.6395 - val_ac
```

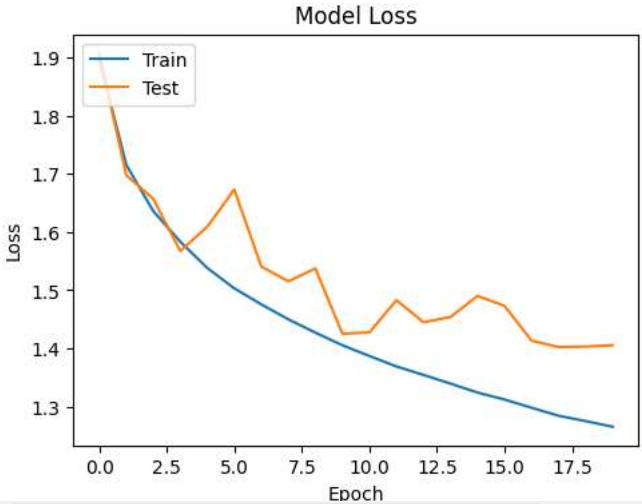
```
Epoch 4/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 26s 10ms/step - accuracy: 0.4306 - loss: 1.5924 - val_a
Epoch 5/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 15s 7ms/step - accuracy: 0.4540 - loss: 1.5477 - val_ac
Epoch 6/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 19s 6ms/step - accuracy: 0.4606 - loss: 1.5114 - val_ac
Epoch 7/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 12s 7ms/step - accuracy: 0.4774 - loss: 1.4756 - val_ac
Epoch 8/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 13s 8ms/step - accuracy: 0.4884 - loss: 1.4563 - val_ac
Epoch 9/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 20s 8ms/step - accuracy: 0.4944 - loss: 1.4280 - val_ac
Epoch 10/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 14s 9ms/step - accuracy: 0.5010 - loss: 1.4065 - val_ac
Epoch 11/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 10s 6ms/step - accuracy: 0.5055 - loss: 1.3942 - val_ac
Epoch 12/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 11s 7ms/step - accuracy: 0.5141 - loss: 1.3629 - val_ac
Epoch 13/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 11s 7ms/step - accuracy: 0.5185 - loss: 1.3552 - val_ac
Epoch 14/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 19s 6ms/step - accuracy: 0.5271 - loss: 1.3347 - val_ac
Epoch 15/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 11s 7ms/step - accuracy: 0.5305 - loss: 1.3171 - val_ac
Epoch 16/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 24s 9ms/step - accuracy: 0.5353 - loss: 1.3102 - val_ac
Epoch 17/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 20s 8ms/step - accuracy: 0.5454 - loss: 1.2890 - val_ac
Epoch 18/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 11s 7ms/step - accuracy: 0.5489 - loss: 1.2787 - val_ac
Epoch 19/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 13s 8ms/step - accuracy: 0.5486 - loss: 1.2726 - val_ac
Epoch 20/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 12s 8ms/step - accuracy: 0.5534 - loss: 1.2637 - val_ac
```

```
test_loss, test_acc = model.evaluate(x_test, y_test)
print(f'Test Loss: {test_loss}')
print(f'Test Accuracy: {test_acc}')
```

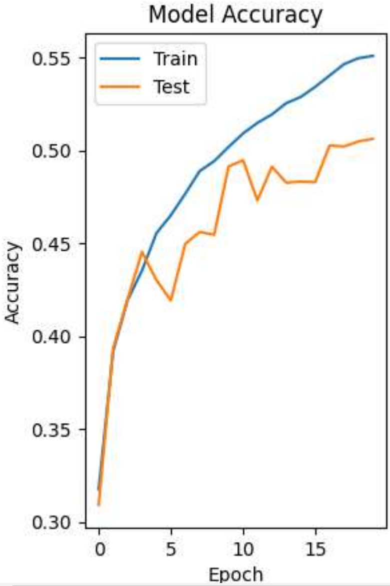
```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.5167 - loss: 1.3957
Test Loss: 1.4053822755813599
Test Accuracy: 0.5060999989509583
```

```
plt.figure(figsize=(12, 4))

# Plot loss
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
```

<matplotlib.legend.Legend at 0x7e3d2efe8f10>




```
# Plot accuracy
plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')

plt.show()
```




```
# Plot one training image
plt.figure(figsize=(4, 4))
plt.imshow(x_train[0])
plt.title(f'Train Image: {class_names[np.argmax(y_train[0])]}')
plt.axis('off')
plt.show()
```

```
# Plot one testing image
n = random.randint(0, len(x_test) - 1)
plt.figure(figsize=(4, 4))
plt.imshow(x_test[n])
plt.title(f'Test Image: {class_names[np.argmax(y_test[n])]}')
plt.axis('off')
plt.show()
```

Test Image: automobile