```
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()


#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)


class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'h



model = Sequential([
 Flatten(input_shape=(32,32,3)),
 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: Use
  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 ━━━━━━━━━━━━━━━━━━━━ 10s 6ms/step - accuracy: 0.1042 - loss: 2.3023 - val_a
Epoch 2/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 9s 5ms/step - accuracy: 0.1057 - loss: 2.3020 - val_ac
Epoch 3/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 11s 6ms/step - accuracy: 0.1088 - loss: 2.3017 - val_a
Epoch 4/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 7s 5ms/step - accuracy: 0.1136 - loss: 2.3015 - val_ac
```

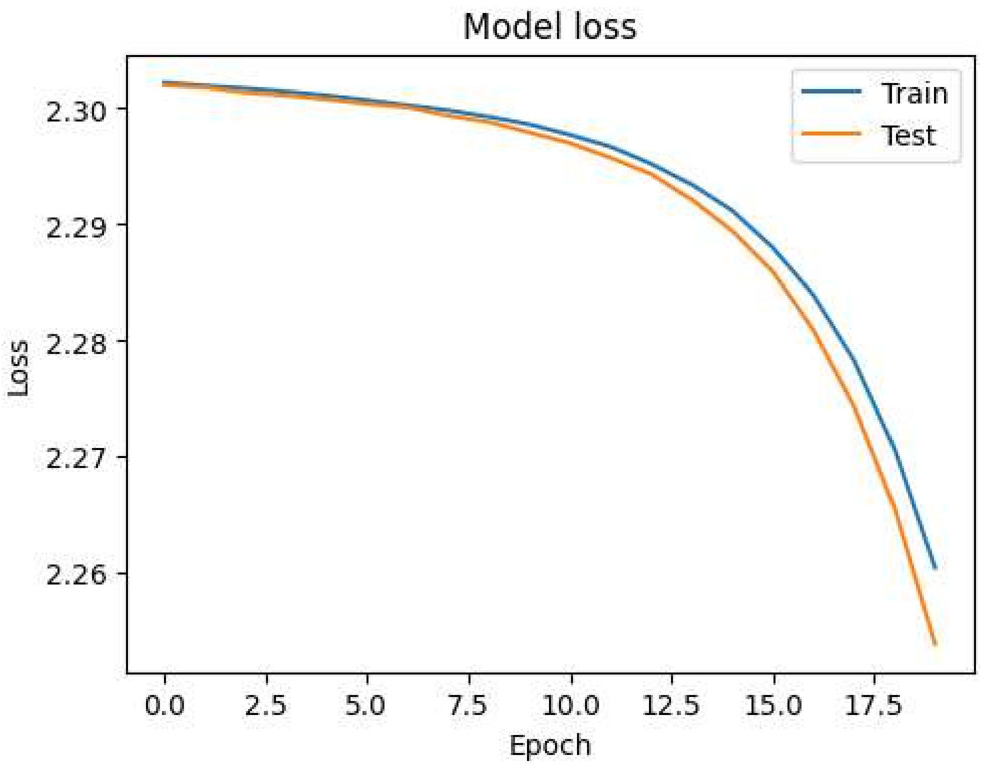
```
Epoch 5/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 9s 6ms/step - accuracy: 0.1143 - loss: 2.3011 - val_ac
Epoch 6/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 8s 5ms/step - accuracy: 0.1198 - loss: 2.3007 - val_ac
Epoch 7/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 10s 5ms/step - accuracy: 0.1158 - loss: 2.3003 - val_a
Epoch 8/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 9s 6ms/step - accuracy: 0.1163 - loss: 2.3000 - val_ac
Epoch 9/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 10s 6ms/step - accuracy: 0.1410 - loss: 2.2994 - val_a
Epoch 10/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 8s 5ms/step - accuracy: 0.1264 - loss: 2.2988 - val_ac
Epoch 11/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 10s 5ms/step - accuracy: 0.1462 - loss: 2.2979 - val_a
Epoch 12/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 11s 5ms/step - accuracy: 0.1468 - loss: 2.2971 - val_a
Epoch 13/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 9s 6ms/step - accuracy: 0.1446 - loss: 2.2957 - val_ac
Epoch 14/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 7s 5ms/step - accuracy: 0.1537 - loss: 2.2939 - val_ac
Epoch 15/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 11s 5ms/step - accuracy: 0.1620 - loss: 2.2919 - val_a
Epoch 16/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 12s 6ms/step - accuracy: 0.1491 - loss: 2.2885 - val_a
Epoch 17/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 10s 6ms/step - accuracy: 0.1659 - loss: 2.2848 - val_a
Epoch 18/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 7s 5ms/step - accuracy: 0.1592 - loss: 2.2801 - val_ac
Epoch 19/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 9s 6ms/step - accuracy: 0.1634 - loss: 2.2728 - val_ac
Epoch 20/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 8s 5ms/step - accuracy: 0.1675 - loss: 2.2634 - val_ac
```

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

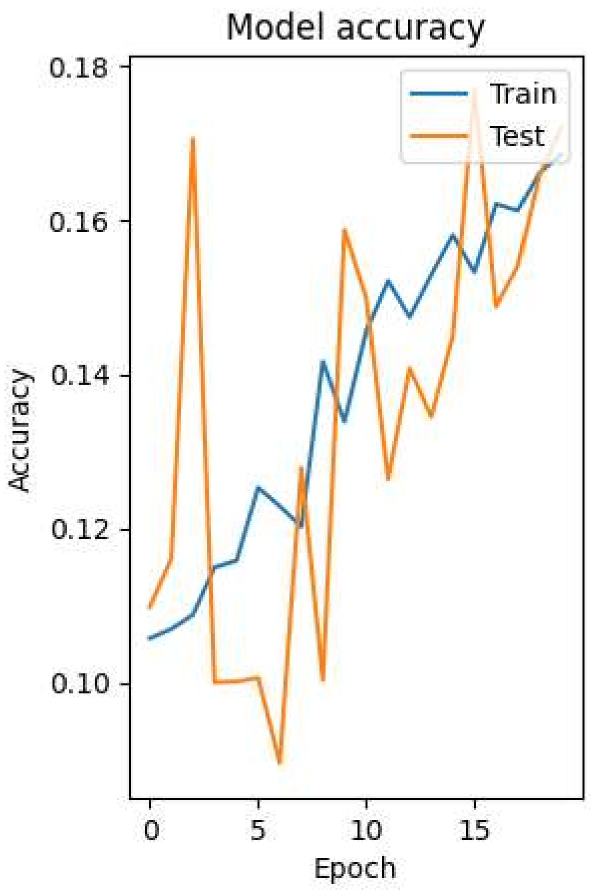
```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.1768 - loss: 2.2532
Test Loss: 2.253901958465576
Test Accuracy: 0.17190000414848328
```

```python
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 right')
```

```
<matplotlib.legend.Legend at 0x7ebf99ed5ba0>
```



```
#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 right')
plt.show()
```



```
n = 110  # Set the index to display a different image, change this value to test differen

plt.figure(figsize=(4, 4))
plt.imshow(x_train[n] * 255)  # Rescale the image back to [0, 255] for proper display
plt.title(f'Train Image: {class_names[np.argmax(y_train[n])]}')  # Use the same index 'n'
plt.axis('off')
plt.show()
```

```
n = random.randint(0, len(x_test) - 1)
```

```
    plt.figure(figsize=(4, 4))
    plt.imshow(x_test[n] * 255)  # Rescale the image back to [0, 255] for proper disp
    plt.title(f'Test Image: {class_names[np.argmax(y_test[n])]}')
    plt.axis('off')
    plt.show()
```

Test Image: bird