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

```
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
x_train = x_train / 255
x_test = x_test / 255
```

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

```
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']
```

```
# Define the model
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: UserWarning: Do not pass an `input_shape`/`input_dim`
  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 ━━━━━━━━━━━━━━━━━━━━ 11s 7ms/step - accuracy: 0.4604 - loss: 1.5190 - val_accuracy: 0.4449 - val_loss: 1.5492
Epoch 2/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 11s 7ms/step - accuracy: 0.4702 - loss: 1.4943 - val_accuracy: 0.4024 - val_loss: 1.6950
Epoch 3/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 9s 6ms/step - accuracy: 0.4819 - loss: 1.4650 - val_accuracy: 0.4677 - val_loss: 1.4987
Epoch 4/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 10s 5ms/step - accuracy: 0.4884 - loss: 1.4478 - val_accuracy: 0.4805 - val_loss: 1.4609
Epoch 5/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 14s 8ms/step - accuracy: 0.4945 - loss: 1.4198 - val_accuracy: 0.4593 - val_loss: 1.4932
Epoch 6/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 19s 7ms/step - accuracy: 0.5055 - loss: 1.4049 - val_accuracy: 0.4876 - val_loss: 1.4380
Epoch 7/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 20s 7ms/step - accuracy: 0.5118 - loss: 1.3790 - val_accuracy: 0.4714 - val_loss: 1.4802
Epoch 8/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 19s 6ms/step - accuracy: 0.5194 - loss: 1.3624 - val_accuracy: 0.4659 - val_loss: 1.4784
Epoch 9/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 9s 6ms/step - accuracy: 0.5198 - loss: 1.3487 - val_accuracy: 0.4754 - val_loss: 1.4686
Epoch 10/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 11s 6ms/step - accuracy: 0.5290 - loss: 1.3331 - val_accuracy: 0.4528 - val_loss: 1.5363
Epoch 11/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 11s 7ms/step - accuracy: 0.5361 - loss: 1.3111 - val_accuracy: 0.5074 - val_loss: 1.3918
Epoch 12/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 11s 7ms/step - accuracy: 0.5371 - loss: 1.3001 - val_accuracy: 0.4961 - val_loss: 1.4387
Epoch 13/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 9s 6ms/step - accuracy: 0.5512 - loss: 1.2792 - val_accuracy: 0.4662 - val_loss: 1.5232
Epoch 14/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 11s 7ms/step - accuracy: 0.5513 - loss: 1.2743 - val_accuracy: 0.4912 - val_loss: 1.4415
Epoch 15/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 20s 7ms/step - accuracy: 0.5549 - loss: 1.2620 - val_accuracy: 0.4989 - val_loss: 1.4511
Epoch 16/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 9s 6ms/step - accuracy: 0.5575 - loss: 1.2490 - val_accuracy: 0.5058 - val_loss: 1.4045
Epoch 17/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 11s 7ms/step - accuracy: 0.5613 - loss: 1.2456 - val_accuracy: 0.5063 - val_loss: 1.3928
Epoch 18/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 19s 6ms/step - accuracy: 0.5617 - loss: 1.2316 - val_accuracy: 0.4815 - val_loss: 1.4673
Epoch 19/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 9s 6ms/step - accuracy: 0.5727 - loss: 1.2118 - val_accuracy: 0.5094 - val_loss: 1.4086
Epoch 20/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 11s 7ms/step - accuracy: 0.5748 - loss: 1.2103 - val_accuracy: 0.5111 - val_loss: 1.3756
```

```
test_loss, test_acc = model.evaluate(x_test, y_test)
print("Loss = %.3f" % test_loss)
print("Accuracy = %.3f" % test_acc)
```

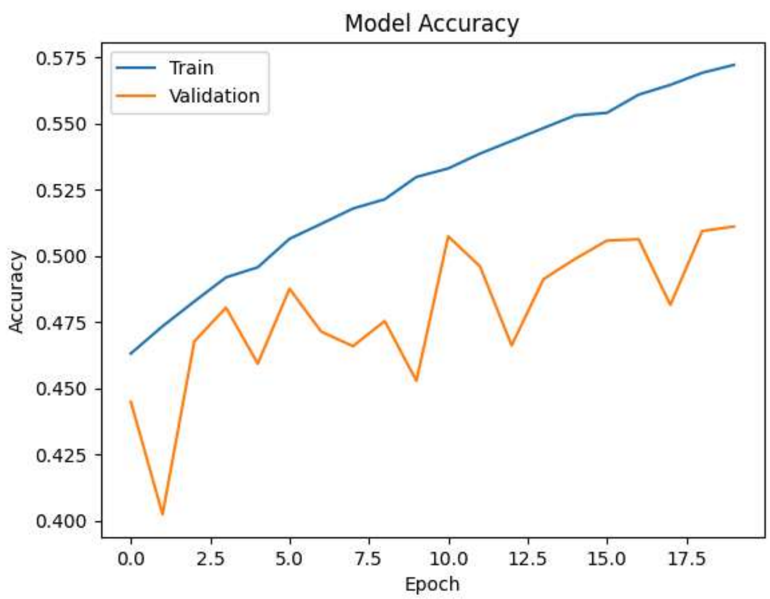
```
313/313 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - accuracy: 0.5211 - loss: 1.3691
Loss = 1.376
Accuracy = 0.511
```

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

Test Image: frog

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

```
<matplotlib.legend.Legend at 0x78bda05629b0>
```




```
#for printing loss just replace each acuuracy with loss word
```