In [1]:
```
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
```

```
C:\Users\hpcnd\anaconda3\lib\site-packages\scipy\__init__.py:146: UserWarning: A N
umPy version >=1.16.5 and <1.23.0 is required for this version of SciPy (detected
version 1.26.4
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

In [2]:
```
(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 ━━━━━━━━━━━━━━━━━━━━ 110s 1us/step
```

In [4]:
```
#normalize the images to the range [0,1]
x_train, x_test = x_train / 255.0, x_test / 255.0
```

In [5]:
```
#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)
```

In [6]:
```
#define class names for CIFAR-10
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'hor
```

In [8]:
```
model = Sequential([
    Flatten(input_shape=(32,32,3)),   #cifar-10 imgs are 32*32 with 3 channels(RGB)
    Dense(128, activation='relu'),
    Dense(64, activation='relu'),
    Dense(10, activation='softmax')
])
```

```
C:\Users\hpcnd\anaconda3\lib\site-packages\keras\src\layers\reshaping\flatten.py:3
7: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first layer
in the model instead.
  super().__init__(**kwargs)
```

In [11]:
```
model.compile(optimizer=SGD(),
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

In [12]:
```
history = model.fit(x_train, y_train,
                    epochs=20,
                    batch_size=32,
                    validation_data=(x_test, y_test))
```

```
Epoch 1/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.2699 - loss: 2.0051 - val
_accuracy: 0.3342 - val_loss: 1.8434
Epoch 2/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.3801 - loss: 1.7435 - val
_accuracy: 0.4240 - val_loss: 1.6395
Epoch 3/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.4164 - loss: 1.6364 - val
_accuracy: 0.4169 - val_loss: 1.6265
Epoch 4/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.4381 - loss: 1.5865 - val
_accuracy: 0.4235 - val_loss: 1.6379
Epoch 5/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 6s 4ms/step - accuracy: 0.4566 - loss: 1.5372 - val
_accuracy: 0.4508 - val_loss: 1.5362
Epoch 6/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 6s 4ms/step - accuracy: 0.4640 - loss: 1.5035 - val
_accuracy: 0.4579 - val_loss: 1.5216
Epoch 7/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.4760 - loss: 1.4708 - val
_accuracy: 0.4670 - val_loss: 1.4929
Epoch 8/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.4881 - loss: 1.4458 - val
_accuracy: 0.4694 - val_loss: 1.4941
Epoch 9/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.4955 - loss: 1.4205 - val
_accuracy: 0.4850 - val_loss: 1.4481
Epoch 10/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 6s 4ms/step - accuracy: 0.4997 - loss: 1.4074 - val
_accuracy: 0.4501 - val_loss: 1.5299
Epoch 11/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.5107 - loss: 1.3814 - val
_accuracy: 0.4687 - val_loss: 1.4860
Epoch 12/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 6s 4ms/step - accuracy: 0.5132 - loss: 1.3700 - val
_accuracy: 0.4424 - val_loss: 1.5705
Epoch 13/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 6s 4ms/step - accuracy: 0.5213 - loss: 1.3550 - val
_accuracy: 0.4437 - val_loss: 1.6226
Epoch 14/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 6s 4ms/step - accuracy: 0.5289 - loss: 1.3304 - val
_accuracy: 0.4888 - val_loss: 1.4365
Epoch 15/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.5303 - loss: 1.3255 - val
_accuracy: 0.4989 - val_loss: 1.4002
Epoch 16/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 6s 4ms/step - accuracy: 0.5384 - loss: 1.3004 - val
_accuracy: 0.5005 - val_loss: 1.4093
Epoch 17/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 6s 4ms/step - accuracy: 0.5438 - loss: 1.2901 - val
_accuracy: 0.5009 - val_loss: 1.4142
Epoch 18/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.5444 - loss: 1.2789 - val
_accuracy: 0.5031 - val_loss: 1.4086
Epoch 19/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 6s 4ms/step - accuracy: 0.5510 - loss: 1.2629 - val
_accuracy: 0.4870 - val_loss: 1.4644
Epoch 20/20
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 6s 4ms/step - accuracy: 0.5543 - loss: 1.2509 - val
_accuracy: 0.5069 - val_loss: 1.3958
```

In [13]:

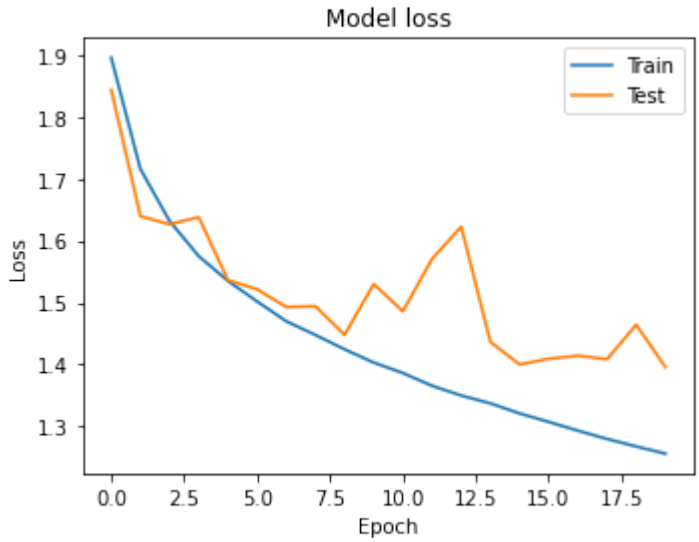
```
test_loss, test_acc = model.evaluate(x_test, y_test)
print(f'Test Loss: {test_loss}')
print(f'Test Accuracy: {test_acc}')
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.5128 - loss: 1.3852
Test Loss: 1.3957630395889282
Test Accuracy: 0.5069000124931335
```

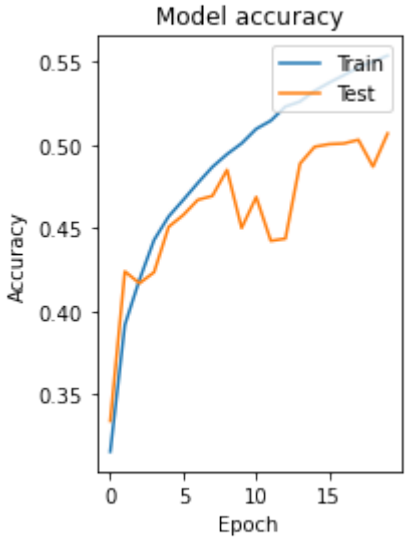
In [14]:
```
plt.figure(figsize=(12, 4))

#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')
```

Out[14]:
```
<matplotlib.legend.Legend at 0x1465d15d5b0>
```

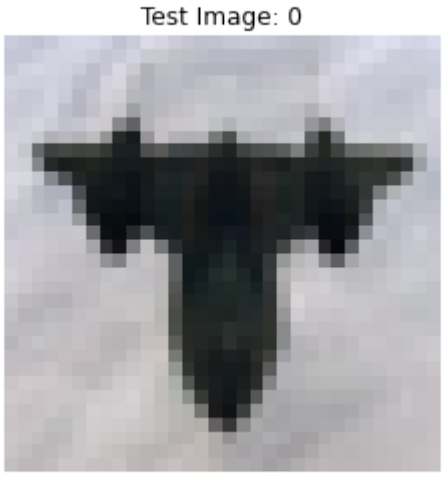


In [15]:
```
#plot accuracy
plt.subplot(1,2,2)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train','Test'],loc='upper right')
plt.show()
```



In [16]:
```
#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()
```

In [25]:
```
#plot one testing img
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()
```



In [ ]:

In [ ]: