The Data

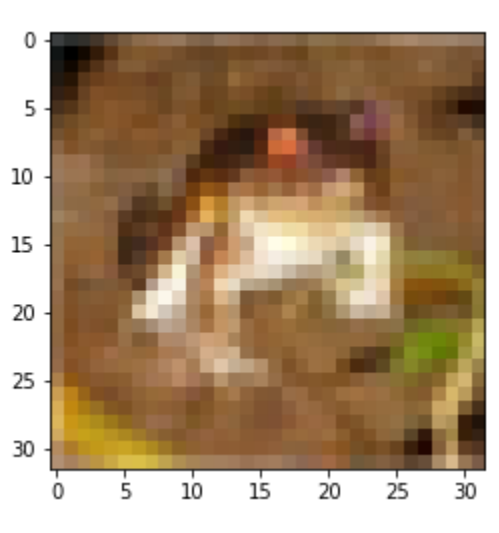
In [1]:
```python
from tensorflow.keras.datasets import cifar10

(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 [==============================] - 13s 0us/step
```

In [2]:
```python
x_train.shape
```
Out[2]:
```
(50000, 32, 32, 3)
```

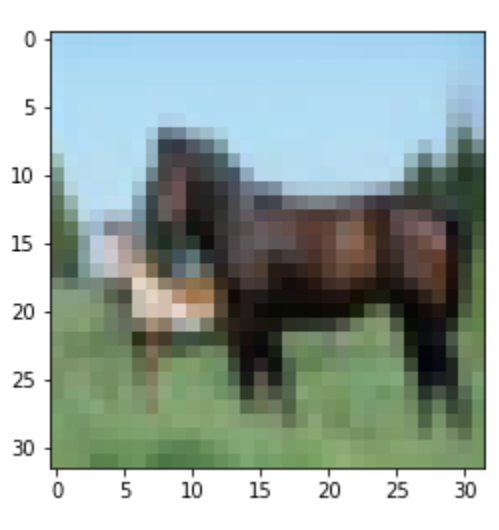
In [3]:
```python
x_train[0].shape
```
Out[3]:
```
(32, 32, 3)
```

In [4]:
```python
import matplotlib.pyplot as plt
```

In [5]:
```python
# FROG
plt.imshow(x_train[0])
```
Out[5]:
```
<matplotlib.image.AxesImage at 0x7f0f659cb1d0>
```



In [6]:
```python
# HORSE
plt.imshow(x_train[12])
```
Out[6]:
```
<matplotlib.image.AxesImage at 0x7f0f65494850>
```



PreProcessing

In [7]:
```python
x_train[0]
```
Out[7]:
```
array([[[ 59,  62,  63],
        [ 43,  46,  45],
        [ 50,  48,  43],
        ...,
        [158, 132, 108],
        [152, 125, 102],
        [148, 124, 103]],

       [[ 16,  20,  20],
        [  0,   0,   0],
        [ 18,   8,   0],
        ...,
        [123,  88,  55],
        [119,  83,  50],
        [122,  87,  57]],

       [[ 25,  24,  21],
        [ 16,   7,   0],
        [ 49,  27,   8],
        ...,
        [118,  84,  50],
        [120,  84,  50],
        [109,  73,  42]],

       ...,

       [[208, 170,  96],
        [201, 153,  34],
        [198, 161,  26],
        ...,
        [160, 133,  70],
        [ 56,  31,   7],
        [ 53,  34,  20]],

       [[180, 139,  96],
        [173, 123,  42],
        [186, 144,  30],
        ...,
        [184, 148,  94],
        [ 97,  62,  34],
        [ 83,  53,  34]],

       [[177, 144, 116],
        [168, 129,  94],
        [179, 142,  87],
        ...,
        [216, 184, 140],
        [151, 118,  84],
        [123,  92,  72]]], dtype=uint8)
```

In [8]:
```python
x_train[0].shape
```
Out[8]:
```
(32, 32, 3)
```

In [9]:
```python
x_train.max()
```
Out[9]:
```
255
```

In [10]:
```python
x_train = x_train / 225
```

In [11]:
```python
x_test = x_test / 255
```

In [12]:
```python
x_train.shape
```
Out[12]:
```
(50000, 32, 32, 3)
```

In [13]:
```python
x_test.shape
```
Out[13]:
```
(10000, 32, 32, 3)
```

Labels

In [14]:
```python
from tensorflow.keras.utils import to_categorical
```

In [15]:
```python
y_train.shape
```
Out[15]:
```
(50000, 1)
```

In [16]:
```python
y_train[0]
```
Out[16]:
```
array([6], dtype=uint8)
```

In [17]:
```python
y_cat_train = to_categorical(y_train, 10)
```

In [18]:
```python
y_cat_train.shape
```
Out[18]:
```
(50000, 10)
```

In [19]:
```python
y_cat_train[0]
```
Out[19]:
```
array([0., 0., 0., 0., 0., 0., 1., 0., 0., 0.], dtype=float32)
```

In [20]:
```python
y_cat_test = to_categorical(y_test, 10)
```

Building the Model

In [21]:
```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, MaxPool2D, Flatten
```

In [22]:
```python
model = Sequential()

## FIRST SET OF LAYERS

# CONVOLUTIONAL LAYER
model.add(Conv2D(filters=32, kernel_size=(4, 4),input_shape=(32, 32, 3), activation='relu',))
# POOLING LAYER
model.add(MaxPool2D(pool_size=(2, 2)))

## SECOND SET OF LAYERS

# CONVOLUTIONAL LAYER
model.add(Conv2D(filters=32, kernel_size=(4, 4),input_shape=(32, 32, 3), activation='relu',))
# POOLING LAYER
model.add(MaxPool2D(pool_size=(2, 2)))

# FLATTEN IMAGES FROM 28 by 28 to 764 BEFORE FINAL LAYER
model.add(Flatten())

# 256 NEURONS IN DENSE HIDDEN LAYER (YOU CAN CHANGE THIS NUMBER OF NEURONS)
model.add(Dense(256, activation='relu'))

# LAST LAYER IS THE CLASSIFIER, THUS 10 POSSIBLE CLASSES
model.add(Dense(10, activation='softmax'))


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

In [23]:
```python
model.summary()
```
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 29, 29, 32)        1568

 max_pooling2d (MaxPooling2D  (None, 14, 14, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 11, 11, 32)        16416

 max_pooling2d_1 (MaxPooling  (None, 5, 5, 32)         0
 2D)

 flatten (Flatten)           (None, 800)               0

 dense (Dense)               (None, 256)               205056

 dense_1 (Dense)             (None, 10)                2570

=================================================================
Total params: 225,610
Trainable params: 225,610
Non-trainable params: 0
_________________________________________________________________
```

In [24]:
```python
from tensorflow.keras.callbacks import EarlyStopping
```

In [25]:
```python
early_stop = EarlyStopping(monitor='val_loss', patience=3)
```

In [26]:
```python
model.fit(x_train,
          y_cat_train,
          epochs=15,
          validation_data=(x_test, y_cat_test),
          callbacks=[early_stop])
```
```
Epoch 1/15
1563/1563 [==============================] - 14s 5ms/step - loss: 1.5250 - accuracy: 0.4512 - val_loss: 1.3580 - val_
accuracy: 0.5182
Epoch 2/15
1563/1563 [==============================] - 8s 5ms/step - loss: 1.1751 - accuracy: 0.5873 - val_loss: 1.1128 - val_a
ccuracy: 0.6082
Epoch 3/15
1563/1563 [==============================] - 7s 5ms/step - loss: 1.0162 - accuracy: 0.6464 - val_loss: 1.2543 - val_a
ccuracy: 0.5785
Epoch 4/15
1563/1563 [==============================] - 7s 5ms/step - loss: 0.9137 - accuracy: 0.6851 - val_loss: 1.0203 - val_a
ccuracy: 0.6465
Epoch 5/15
1563/1563 [==============================] - 7s 5ms/step - loss: 0.8324 - accuracy: 0.7142 - val_loss: 0.9758 - val_a
ccuracy: 0.6677
Epoch 6/15
1563/1563 [==============================] - 7s 5ms/step - loss: 0.7716 - accuracy: 0.7365 - val_loss: 0.9850 - val_a
ccuracy: 0.6682
Epoch 7/15
1563/1563 [==============================] - 8s 5ms/step - loss: 0.7125 - accuracy: 0.7574 - val_loss: 1.0369 - val_a
ccuracy: 0.6588
Epoch 8/15
1563/1563 [==============================] - 8s 5ms/step - loss: 0.6688 - accuracy: 0.7747 - val_loss: 1.0072 - val_a
ccuracy: 0.6857
```
Out[26]:
```
<keras.callbacks.History at 0x7f0f6541da90>
```

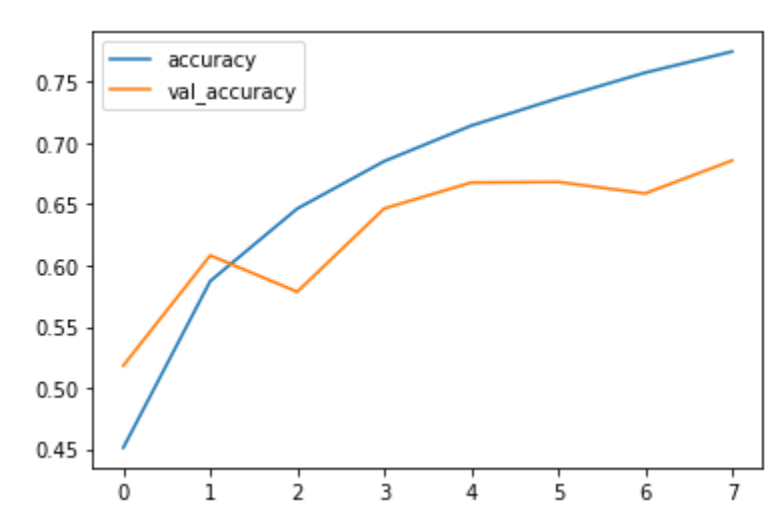
In [27]:
```python
import pandas as pd
import numpy as np
```

In [28]:
```python
losses = pd.DataFrame(model.history.history)
```

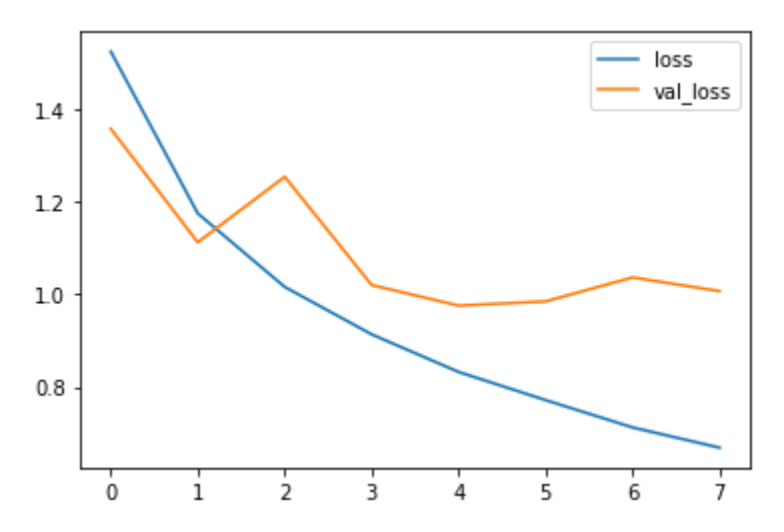
In [29]:
```python
losses.head()
```
Out[29]:

	loss	accuracy	val_loss	val_accuracy
0	1.525001	0.45116	1.358043	0.5182
1	1.175126	0.58726	1.112785	0.6082
2	1.016194	0.64640	1.254319	0.5785
3	0.913704	0.68512	1.020349	0.6465
4	0.832401	0.71416	0.975840	0.6677

In [30]:
```python
losses[['accuracy', 'val_accuracy']].plot()
```
Out[30]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0f0021afd0>
```



In [31]:
```python
losses[['loss', 'val_loss']].plot()
```
Out[31]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0f001c6c10>
```



In [32]:
```python
model.metrics_names
```
Out[32]:
```
['loss', 'accuracy']
```

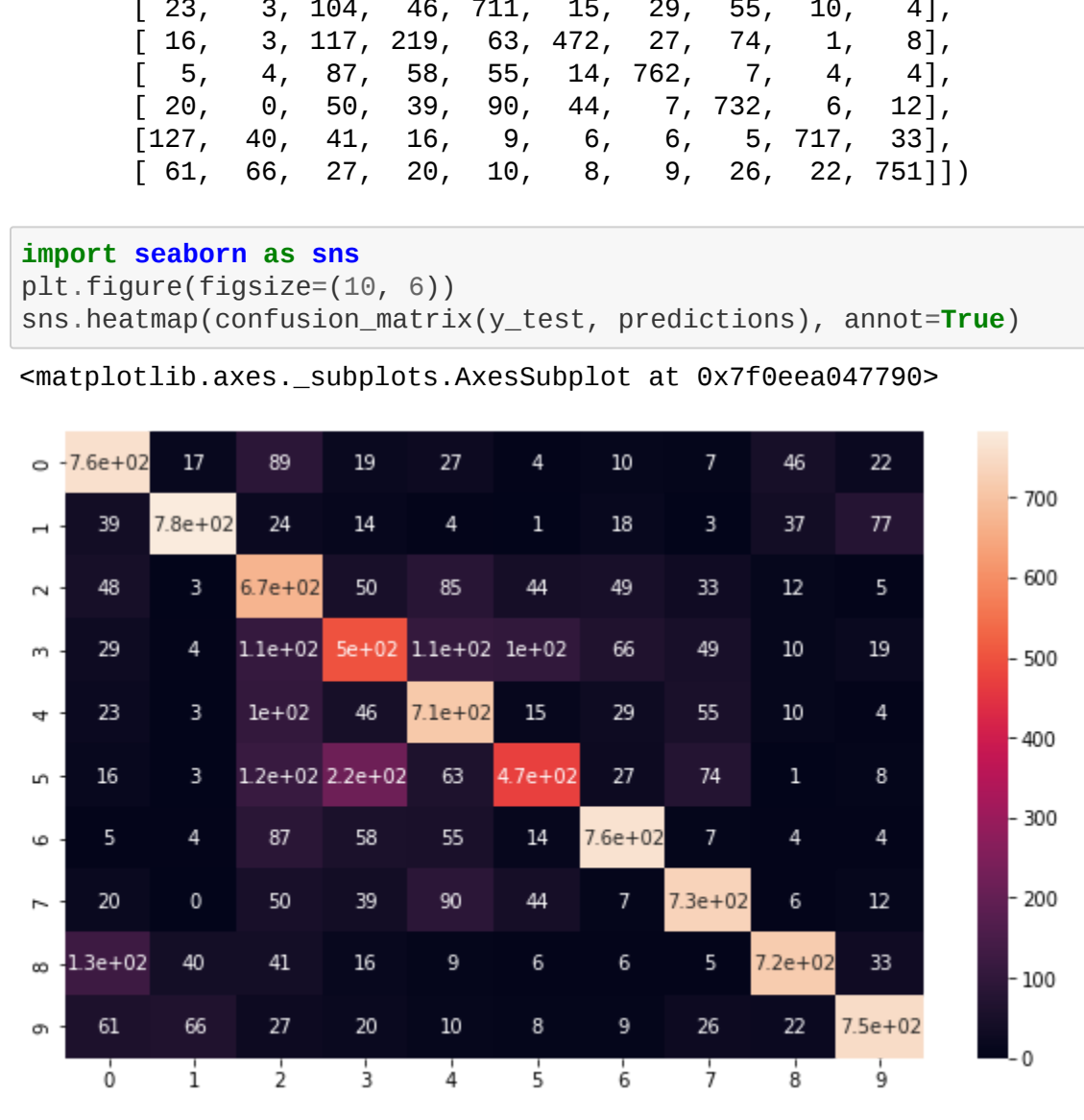
In [33]:
```python
print(model.metrics_names)
print(model.evaluate(x_test, y_cat_test, verbose=0))
```
```
['loss', 'accuracy']
[1.0071938037872314, 0.685699999332428]
```

In [34]:
```python
from sklearn.metrics import classification_report, confusion_matrix

# predictions = model.predict_classes(x_test)
predictions = model.predict(x_test).argmax(axis=1).astype('uint8')
```
```
313/313 [==============================] - 1s 2ms/step
```

In [35]:
```python
print(classification_report(y_test, predictions))
```
```
              precision    recall  f1-score   support

           0       0.67      0.76      0.71      1000
           1       0.85      0.78      0.81      1000
           2       0.51      0.67      0.58      1000
           3       0.51      0.50      0.50      1000
           4       0.61      0.71      0.66      1000
           5       0.66      0.47      0.55      1000
           6       0.78      0.76      0.77      1000
           7       0.74      0.73      0.74      1000
           8       0.83      0.72      0.77      1000
           9       0.80      0.75      0.78      1000

    accuracy                           0.69     10000
   macro avg       0.70      0.69      0.69     10000
weighted avg       0.70      0.69      0.69     10000
```

In [36]:
```python
confusion_matrix(y_test, predictions)
```
Out[36]:
```
array([[759,  17,  89,  19,  27,   4,  10,   7,  46,  22],
       [ 39, 783,  24,  14,   4,   1,  18,   3,  37,  77],
       [ 48,   3, 671,  50,  85,  44,  49,  33,  12,   5],
       [ 29,   4, 113, 499, 108, 103,  66,  49,  10,  19],
       [ 23,   3, 104,  46, 711,  15,  29,  55,  10,   4],
       [ 16,   3, 117, 219,  63, 472,  27,  74,   1,   8],
       [  5,   4,  87,  58,  55,  14, 762,   7,   4,   4],
       [ 20,   0,  50,  39,  90,  44,   7, 732,   6,  12],
       [127,  40,  41,  16,   9,   6,   6,   5, 717,  33],
       [ 61,  66,  27,  20,  10,   8,   9,  26,  22, 751]])
```

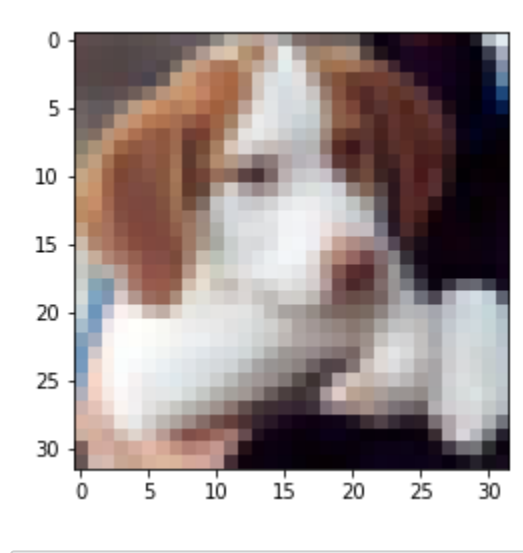
In [37]:
```python
import seaborn as sns
plt.figure(figsize=(10, 6))
sns.heatmap(confusion_matrix(y_test, predictions), annot=True)
```
Out[37]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0eea047790>
```



Predicting a given image

In [38]:
```python
my_image = x_test[16]
```

In [39]:
```python
plt.imshow(my_image)
```
Out[39]:
```
<matplotlib.image.AxesImage at 0x7f0ea9caedd0>
```



In [40]:
```python
model.predict(my_image.reshape(1, 32, 32, 3)).argmax(axis=1)
```
```
1/1 [==============================] - 0s 58ms/step
```
Out[40]:
```
array([5])
```

In [41]:
```python
# 5 is DOG
```