In [1]:
```
# import relevant libraries
import numpy as np
import matplotlib.pyplot as plt
import keras.layers as l
import keras.optimizers as o
import keras.models as m
import keras
from keras.utils import to_categorical
```

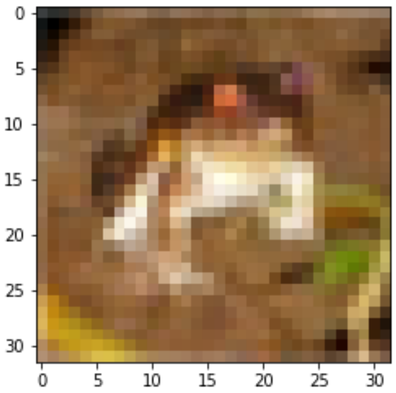
```
Using TensorFlow backend.
```

In [3]:
```
# load cifar10 dataset
from keras.datasets import cifar10

(x_train,y_train), (x_test,y_test) = cifar10.load_data()
# rescale pixel values
x_train = (1/255.0)*x_train
x_test = (1/255.0)*x_test
# labels -> one-hot encodings
y_train = to_categorical(y_train.reshape([-1, 1]))
y_test = to_categorical(y_test.reshape([-1, 1]))
```

In [4]:
```
# for jupyter notebook (ignore)
%matplotlib inline
```

In [5]:
```
# plot image
plt.imshow(x_train[0])
```

Out[5]:
```
<matplotlib.image.AxesImage at 0x7ff7359c2ef0>
```



Sequential API

In [6]:

```
# define model using sequential api
model = m.Sequential([
    l.Conv2D(32, (3,3), padding='same', activation='relu', input_shape=(32,
    l.MaxPooling2D((3,3)),
    l.Conv2D(64, (3,3), padding='same', activation='relu'),
    l.MaxPooling2D((3,3)),
    l.Flatten(),
    l.Dense(32, activation='relu'),
    l.Dense(10, activation="softmax")
])

# print a model summary
model.summary()
```

```
WARNING: Logging before flag parsing goes to stderr.
W1230 13:55:03.580109 140701437798144 deprecation_wrapper.py:119] From /ho
me/gpik/miniconda3/lib/python3.7/site-packages/keras/backend/tensorflow_ba
ckend.py:74: The name tf.get_default_graph is deprecated. Please use tf.co
mpat.v1.get_default_graph instead.

W1230 13:55:03.607579 140701437798144 deprecation_wrapper.py:119] From /ho
me/gpik/miniconda3/lib/python3.7/site-packages/keras/backend/tensorflow_ba
ckend.py:517: The name tf.placeholder is deprecated. Please use tf.compat.
v1.placeholder instead.

W1230 13:55:03.611763 140701437798144 deprecation_wrapper.py:119] From /ho
me/gpik/miniconda3/lib/python3.7/site-packages/keras/backend/tensorflow_ba
ckend.py:4138: The name tf.random_uniform is deprecated. Please use tf.ran
dom.uniform instead.

W1230 13:55:03.630030 140701437798144 deprecation_wrapper.py:119] From /ho
me/gpik/miniconda3/lib/python3.7/site-packages/keras/backend/tensorflow_ba
ckend.py:3976: The name tf.nn.max_pool is deprecated. Please use tf.nn.max
_pool2d instead.


_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 32, 32, 32)        896
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 10, 10, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 10, 10, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 3, 3, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 576)               0
_________________________________________________________________
dense_1 (Dense)              (None, 32)                18464
_________________________________________________________________
dense_2 (Dense)              (None, 10)                330
=================================================================
Total params: 38,186
Trainable params: 38,186
Non-trainable params: 0
_________________________________________________________________
```

In [7]:

```
# compile a model
model.compile(optimizer=o.Adam(),
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

```
W1230 13:55:03.697125 140701437798144 deprecation_wrapper.py:119] From /ho
me/gpik/miniconda3/lib/python3.7/site-packages/keras/optimizers.py:790: Th
e name tf.train.Optimizer is deprecated. Please use tf.compat.v1.train.Opt
imizer instead.

W1230 13:55:03.705946 140701437798144 deprecation_wrapper.py:119] From /ho
me/gpik/miniconda3/lib/python3.7/site-packages/keras/backend/tensorflow_ba
ckend.py:3295: The name tf.log is deprecated. Please use tf.math.log inste
ad.
```

In [8]:

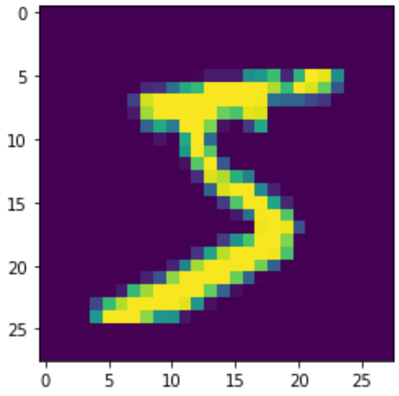
```
# fit model to data
model.fit(x_train,y_train, epochs=10, validation_data = (x_test,y_test))
```

```
W1230 13:55:03.814929 140701437798144 deprecation.py:323] From /home/gpik/
miniconda3/lib/python3.7/site-packages/tensorflow/python/ops/math_grad.py:
1250: add_dispatch_support.<locals>.wrapper (from tensorflow.python.ops.ar
ray_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
W1230 13:55:03.862976 140701437798144 deprecation_wrapper.py:119] From /ho
me/gpik/miniconda3/lib/python3.7/site-packages/keras/backend/tensorflow_ba
ckend.py:986: The name tf.assign_add is deprecated. Please use tf.compat.v
1.assign_add instead.


Train on 50000 samples, validate on 10000 samples
Epoch 1/10
50000/50000 [==============================] - 15s 306us/step - loss: 1.57
07 - acc: 0.4331 - val_loss: 1.2921 - val_acc: 0.5419
Epoch 2/10
50000/50000 [==============================] - 14s 290us/step - loss: 1.18
55 - acc: 0.5826 - val_loss: 1.1453 - val_acc: 0.5940
Epoch 3/10
```

Functional API

In [9]:
```
from keras.datasets import mnist

# load mnist data
(x_train,y_train), (x_test,y_test) = mnist.load_data()
# rescale pixel values
x_train = (1/255.0)*x_train
x_test = (1/255.0)*x_test
# labels -> one-hot encodings
y_train = to_categorical(y_train.reshape([-1, 1]))
y_test = to_categorical(y_test.reshape([-1, 1]))
plt.imshow(x_train[0])
```

Out[9]:
```
<matplotlib.image.AxesImage at 0x7ff73471e898>
```



In [10]:
```
# define model & compile
input_layer = l.Input((28,28))
Intermediate_layer = l.Flatten()(input_layer)
Intermediate_layer=l.Dense(256, activation='relu')(Intermediate_layer)
output_layer = l.Dense(10, activation="softmax")(Intermediate_layer)

model = m.Model(input_layer, output_layer)
model.summary()
model.compile(optimizer=o.SGD(),
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_1 (InputLayer)         (None, 28, 28)            0         
_________________________________________________________________
flatten_2 (Flatten)          (None, 784)               0         
_________________________________________________________________
dense_3 (Dense)              (None, 256)               200960    
_________________________________________________________________
dense_4 (Dense)              (None, 10)                2570      
=================================================================
Total params: 203,530
Trainable params: 203,530
Non-trainable params: 0
_________________________________________________________________
```

In [12]:

```
model.fit(x_train,y_train, epochs=10, validation_data = (x_test,y_test))
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/10
60000/60000 [==============================] - 3s 55us/step - loss: 0.6223
- acc: 0.8451 - val_loss: 0.3476 - val_acc: 0.9071
Epoch 2/10
60000/60000 [==============================] - 3s 53us/step - loss: 0.3284
- acc: 0.9082 - val_loss: 0.2839 - val_acc: 0.9204
Epoch 3/10
60000/60000 [==============================] - 3s 53us/step - loss: 0.2797
- acc: 0.9218 - val_loss: 0.2522 - val_acc: 0.9309
Epoch 4/10
60000/60000 [==============================] - 3s 53us/step - loss: 0.2490
- acc: 0.9309 - val_loss: 0.2309 - val_acc: 0.9333
Epoch 5/10
60000/60000 [==============================] - 3s 52us/step - loss: 0.2260
- acc: 0.9372 - val_loss: 0.2110 - val_acc: 0.9391
Epoch 6/10
60000/60000 [==============================] - 3s 52us/step - loss: 0.2072
- acc: 0.9423 - val_loss: 0.1973 - val_acc: 0.9444
Epoch 7/10
60000/60000 [==============================] - 3s 52us/step - loss: 0.1916
- acc: 0.9468 - val_loss: 0.1832 - val_acc: 0.9482
Epoch 8/10
60000/60000 [==============================] - 3s 53us/step - loss: 0.1781
- acc: 0.9502 - val_loss: 0.1716 - val_acc: 0.9509
Epoch 9/10
60000/60000 [==============================] - 3s 52us/step - loss: 0.1665
- acc: 0.9541 - val_loss: 0.1624 - val_acc: 0.9533
Epoch 10/10
60000/60000 [==============================] - 3s 55us/step - loss: 0.1563
- acc: 0.9564 - val_loss: 0.1558 - val_acc: 0.9551
```

Out[12]: `<keras.callbacks.History at 0x7ff7347d0c18>`

Obtain model gradients

In [1]:

```
import keras.backend as K

# function to obtain grads for each parameter
def get_gradients(model, inputs, outputs):
  grads = model.optimizer.get_gradients(model.total_loss, model.trainable_v
  symb_inputs = (model._feed_inputs + model._feed_targets + model._feed_sam
  f = K.function(symb_inputs, grads)
  x, y, weight = model._standardize_user_data(inputs, outputs)
  output_grad = f(x + y + weight)
  return np.array(output_grad)
```

```
Using TensorFlow backend.
```

In [23]:

```
# test
grads = get_gradients(model, x_train[0:1], y_train[0:1])
```

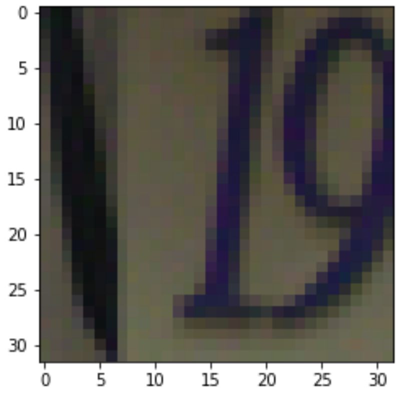
In [33]:
```
# Print max from each layer, weight and biases. Even indices hold biases
print(grads.shape)
for i,_ in enumerate(grads):
    print(grads[i].shape)
    max_gradient_layer_i = np.max(grads[i])
    print(max_gradient_layer_i)
```

```
(4,)
(784, 256)
0.0068208314
(256,)
0.0068208314
(256, 10)
0.034386244
(10,)
0.011868487
```

Load .mat files

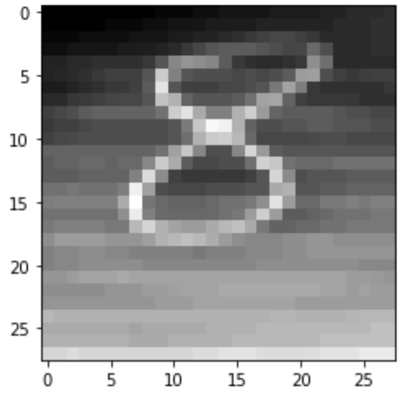
In [50]:
```
from scipy.io import loadmat

SVHN_directory = "train_32x32.mat"
# load .mat file
data_raw = loadmat(SVHN_directory)
data = np.array(data_raw['X'])
# make correct shape
data = np.moveaxis(data, -1, 0)
print(data.shape)
plt.imshow(data[0])
labels = data_raw['y']
print(labels.shape)
print(labels[0])
```

```
(73257, 32, 32, 3)
(73257, 1)
[1]
```



Make your own digits

```
In [96]: import cv2 as cv
         from skimage.transform import resize

         image_dir = "test_digit2.jpg"
         # load & smoothen image
         kernel = np.ones((7,7),np.float32)/49
         image = cv.imread(image_dir,cv.IMREAD_GRAYSCALE)
         image = cv.filter2D(image,-1,kernel)

         # make numpy array
         image = np.array(image)
         image = resize(image, (28,28))
         # make negative
         image = np.ones(image.shape) - image

         plt.imshow(image, cmap="gray")
         plt.show()
```



```
In [97]: print(image.shape)
         # predict label
         np.argmax(model.predict(np.array([image])))

         (28, 28)

Out[97]: 8

In [ ]:
```