Classify_Traffic_Sign_Images_Using_LeNet

October 18, 2019

1 PROBLEM STATEMENT

In this case study, we have been provided with images of traffic signs and the goal is to train a Deep Network to classify them

Figure 1. Traffic Sign Classification-

- The dataset contains 43 different classes of images.
- Classes are as listed below:
  - (0, 'Speed limit (20km/h)')
  - (1, 'Speed limit (30km/h)')
  - (2, 'Speed limit (50km/h)')
  - (3, 'Speed limit (60km/h)')
  - (4, 'Speed limit (70km/h)')
  - (5, 'Speed limit (80km/h)')
  - (6, 'End of speed limit (80km/h)')
  - (7, 'Speed limit (100km/h)')
  - (8, 'Speed limit (120km/h)')
  - (9, 'No passing')
  - (10, 'No passing for vehicles over 3.5 metric tons')
  - (11, 'Right-of-way at the next intersection')
  - (12, 'Priority road')
  - (13, 'Yield') (14, b'Stop')
  - (15, 'No vehicles')
  - (16, 'Vehicles over 3.5 metric tons prohibited')
  - (17, 'No entry')
  - (18, 'General caution')
  - (19, 'Dangerous curve to the left')
  - (20, 'Dangerous curve to the right')
  - (21, 'Double curve')
  - (22, 'Bumpy road')
  - (23, 'Slippery road')
  - (24, 'Road narrows on the right')
  - (25, 'Road work')
  - (26, 'Traffic signals')
  - (27, 'Pedestrians')
  - (28, 'Children crossing')

- (29, 'Bicycles crossing')
  - (30, 'Beware of ice/snow')
  - (31, 'Wild animals crossing')
  - (32, 'End of all speed and passing limits')
  - (33, 'Turn right ahead')
  - (34, 'Turn left ahead')
  - (35, 'Ahead only')
  - (36, 'Go straight or right')
  - (37, 'Go straight or left')
  - (38, 'Keep right')
  - (39, 'Keep left')
  - (40, 'Roundabout mandatory')
  - (41, 'End of no passing')
  - (42, 'End of no passing by vehicles over 3.5 metric tons')
- The network used is called LeNet that was presented by Yann LeCun http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf
- Citation: *J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In Proceedings of the IEEE International Joint Conference on Neural Networks, pages 1453–1460. 2011. @inproceedings{Stallkamp-IJCNN-2011, author = {Johannes Stallkamp and Marc Schlipsing and Jan Salmen and Christian Igel}, booktitle = {IEEE International Joint Conference on Neural Networks}, title = {The {G}erman {T}raffic {S}ign {R}ecognition {B}enchmark: A multi-class classification competition}, year = {2011}, pages = {1453–1460} }*

2 IMPORT LIBRARIES

[1]:
```
import matplotlib.pyplot as plt
import numpy as np
import os
```

[2]:
```
import pickle
import pandas as pd
import seaborn as sns
import PIL
```

[3]:
```
from tensorflow.keras import layers
import tensorflow as tf
```

3 IMPORT DATASETS AND NORMALIZE IT

[4]:
```
with open("traffic-signs-data/train.p", mode='rb') as training_data:
    train = pickle.load(training_data)
with open("traffic-signs-data/valid.p", mode='rb') as validation_data:
    valid = pickle.load(validation_data)
with open("traffic-signs-data/test.p", mode='rb') as testing_data:
    test = pickle.load(testing_data)
```

[5]:
```
X_train, y_train = train['features'], train['labels']
X_validation, y_validation = valid['features'], valid['labels']
X_test, y_test = test['features'], test['labels']
```

[6]:
```
X_train.shape
```

[6]:
```
(34799, 32, 32, 3)
```

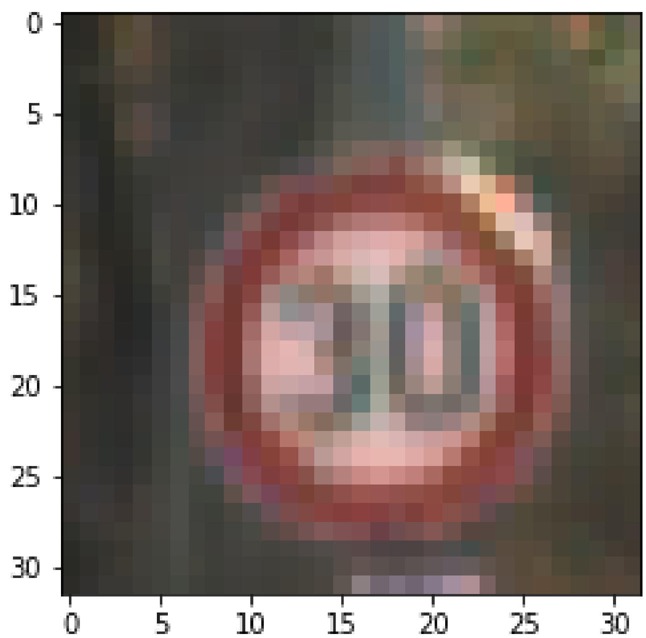
[7]:
```
y_train.shape
```

[7]:
```
(34799,)
```

4 VISUALIZE DATASET

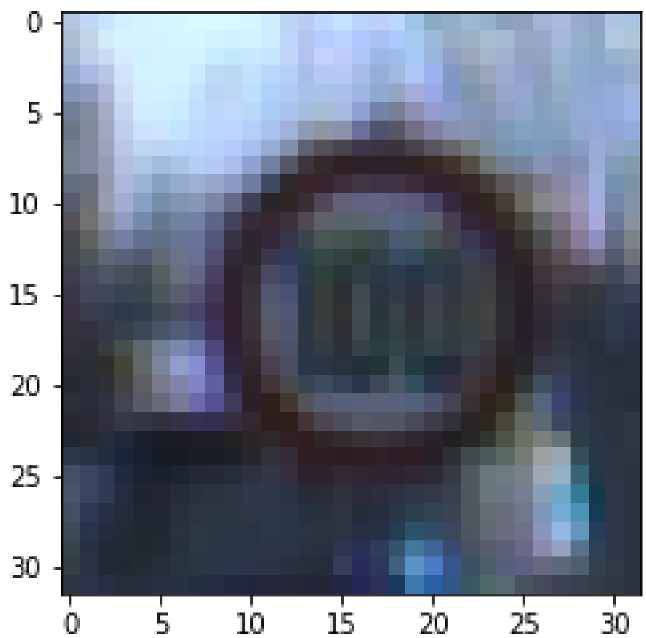
[8]:
```
i = 3100
plt.imshow(X_train[i])
y_train[i]
```

[8]:
```
1
```



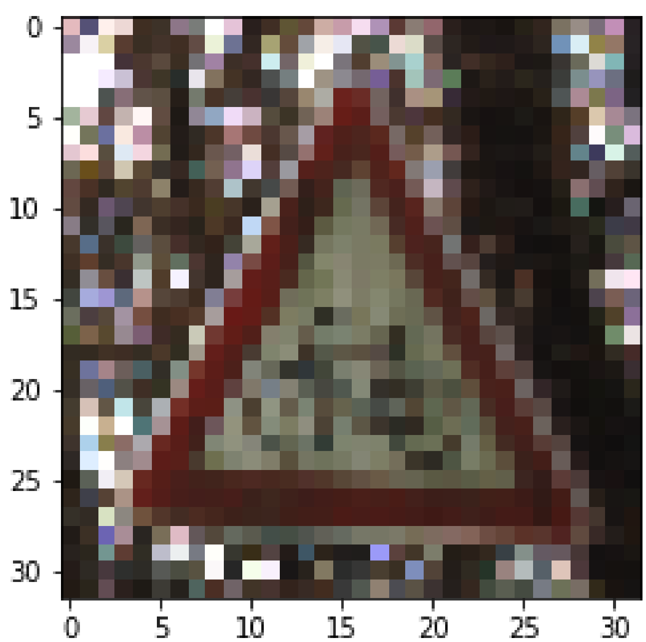
[9]:
```
i = 3001
plt.imshow(X_validation[i])
y_validation[i]
```

[9]:
```
7
```



[10]:
```
i = 2100
plt.imshow(X_test[i])
y_test[i]
```

[10]: 29



5 DATA PREPARATION

[11]:
```
from sklearn.utils import shuffle
X_train, y_train = shuffle(X_train, y_train)
```

[12]:
```
X_train_gray = np.sum(X_train/3, axis = 3, keepdims = True)
X_test_gray = np.sum(X_test/3, axis = 3, keepdims = True)
X_validation_gray = np.sum(X_validation/3, axis = 3, keepdims = True)
```

[13]:
```
X_train_gray.shape
```

[13]:
```
(34799, 32, 32, 1)
```

[14]:
```
X_test_gray.shape
```

[14]:
```
(12630, 32, 32, 1)
```

[15]:
```
X_validation_gray.shape
```

[15]:
```
(4410, 32, 32, 1)
```

[16]:
```
X_train_gray_norm = (X_train_gray - 128)/128
X_test_gray_norm = (X_test_gray - 128)/128
X_validation_gray_norm = (X_validation_gray - 128)/128
```

[17]:
```
X_train_gray_norm
```

[17]:
```
array([[[[-0.7890625 ],
         [-0.78125   ],
         [-0.76822917],
         ...,
         [-0.76041667],
         [-0.77083333],
         [-0.78385417]],

        [[-0.78125   ],
         [-0.77864583],
         [-0.77604167],
         ...,
         [-0.765625  ],
         [-0.76302083],
         [-0.77864583]],

        [[-0.77864583],
         [-0.77604167],
         [-0.77604167],
         ...,
         [-0.77083333],
         [-0.77083333],
         [-0.78385417]],

        ...,
```

```
 [[-0.79947917],
  [-0.80729167],
  [-0.81510417],
  ...,
  [-0.7890625 ],
  [-0.7890625 ],
  [-0.7890625 ]],

 [[-0.79947917],
  [-0.80729167],
  [-0.80729167],
  ...,
  [-0.796875  ],
  [-0.79427083],
  [-0.7890625 ]],

 [[-0.79166667],
  [-0.79947917],
  [-0.796875  ],
  ...,
  [-0.7890625 ],
  [-0.79427083],
  [-0.79166667]]],


[[[ 0.9921875 ],
  [ 0.9921875 ],
  [ 0.9921875 ],
  ...,
  [ 0.984375  ],
  [ 0.88802083],
  [ 0.79427083]],

 [[ 0.9921875 ],
  [ 0.9921875 ],
  [ 0.9921875 ],
  ...,
  [ 0.9921875 ],
  [ 0.79166667],
  [ 0.91927083]],

 [[ 0.9921875 ],
  [ 0.9921875 ],
  [ 0.9921875 ],
  ...,
  [ 0.98958333],
```

```
  [ 0.5703125 ],
  [ 0.77864583]],

 ...,

 [[ 0.9921875 ],
  [ 0.9921875 ],
  [ 0.9921875 ],
  ...,
  [ 0.953125  ],
  [ 0.96354167],
  [ 0.96614583]],

 [[ 0.9921875 ],
  [ 0.9921875 ],
  [ 0.9921875 ],
  ...,
  [ 0.9921875 ],
  [ 0.9921875 ],
  [ 0.9921875 ]],

 [[ 0.9921875 ],
  [ 0.9921875 ],
  [ 0.9921875 ],
  ...,
  [ 0.9921875 ],
  [ 0.9921875 ],
  [ 0.9921875 ]]],


[[[-0.64322917],
  [-0.63541667],
  [-0.625     ],
  ...,
  [-0.7265625 ],
  [-0.71614583],
  [-0.7109375 ]],

 [[-0.65625   ],
  [-0.640625  ],
  [-0.64583333],
  ...,
  [-0.734375  ],
  [-0.734375  ],
  [-0.72135417]],

 [[-0.6953125 ],
```

```
  [-0.66145833],
  [-0.66666667],
  ...,
  [-0.72135417],
  [-0.734375  ],
  [-0.734375  ]],

 ...,

 [[-0.67447917],
  [-0.69270833],
  [-0.69791667],
  ...,
  [-0.72395833],
  [-0.734375  ],
  [-0.73958333]],

 [[-0.6640625 ],
  [-0.69270833],
  [-0.7109375 ],
  ...,
  [-0.68489583],
  [-0.70052083],
  [-0.72916667]],

 [[-0.69270833],
  [-0.70833333],
  [-0.7109375 ],
  ...,
  [-0.69270833],
  [-0.70833333],
  [-0.7265625 ]]],


...,


[[[ 0.25520833],
  [ 0.47135417],
  [ 0.6328125 ],
  ...,
  [-0.47916667],
  [-0.7734375 ],
  [-0.79947917]],

 [[ 0.6796875 ],
  [ 0.59635417],
```

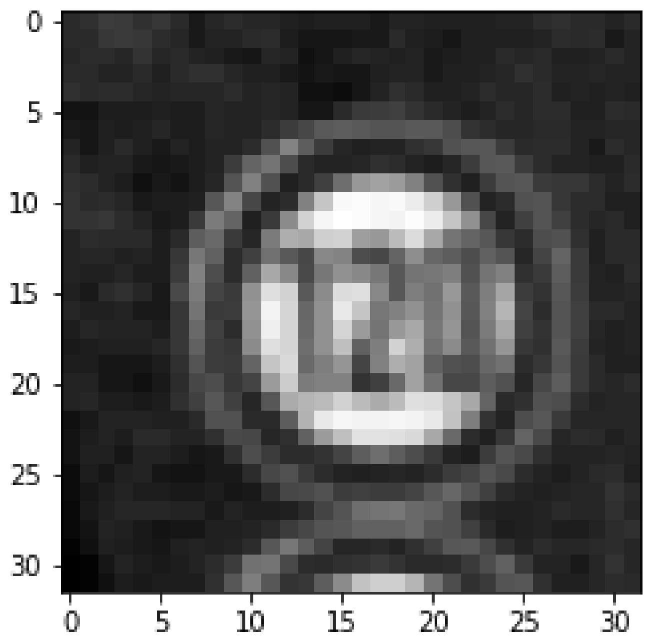
```
  [ 0.57552083],
  ...,
  [-0.46354167],
  [-0.76822917],
  [-0.81510417]],

 [[ 0.68229167],
  [ 0.703125  ],
  [ 0.671875  ],
  ...,
  [-0.44791667],
  [-0.7578125 ],
  [-0.80208333]],

 ...,

 [[-0.76302083],
  [-0.734375  ],
  [-0.75260417],
  ...,
  [-0.70833333],
  [-0.52864583],
  [-0.796875  ]],

 [[-0.77864583],
  [-0.7734375 ],
  [-0.77604167],
  ...,
  [-0.72135417],
  [-0.53645833],
  [-0.78385417]],

 [[-0.80729167],
  [-0.78645833],
  [-0.78125   ],
  ...,
  [-0.73697917],
  [-0.54427083],
  [-0.76302083]]],


[[[-0.38802083],
  [-0.3828125 ],
  [-0.24479167],
  ...,
  [-0.48958333],
  [-0.55729167],
```

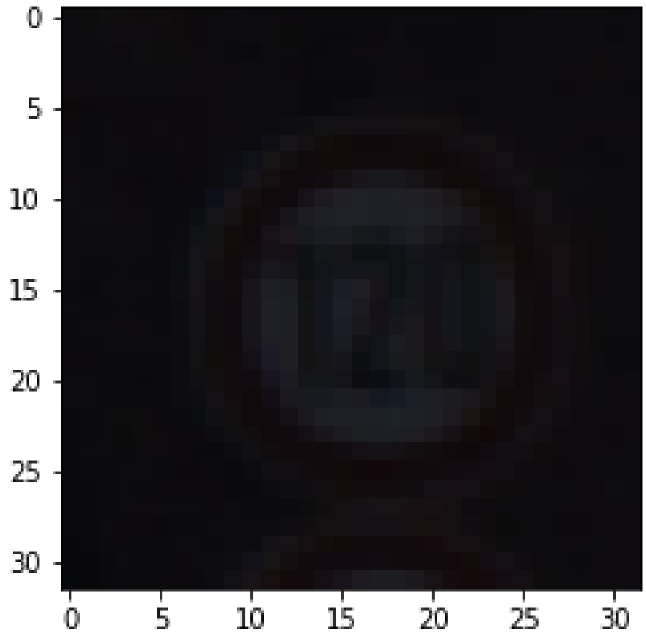
```
 [-0.5703125 ]],

[[-0.41927083],
 [-0.48177083],
 [-0.34635417],
 ...,
 [-0.484375  ],
 [-0.55208333],
 [-0.5078125 ]],

[[-0.59895833],
 [-0.61458333],
 [-0.48958333],
 ...,
 [-0.5       ],
 [-0.56510417],
 [-0.48177083]],

...,

[[ 0.17708333],
 [ 0.1875    ],
 [ 0.21614583],
 ...,
 [-0.4375    ],
 [-0.546875  ],
 [-0.63020833]],

[[ 0.33072917],
 [ 0.33333333],
 [ 0.33333333],
 ...,
 [-0.45052083],
 [-0.56770833],
 [-0.64322917]],

[[ 0.33333333],
 [ 0.328125  ],
 [ 0.3359375 ],
 ...,
 [-0.4453125 ],
 [-0.54166667],
 [-0.59114583]]],


[[[-0.640625  ],
 [-0.62239583],
```

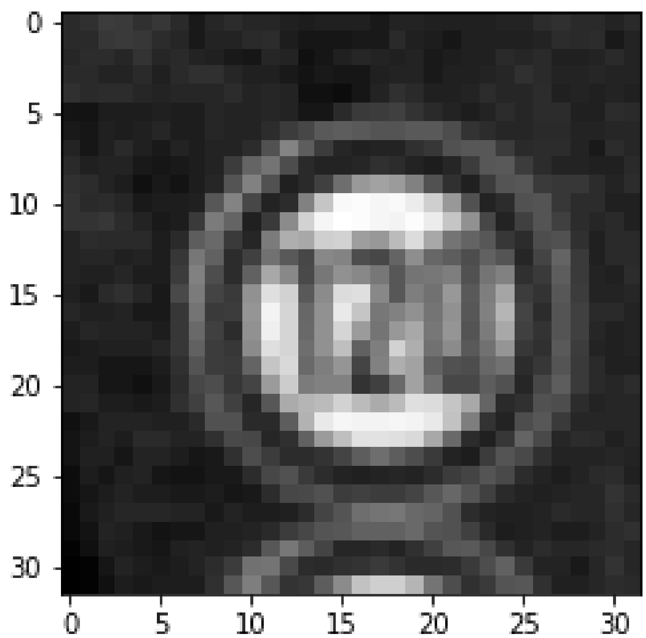
```
 [-0.61979167],
 ...,
 [-0.58854167],
 [-0.65625   ],
 [-0.64583333]],

[[-0.65364583],
 [-0.65364583],
 [-0.65885417],
 ...,
 [-0.61197917],
 [-0.671875  ],
 [-0.65364583]],

[[-0.67447917],
 [-0.65885417],
 [-0.64583333],
 ...,
 [-0.64583333],
 [-0.68229167],
 [-0.6328125 ]],

...,

[[-0.61197917],
 [-0.61458333],
 [-0.61458333],
 ...,
 [-0.65625   ],
 [-0.65364583],
 [-0.65364583]],

[[-0.59895833],
 [-0.58854167],
 [-0.57552083],
 ...,
 [-0.65104167],
 [-0.63802083],
 [-0.63020833]],

[[-0.56510417],
 [-0.57552083],
 [-0.58333333],
 ...,
 [-0.6484375 ],
 [-0.59895833],
 [-0.60416667]]]])
```

[18]:
```python
i = 60
plt.imshow(X_train_gray[i].squeeze(), cmap = 'gray')
plt.figure()
plt.imshow(X_train[i])
plt.figure()
plt.imshow(X_train_gray_norm[i].squeeze(), cmap = 'gray')
```

[18]: <matplotlib.image.AxesImage at 0x1c70007fcf8>



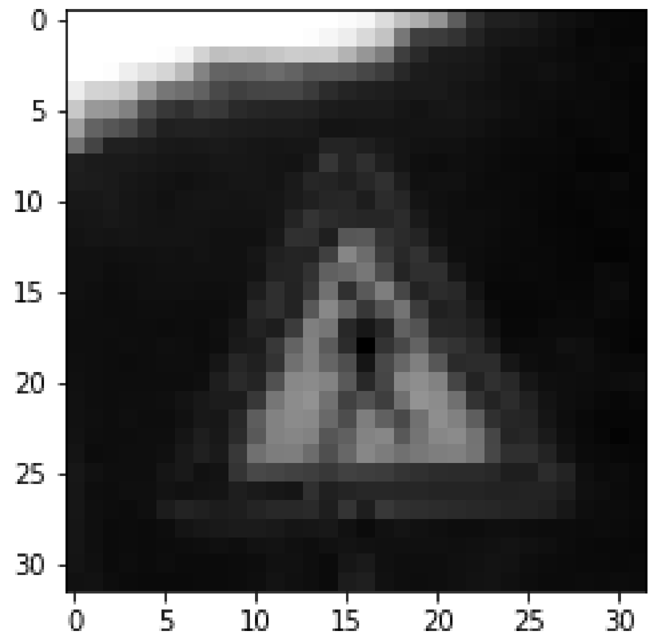


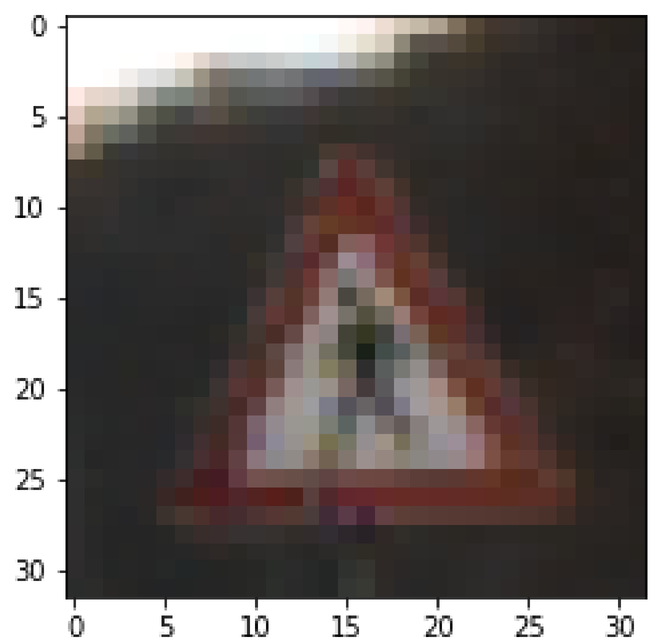


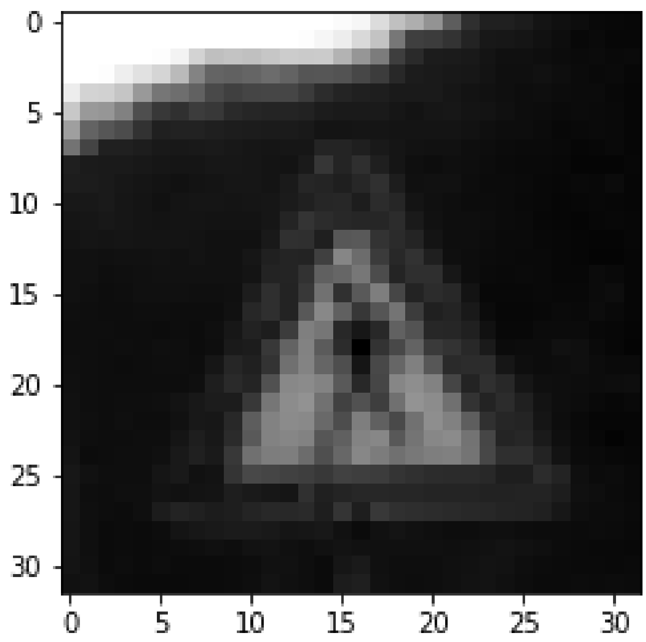
[19]:
```
i = 610
plt.imshow(X_test_gray[i].squeeze(), cmap = 'gray')
plt.figure()
plt.imshow(X_test[i])
plt.figure()
```

```
plt.imshow(X_test_gray_norm[i].squeeze(), cmap = 'gray')
```

[19]: <matplotlib.image.AxesImage at 0x1c7001b34e0>

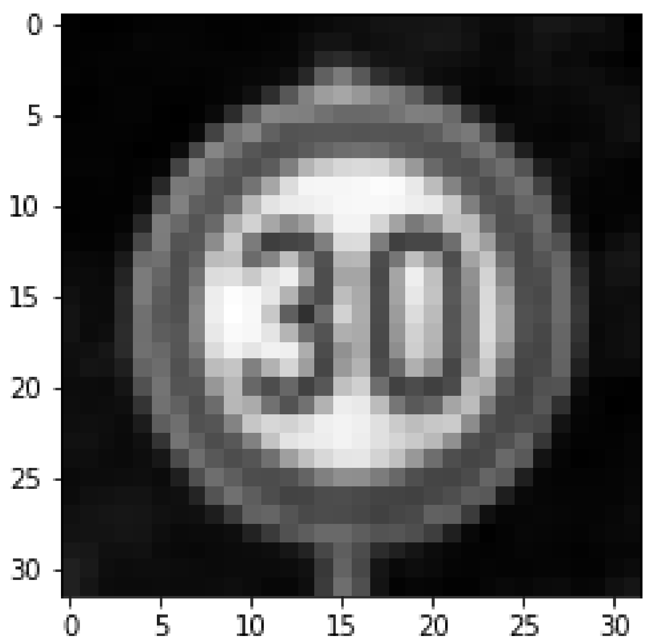


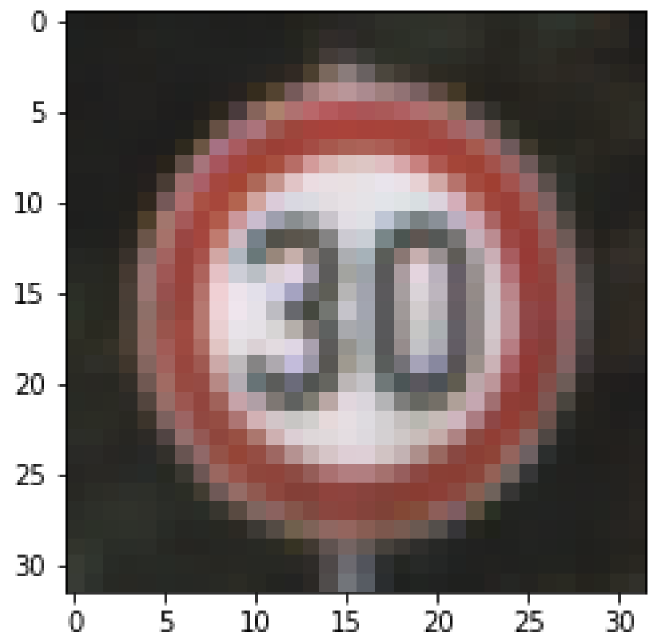


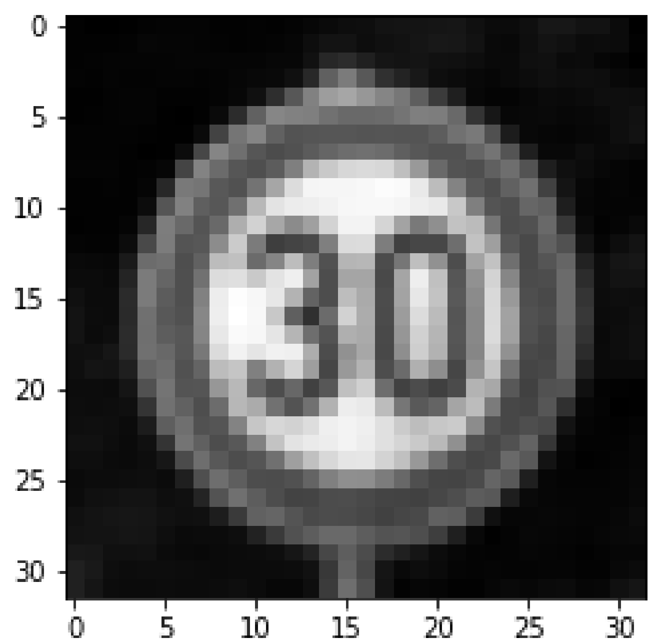


[20]:
```
i = 500
plt.imshow(X_validation_gray[i].squeeze(), cmap = 'gray')
plt.figure()
plt.imshow(X_validation[i])
plt.figure()
plt.imshow(X_validation_gray_norm[i].squeeze(), cmap = 'gray')
```

[20]:
```
<matplotlib.image.AxesImage at 0x1c75bb52f28>
```




0
5
10
15
20
25
30
0
5
10
15
20
25
30


0
5
10
15
20
25
30
0
5
10
15
20
25
30

6 MODEL TRAINING

The model consists of the following layers:

- **STEP 1:** THE FIRST CONVOLUTIONAL LAYER #1
  - Input = 32x32x1
  - Output = 28x28x6
  - Output = (Input-filter+1)/Stride* => (32-5+1)/1=28
  - Used a 5x5 Filter with input depth of 3 and output depth of 6
  - Apply a RELU Activation function to the output
  - pooling for input, Input = 28x28x6 and Output = 14x14x6
  - Stride is the amount by which the kernel is shifted when the kernel is passed over the image.
- **STEP 2:** THE SECOND CONVOLUTIONAL LAYER #2
  - Input = 14x14x6
  - Output = 10x10x16
  - Layer 2: Convolutional layer with Output = 10x10x16
  - Output = (Input-filter+1)/strides => 10 = 14-5+1/1
  - Apply a RELU Activation function to the output
  - Pooling with Input = 10x10x16 and Output = 5x5x16
- **STEP 3:** FLATTENING THE NETWORK
  - Flatten the network with Input = 5x5x16 and Output = 400
- **STEP 4:** FULLY CONNECTED LAYER
  - Layer 3: Fully Connected layer with Input = 400 and Output = 120
  - Apply a RELU Activation function to the output
- **STEP 5:** ANOTHER FULLY CONNECTED LAYER
  - Layer 4: Fully Connected Layer with Input = 120 and Output = 84
  - Apply a RELU Activation function to the output
- **STEP 6:** FULLY CONNECTED LAYER
  - Layer 5: Fully Connected layer with Input = 84 and Output = 43

[21]:
```python
from tensorflow.keras import datasets, layers, models

LeNet = models.Sequential()

LeNet.add(layers.Conv2D(6, (5,5), activation = 'relu', input_shape = (32,32,1)))
LeNet.add(layers.AveragePooling2D())


LeNet.add(layers.Conv2D(16, (5,5), activation = 'relu'))
```

```
LeNet.add(layers.AveragePooling2D())

LeNet.add(layers.Flatten())

LeNet.add(layers.Dense(120, activation = 'relu'))

LeNet.add(layers.Dense(84, activation = 'relu'))

LeNet.add(layers.Dense(43, activation = 'softmax'))
LeNet.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 6)         156
_________________________________________________________________
average_pooling2d (AveragePo (None, 14, 14, 6)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 10, 10, 16)        2416
_________________________________________________________________
average_pooling2d_1 (Average (None, 5, 5, 16)          0
_________________________________________________________________
flatten (Flatten)            (None, 400)               0
_________________________________________________________________
dense (Dense)                (None, 120)               48120
_________________________________________________________________
dense_1 (Dense)              (None, 84)                10164
_________________________________________________________________
dense_2 (Dense)              (None, 43)                3655
=================================================================
Total params: 64,511
Trainable params: 64,511
Non-trainable params: 0
_________________________________________________________________
```

[22]:
```
LeNet.compile(optimizer = 'Adam', loss = 'sparse_categorical_crossentropy', metrics = ['accuracy'])
```

[23]:
```
history = LeNet.fit(X_train_gray_norm,
                 y_train,
                 batch_size = 500,
                 nb_epoch = 50,
                 verbose = 1,
                 validation_data = (X_validation_gray_norm, y_validation))
```

```
WARNING: Logging before flag parsing goes to stderr.
```

```
W1018 22:13:07.586207 16828 training.py:701] The `nb_epoch` argument in `fit`
has been renamed `epochs`.

Train on 34799 samples, validate on 4410 samples
Epoch 1/50
34799/34799 [==============================] - 17s 478us/sample - loss: 3.1495 -
accuracy: 0.1815 - val_loss: 2.6016 - val_accuracy: 0.3184
Epoch 2/50
34799/34799 [==============================] - 15s 429us/sample - loss: 1.5505 -
accuracy: 0.5693 - val_loss: 1.2840 - val_accuracy: 0.6236
Epoch 3/50
34799/34799 [==============================] - 15s 438us/sample - loss: 0.8503 -
accuracy: 0.7624 - val_loss: 0.9664 - val_accuracy: 0.7333
Epoch 4/50
34799/34799 [==============================] - 15s 435us/sample - loss: 0.6108 -
accuracy: 0.8341 - val_loss: 0.8458 - val_accuracy: 0.7651
Epoch 5/50
34799/34799 [==============================] - 15s 437us/sample - loss: 0.4888 -
accuracy: 0.8687 - val_loss: 0.7665 - val_accuracy: 0.7871
Epoch 6/50
34799/34799 [==============================] - 15s 433us/sample - loss: 0.4101 -
accuracy: 0.8903 - val_loss: 0.7181 - val_accuracy: 0.8111
Epoch 7/50
34799/34799 [==============================] - 16s 454us/sample - loss: 0.3481 -
accuracy: 0.9082 - val_loss: 0.6641 - val_accuracy: 0.8234
Epoch 8/50
34799/34799 [==============================] - 16s 460us/sample - loss: 0.3001 -
accuracy: 0.9228 - val_loss: 0.6393 - val_accuracy: 0.8297
Epoch 9/50
34799/34799 [==============================] - 15s 444us/sample - loss: 0.2659 -
accuracy: 0.9324 - val_loss: 0.6417 - val_accuracy: 0.8363
Epoch 10/50
34799/34799 [==============================] - 17s 481us/sample - loss: 0.2316 -
accuracy: 0.9404 - val_loss: 0.6282 - val_accuracy: 0.8433
Epoch 11/50
34799/34799 [==============================] - 17s 476us/sample - loss: 0.2096 -
accuracy: 0.9467 - val_loss: 0.6213 - val_accuracy: 0.8422
Epoch 12/50
34799/34799 [==============================] - 15s 445us/sample - loss: 0.1943 -
accuracy: 0.9499 - val_loss: 0.6205 - val_accuracy: 0.8531
Epoch 13/50
34799/34799 [==============================] - 15s 434us/sample - loss: 0.1701 -
accuracy: 0.9571 - val_loss: 0.5958 - val_accuracy: 0.8524
Epoch 14/50
34799/34799 [==============================] - 16s 451us/sample - loss: 0.1537 -
accuracy: 0.9620 - val_loss: 0.5945 - val_accuracy: 0.8522
Epoch 15/50
34799/34799 [==============================] - 15s 433us/sample - loss: 0.1381 -
```

```
accuracy: 0.9648 - val_loss: 0.5730 - val_accuracy: 0.8574
Epoch 16/50
34799/34799 [==============================] - 15s 444us/sample - loss: 0.1281 -
accuracy: 0.9685 - val_loss: 0.6105 - val_accuracy: 0.8587
Epoch 17/50
34799/34799 [==============================] - 15s 438us/sample - loss: 0.1196 -
accuracy: 0.9700 - val_loss: 0.5860 - val_accuracy: 0.8583
Epoch 18/50
34799/34799 [==============================] - 15s 436us/sample - loss: 0.1098 -
accuracy: 0.9734 - val_loss: 0.6329 - val_accuracy: 0.8533
Epoch 19/50
34799/34799 [==============================] - 15s 423us/sample - loss: 0.1020 -
accuracy: 0.9741 - val_loss: 0.6231 - val_accuracy: 0.8617
Epoch 20/50
34799/34799 [==============================] - 16s 467us/sample - loss: 0.0899 -
accuracy: 0.9774 - val_loss: 0.5986 - val_accuracy: 0.8719
Epoch 21/50
34799/34799 [==============================] - 18s 508us/sample - loss: 0.0858 -
accuracy: 0.9791 - val_loss: 0.6339 - val_accuracy: 0.8637
Epoch 22/50
34799/34799 [==============================] - 15s 427us/sample - loss: 0.0776 -
accuracy: 0.9814 - val_loss: 0.6581 - val_accuracy: 0.8649
Epoch 23/50
34799/34799 [==============================] - 14s 416us/sample - loss: 0.0746 -
accuracy: 0.9818 - val_loss: 0.6833 - val_accuracy: 0.8553
Epoch 24/50
34799/34799 [==============================] - 14s 401us/sample - loss: 0.0673 -
accuracy: 0.9833 - val_loss: 0.6759 - val_accuracy: 0.8667
Epoch 25/50
34799/34799 [==============================] - 12s 340us/sample - loss: 0.0623 -
accuracy: 0.9856 - val_loss: 0.6929 - val_accuracy: 0.8628
Epoch 26/50
34799/34799 [==============================] - 12s 340us/sample - loss: 0.0563 -
accuracy: 0.9866 - val_loss: 0.7099 - val_accuracy: 0.8594
Epoch 27/50
34799/34799 [==============================] - 12s 339us/sample - loss: 0.0529 -
accuracy: 0.9874 - val_loss: 0.6834 - val_accuracy: 0.8692
Epoch 28/50
34799/34799 [==============================] - 12s 334us/sample - loss: 0.0510 -
accuracy: 0.9879 - val_loss: 0.7080 - val_accuracy: 0.8696
Epoch 29/50
34799/34799 [==============================] - 12s 338us/sample - loss: 0.0475 -
accuracy: 0.9887 - val_loss: 0.7828 - val_accuracy: 0.8660
Epoch 30/50
34799/34799 [==============================] - 12s 348us/sample - loss: 0.0443 -
accuracy: 0.9895 - val_loss: 0.7569 - val_accuracy: 0.8612
Epoch 31/50
34799/34799 [==============================] - 12s 337us/sample - loss: 0.0406 -
```

```
accuracy: 0.9905 - val_loss: 0.7930 - val_accuracy: 0.8669
Epoch 32/50
34799/34799 [==============================] - 12s 334us/sample - loss: 0.0377 -
accuracy: 0.9915 - val_loss: 0.7902 - val_accuracy: 0.8669
Epoch 33/50
34799/34799 [==============================] - 12s 351us/sample - loss: 0.0383 -
accuracy: 0.9905 - val_loss: 0.7467 - val_accuracy: 0.8669
Epoch 34/50
34799/34799 [==============================] - 12s 342us/sample - loss: 0.0344 -
accuracy: 0.9917 - val_loss: 0.7562 - val_accuracy: 0.8660
Epoch 35/50
34799/34799 [==============================] - 12s 334us/sample - loss: 0.0303 -
accuracy: 0.9933 - val_loss: 0.7618 - val_accuracy: 0.8728
Epoch 36/50
34799/34799 [==============================] - 13s 366us/sample - loss: 0.0320 -
accuracy: 0.9919 - val_loss: 0.7860 - val_accuracy: 0.8735
Epoch 37/50
34799/34799 [==============================] - 13s 386us/sample - loss: 0.0318 -
accuracy: 0.9919 - val_loss: 0.8505 - val_accuracy: 0.8680
Epoch 38/50
34799/34799 [==============================] - 12s 357us/sample - loss: 0.0308 -
accuracy: 0.9916 - val_loss: 0.6967 - val_accuracy: 0.8739
Epoch 39/50
34799/34799 [==============================] - 12s 347us/sample - loss: 0.0267 -
accuracy: 0.9937 - val_loss: 0.6780 - val_accuracy: 0.8900
Epoch 40/50
34799/34799 [==============================] - 12s 338us/sample - loss: 0.0235 -
accuracy: 0.9942 - val_loss: 0.8278 - val_accuracy: 0.8671
Epoch 41/50
34799/34799 [==============================] - 12s 341us/sample - loss: 0.0211 -
accuracy: 0.9948 - val_loss: 0.7804 - val_accuracy: 0.8741
Epoch 42/50
34799/34799 [==============================] - 12s 354us/sample - loss: 0.0164 -
accuracy: 0.9966 - val_loss: 0.7662 - val_accuracy: 0.8748
Epoch 43/50
34799/34799 [==============================] - 12s 351us/sample - loss: 0.0180 -
accuracy: 0.9957 - val_loss: 0.7943 - val_accuracy: 0.8680
Epoch 44/50
34799/34799 [==============================] - 13s 367us/sample - loss: 0.0178 -
accuracy: 0.9957 - val_loss: 0.7970 - val_accuracy: 0.8710
Epoch 45/50
34799/34799 [==============================] - 14s 396us/sample - loss: 0.0184 -
accuracy: 0.9951 - val_loss: 0.9367 - val_accuracy: 0.8499
Epoch 46/50
34799/34799 [==============================] - 12s 356us/sample - loss: 0.0238 -
accuracy: 0.9932 - val_loss: 0.9788 - val_accuracy: 0.8578
Epoch 47/50
34799/34799 [==============================] - 13s 386us/sample - loss: 0.0206 -
```

```
accuracy: 0.9937 - val_loss: 0.7591 - val_accuracy: 0.8778
Epoch 48/50
34799/34799 [==============================] - 12s 349us/sample - loss: 0.0145 -
accuracy: 0.9966 - val_loss: 0.9223 - val_accuracy: 0.8624
Epoch 49/50
34799/34799 [==============================] - 12s 344us/sample - loss: 0.0115 -
accuracy: 0.9976 - val_loss: 0.9173 - val_accuracy: 0.8660
Epoch 50/50
34799/34799 [==============================] - 13s 369us/sample - loss: 0.0094 -
accuracy: 0.9980 - val_loss: 0.8907 - val_accuracy: 0.8692
```

7 MODEL EVALUATION

[24]:
```python
score = LeNet.evaluate(X_test_gray_norm, y_test)
print('Test Accuracy: {}'.format(score[1]))
```

```
12630/1 [=======================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
```

===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===

==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==

==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==

==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==

==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==

```
==========================================================================================
==========================================================================================
==========================================================================================
==========================================================================================
==========================================================================================
==========================================================================================
==========================================================================================
==========================================================================================
==========================================================================================
==========================================================================================
==========================================================================================
==========================================================================================
==========================================================================================
==========================================================================================
==========================================================================================
==========================================================================================
==========================================================================================
==========================================================================================
==========================================================================================
==========================================================================================
==========================================================================================
==========================================================================================
==========================================================================================
==========================================================================================
==========================================================================================
==========================================================================================
==========================================================================================
==========================================================================================
==========================================================================================
==========================================================================================
==========================================================================================
==========================================================================================
==========================================================================================
==========================================================================================
==========================================================================================
==========================================================================================
==========================================================================================
==========================================================================================
==========================================================================================
==========================================================================================
==========================================================================================
==========================================================================================
==========================================================================================
==========================================================================================
==========================================================================================
==========================================================================================
==========================================================================================
==========================================================================================
```

==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==

==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==

==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==

==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==

===

===

===

===

===

===

===

===

===

===

===

===

===

===

===

===

===

===

===

===

===

===

===

===

===

===

===

===

===

===

===

===

===

===

===

===

===

===

===

===

===

===

===

===

===

===

===

===

==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==

==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==
==

===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===

```
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
```

===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===
===

```
================================================================================
=============================] - 2s 177us/sample - loss: 2.1799 - accuracy:
0.8745
Test Accuracy: 0.8745051622390747
```

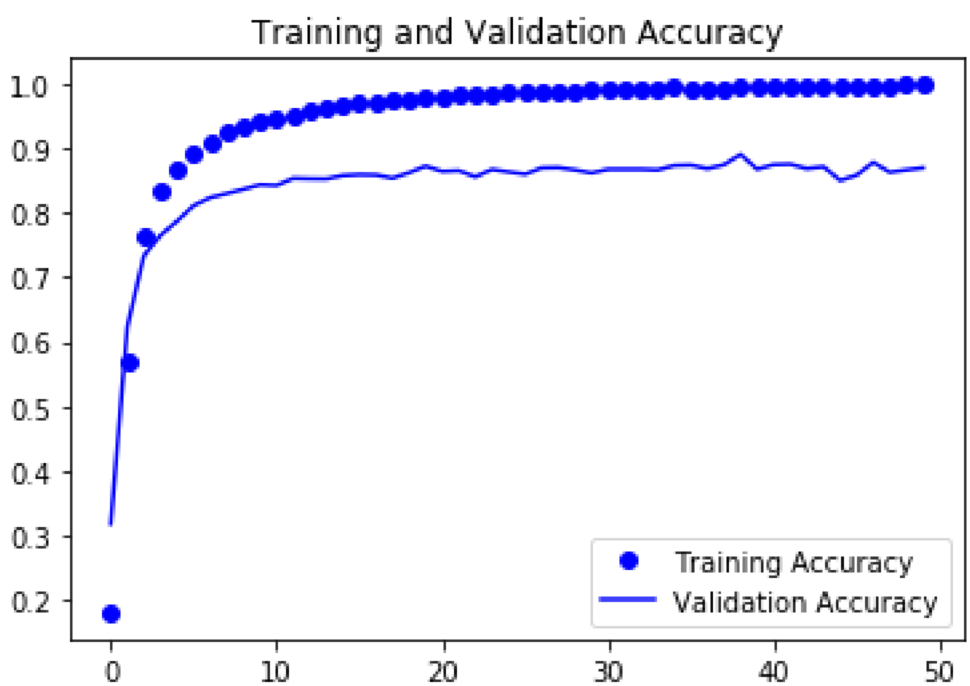
[25]:
```python
history.history.keys()
```

[25]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

[26]:
```python
accuracy = history.history['accuracy']
val_accuracy = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
```

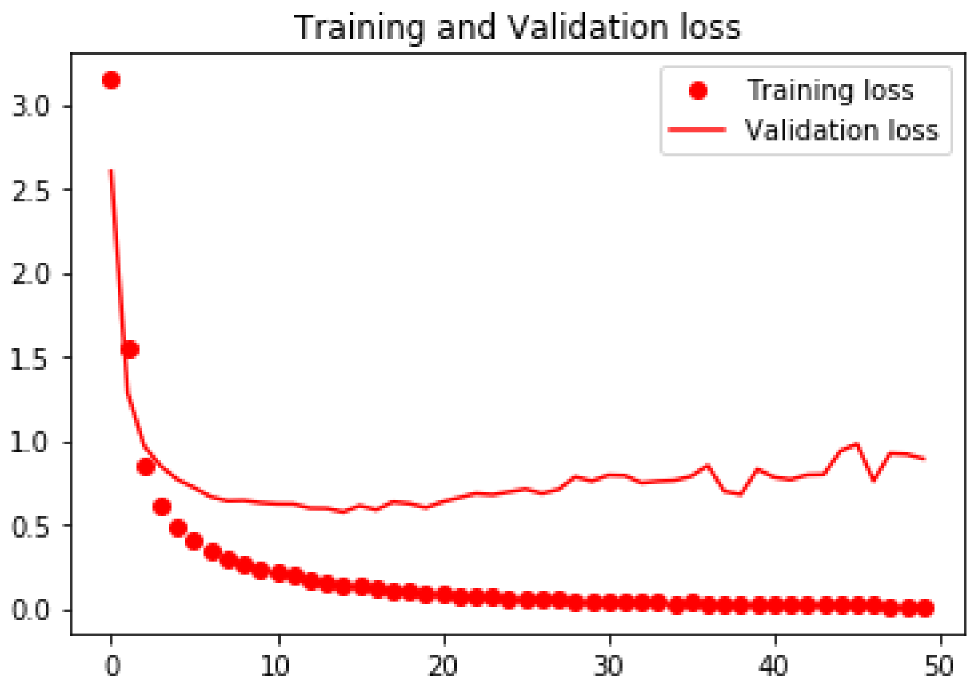
[27]:
```python
epochs = range(len(accuracy))

plt.plot(epochs, accuracy, 'bo', label='Training Accuracy')
plt.plot(epochs, val_accuracy, 'b', label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.legend()
```

[27]:
```
<matplotlib.legend.Legend at 0x1c701a50be0>
```



[28]:
```python
plt.plot(epochs, loss, 'ro', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and Validation loss')
plt.legend()
```

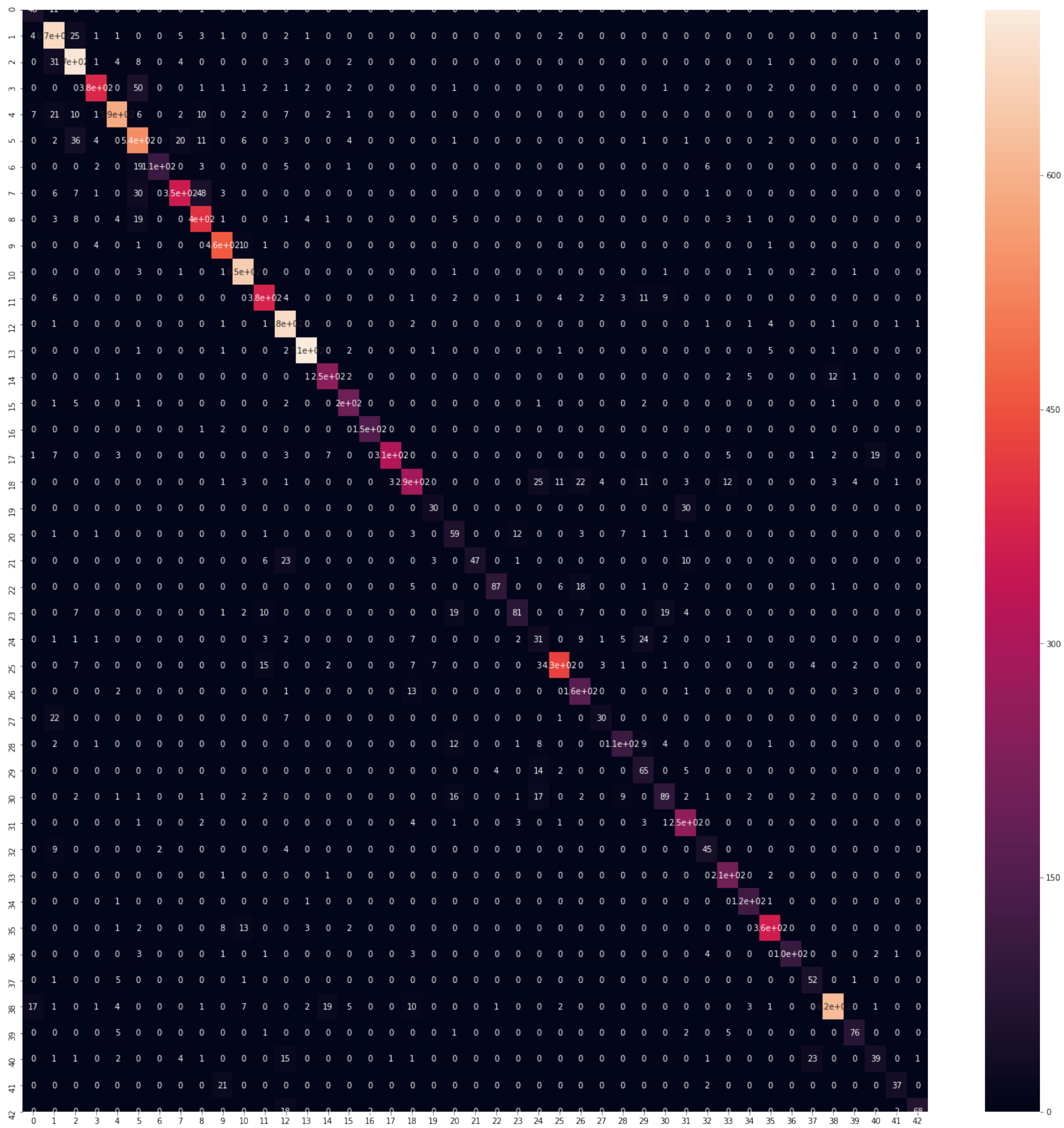
[28]: <matplotlib.legend.Legend at 0x1c70013bbe0>



[29]:
```
predicted_classes = LeNet.predict_classes(X_test_gray_norm)
y_true = y_test
```

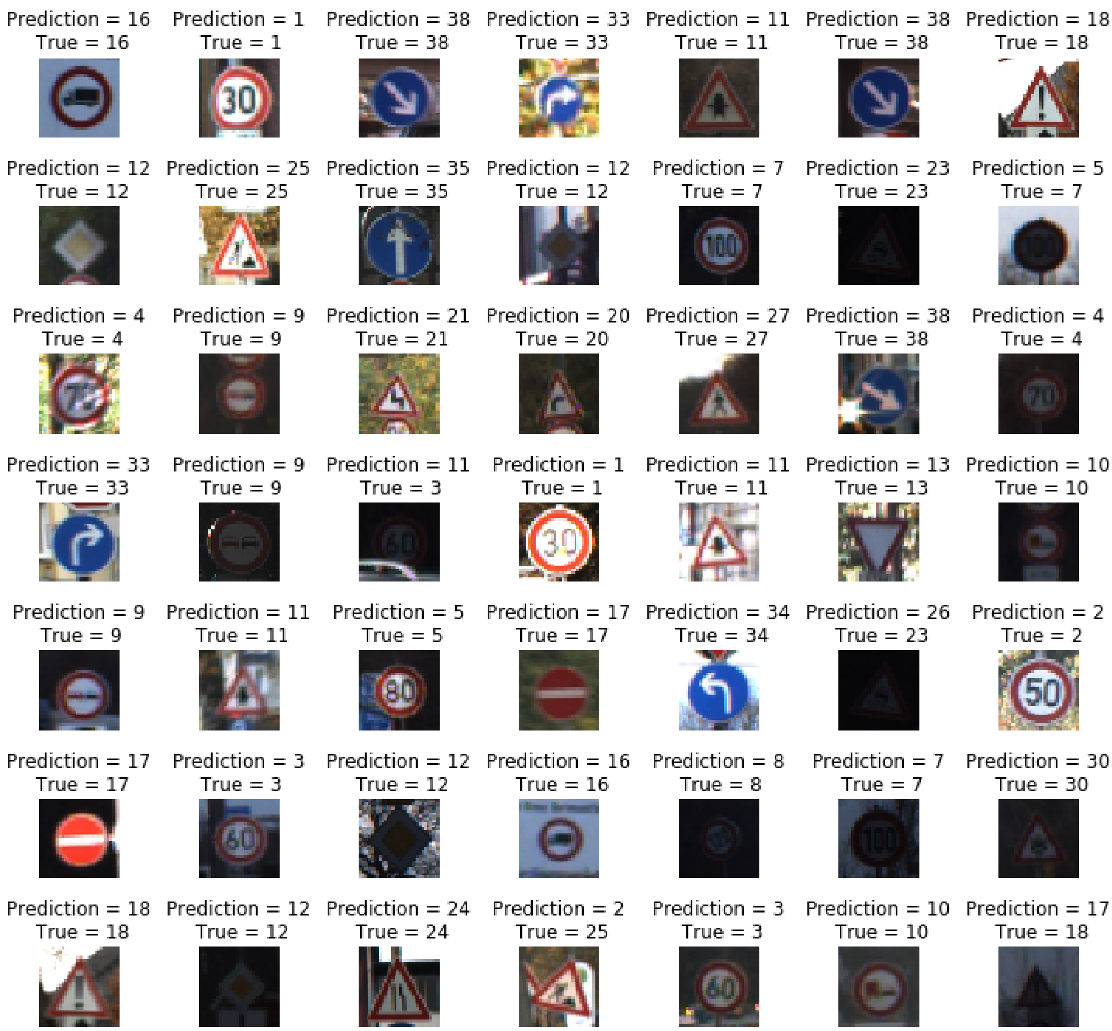
[30]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_true, predicted_classes)
plt.figure(figsize = (25, 25))
sns.heatmap(cm, annot = True)
```

[30]: <matplotlib.axes._subplots.AxesSubplot at 0x1c74d768780>



[31]:
```
L = 7
W = 7

fig, axes = plt.subplots(L, W, figsize = (12, 12))
axes = axes.ravel()

for i in np.arange(0, L*W):
    axes[i].imshow(X_test[i])
    axes[i].set_title('Prediction = {}\n True = {}'.format(predicted_classes[i],␣
 ↪y_true[i]))
    axes[i].axis('off')
```

```
plt.subplots_adjust(wspace = 1)
```



8 Skills Utilized:

1. Convolutional NN
2. LeNet
3. RELU

9 Interesting Links:

- http://setosa.io/ev/image-kernels/
- http://scs.ryerson.ca/~aharley/vis/conv/flat.html

10 Notes:

- RELU layers are used to add non-linearity in the feature map.
- Pooling or down sampling layers are placed after convolutional layers to reduce feature map dimensionality.
- This improves the computational efficiency while preserving the features.
- Pooling helps the model to generalize by avoiding overfitting.
- Improve accuracy by adding more feature detectors/filters or adding a dropout.
- Dropout refers to dropping out units in a neural network
- Dropout is a regularization technique for reducing overfitting in neural networks.
- Dropout enalbes training to occur on several architectures of the neural network.
- Neurons develop co-dependency amongst each other during training