

Step 0: Load The Data

```
In [4]: #source py3/bin/activate
# Load pickled data
import pickle

#needs to be opened by python 3 since pickle is from python 3
#remedy: tell it to use an earlier protocol(0,1 or 2)
# passing the data folder path
#/home/hoof/Desktop/SessionTwo/data/train.p'
#C:\Users\crc52_000\Dropbox\IntelligentOpenCVLessons\LessonTwo'
import os

training_file = r"C:\Users\crc52_000\Dropbox\IntelligentOpenCVLessons\lessonTw
o\train.p"
print(training_file)
testing_file = r"C:\Users\crc52_000\Dropbox\IntelligentOpenCVLessons\lessonTwo
\test.p"
print(testing_file)

with open(training_file, mode='rb') as f:
    train = pickle.load(f)
with open(testing_file, mode='rb') as f:
    test = pickle.load(f)

x_train, y_train = train['features'], train['labels']
x_test, y_test = test['features'], test['labels']

n_classes = len(set(train['labels']))
x_train_split=x_train
y_train_split=y_train

print("Training Set Total: {} samples".format(len(x_train)))

#The traffic sign data set dosen't come with data split therefore
#import sklearn model for splitting data into Learninig traininig and validati
on
from sklearn.cross_validation import train_test_split
from sklearn.metrics import confusion_matrix

#take 20% of data of the training set for testing(validation)
n_test = x_test.shape[0]
x_train_split=x_train
y_train_split=y_train
x_train,x_validation,y_train, y_validation = train_test_split(x_train_split, y
_train_split, test_size=0.2, random_state=0)

x_train = x_train.astype('float32')
x_validation = x_validation.astype('float32')
```

```
x_test= x_test.astype('float32')
x_train = x_train / 255 - 0.5
x_validation = x_validation / 255 - 0.5
x_test = x_test / 255 - 0.5

print("Updated image shape {}".format (x_train[0].shape))

print()
print("Training Set: {} samples".format(len(x_train)))
print("Validation Set: {} samples".format(len(x_validation)))
print("Test Set: {} samples".format(len(x_test)))

print("Number of classes =", n_classes)
```

```
C:\Users\crc52_000\Dropbox\IntelligentOpenCVLessons\lessonTwo\train.p
C:\Users\crc52_000\Dropbox\IntelligentOpenCVLessons\lessonTwo\test.p
Training Set Total: 39209 samples
Updated image shape (32, 32, 3)
```

```
Training Set: 31367 samples
Validation Set: 7842 samples
Test Set: 12630 samples
Number of classes = 43
```

Step 1: Dataset Summary & Exploration

```
In [5]: ### Data exploration visualization goes here.  
### Feel free to use as many code cells as needed.  
#Visualize one sample and the corresponding class to verify dataare consistant  
  
import jupyter  
  
import random  
import csv  
import math  
from tqdm import tqdm  
import cv2  
import numpy as np  
import matplotlib.pyplot as plt  
%matplotlib inline  
  
  
###  
for i in range(0, n_classes):  
    image = train['features'][train['labels'] == i][5]  
    plt.figure(figsize=(3,3))  
    plt.imshow(image)  
    plt.show()  
    print('image number', i)  
###
```

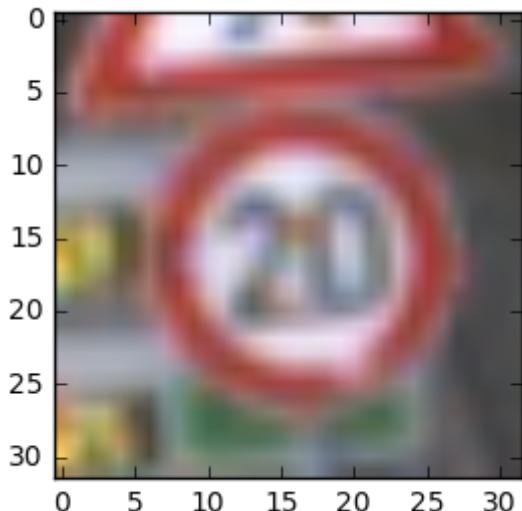


image number 0

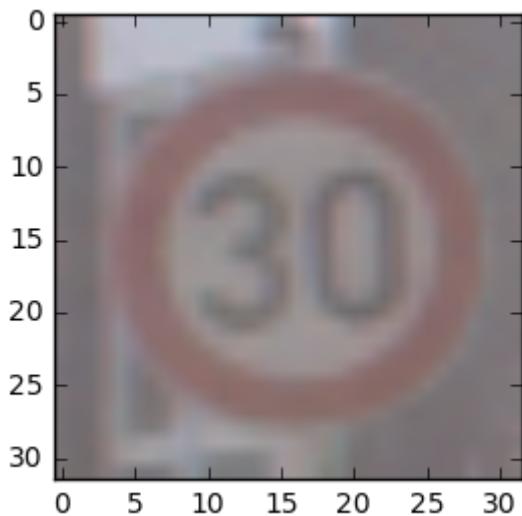


image number 1

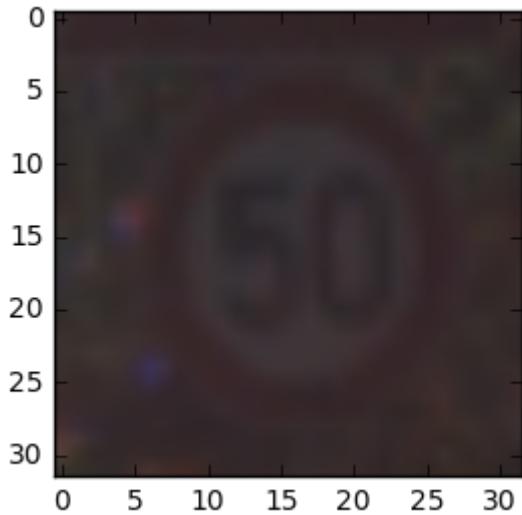


image number 2

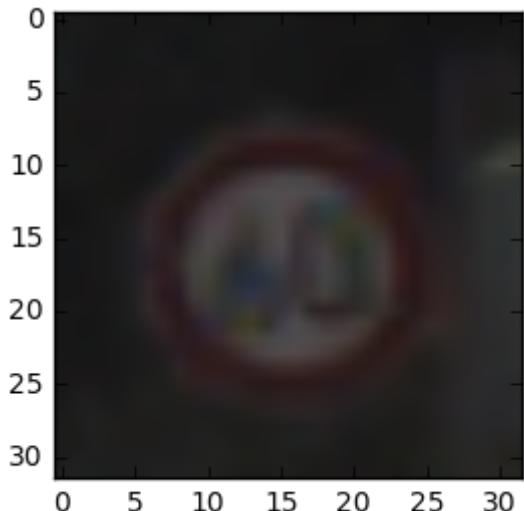


image number 3

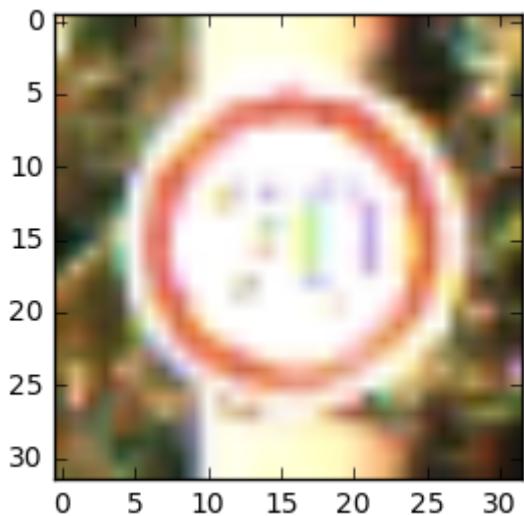


image number 4

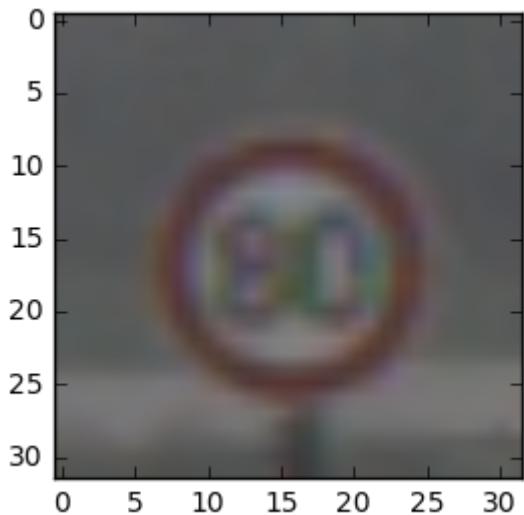


image number 5

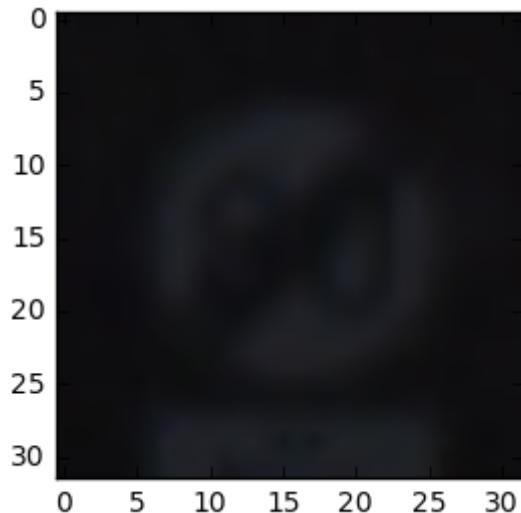


image number 6

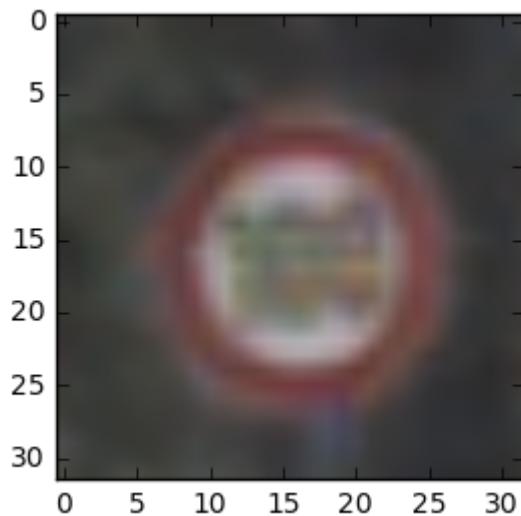


image number 7

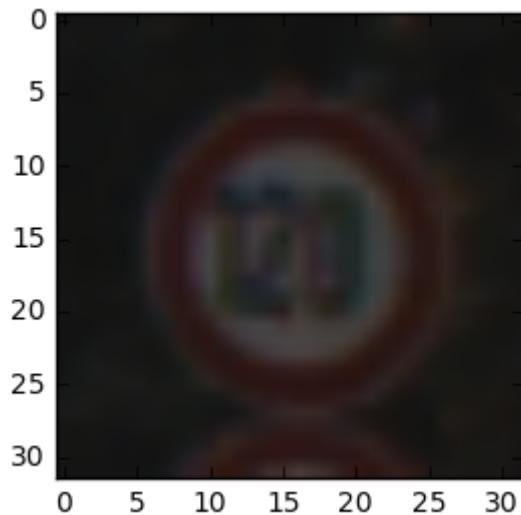


image number 8

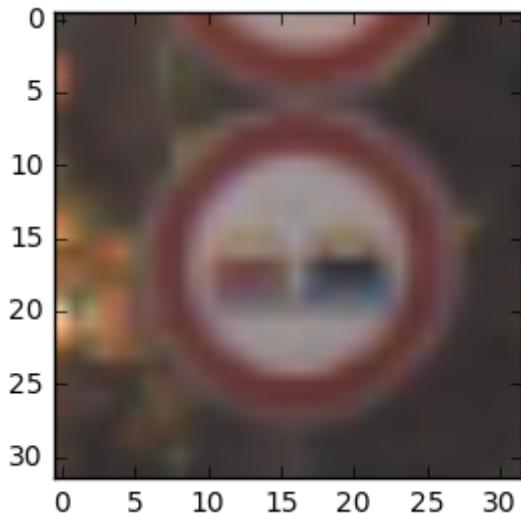


image number 9

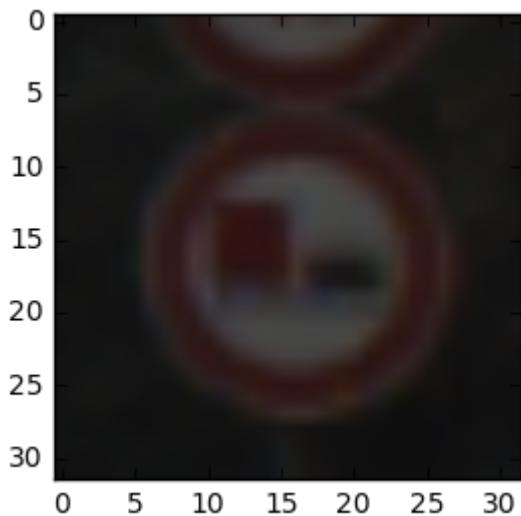


image number 10

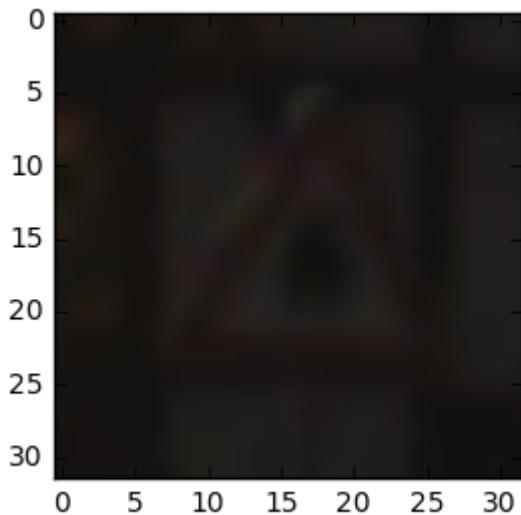


image number 11

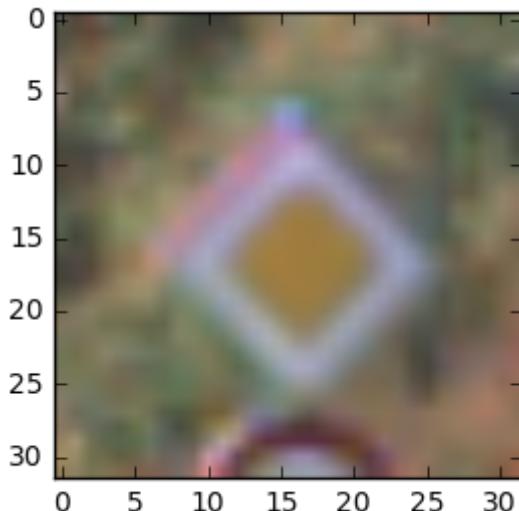


image number 12

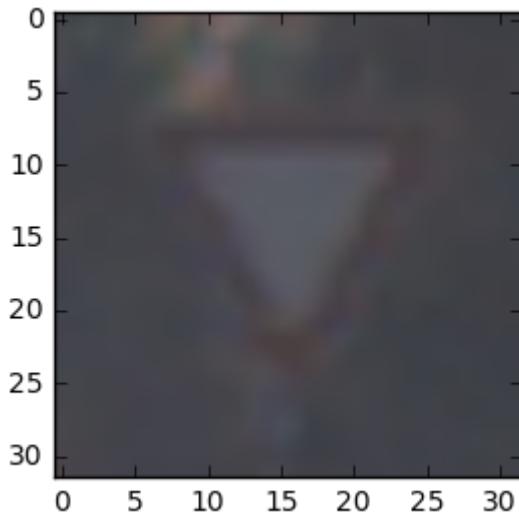


image number 13



image number 14

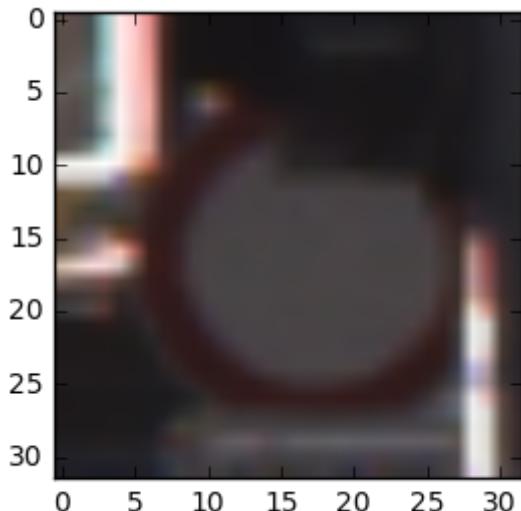


image number 15

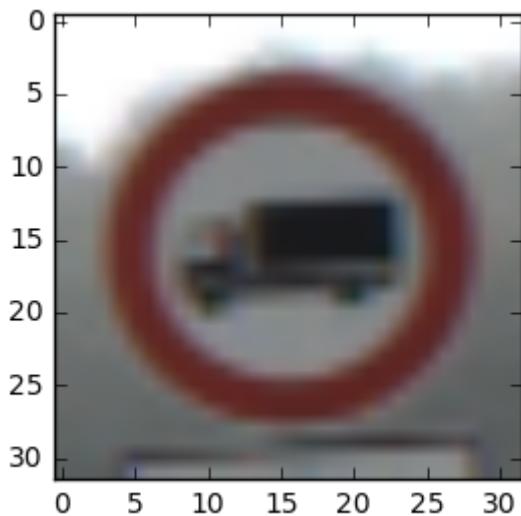


image number 16

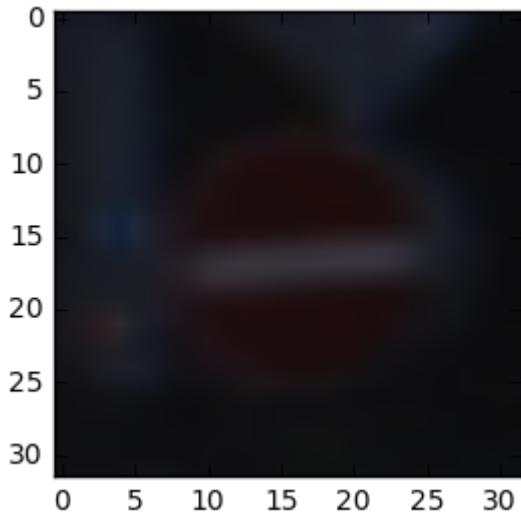


image number 17

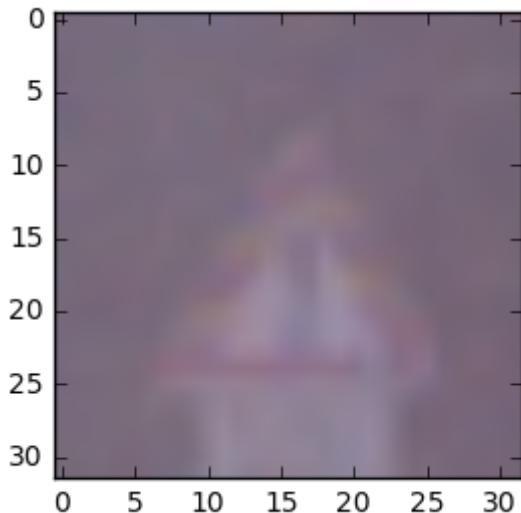


image number 18

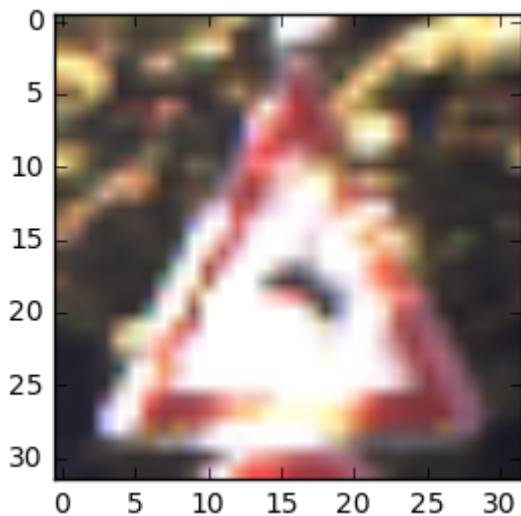


image number 19

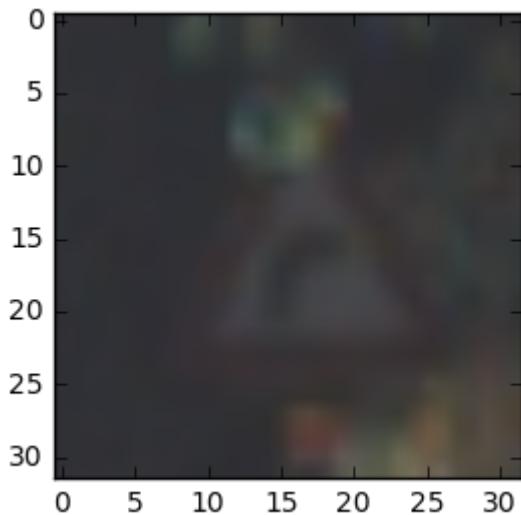


image number 20

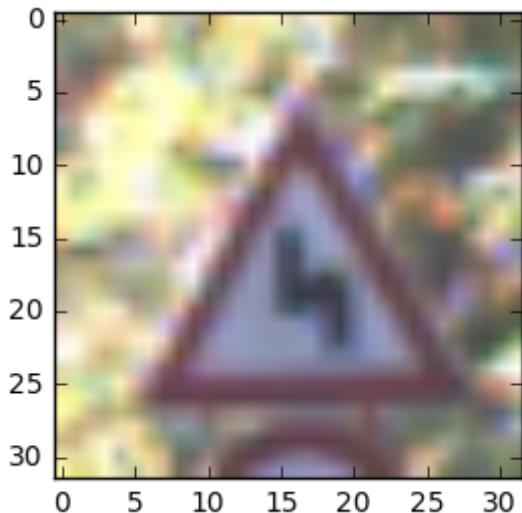


image number 21

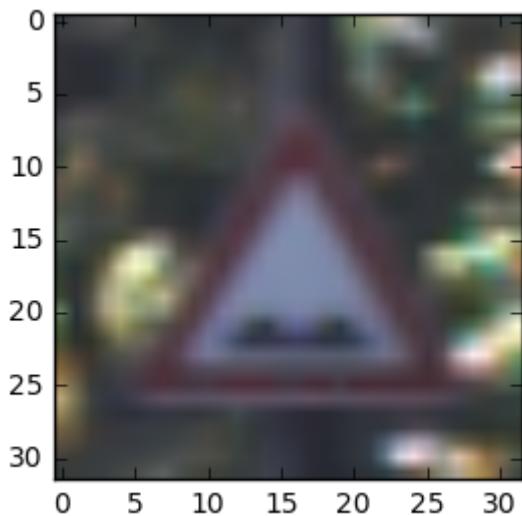


image number 22

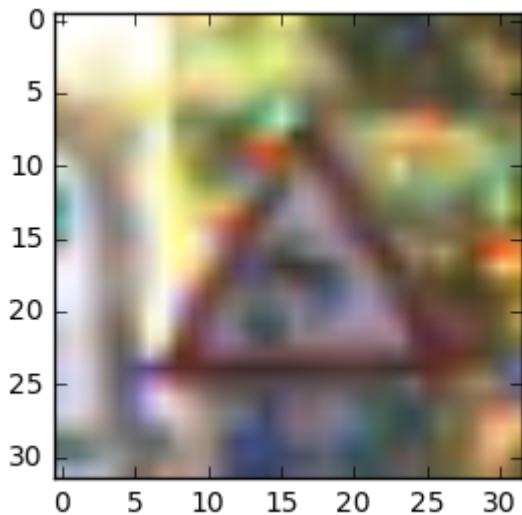


image number 23

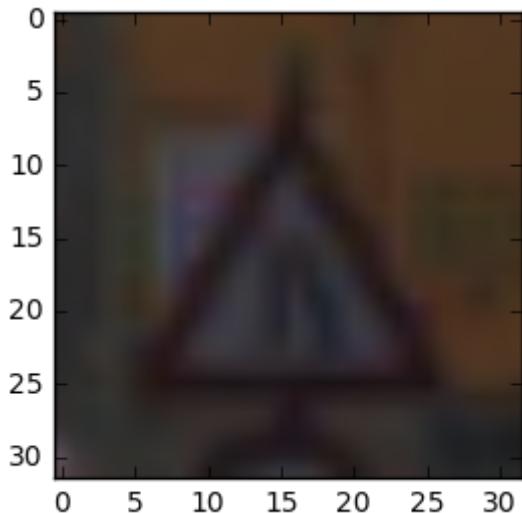


image number 24

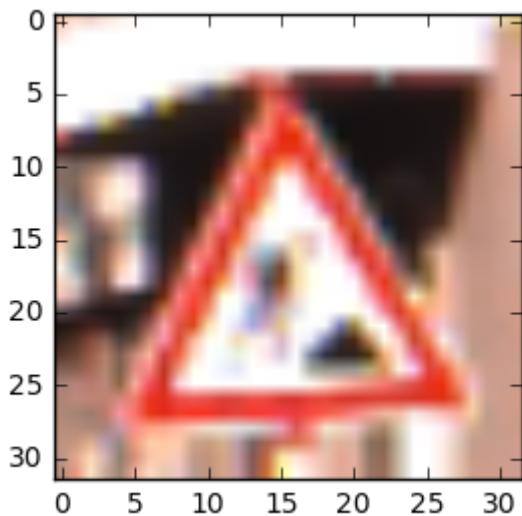


image number 25

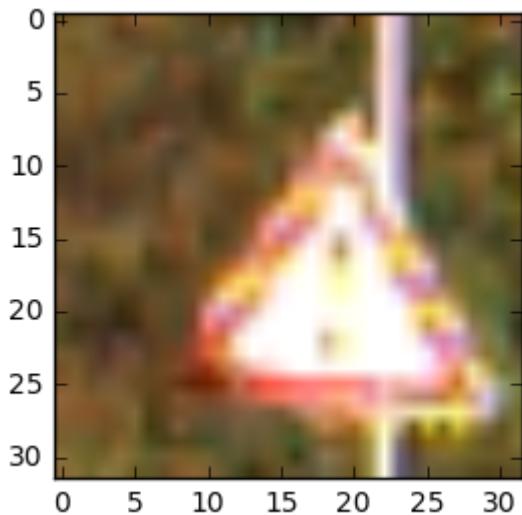


image number 26

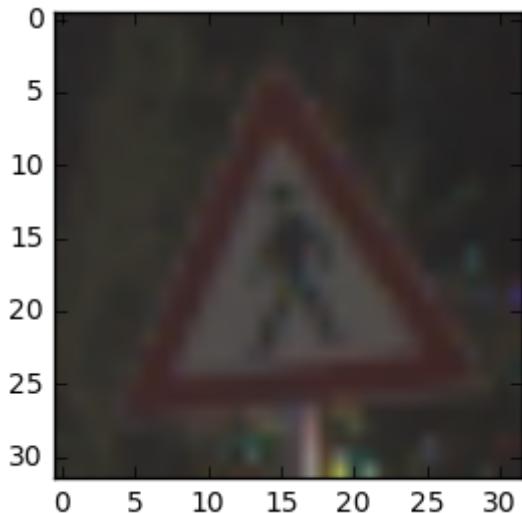


image number 27

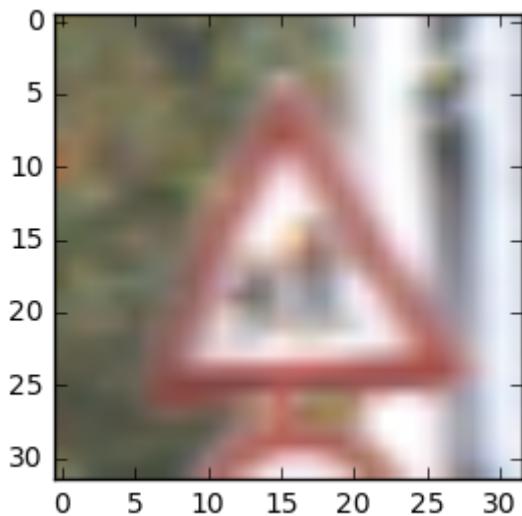


image number 28

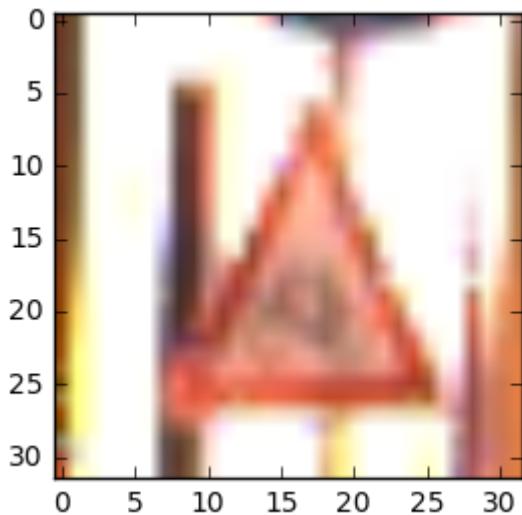


image number 29

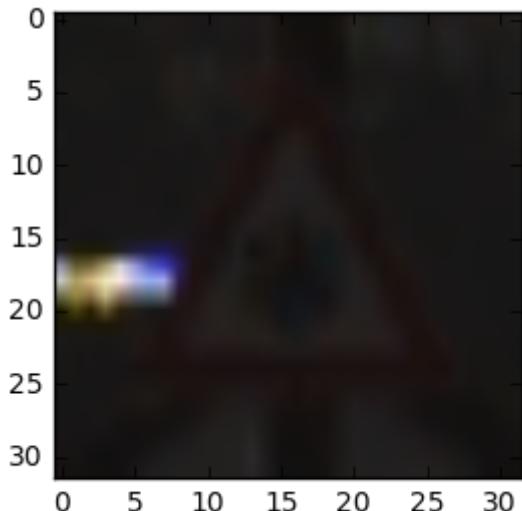


image number 30

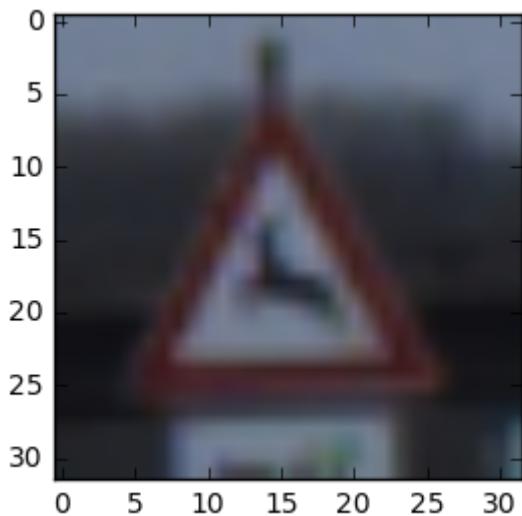


image number 31

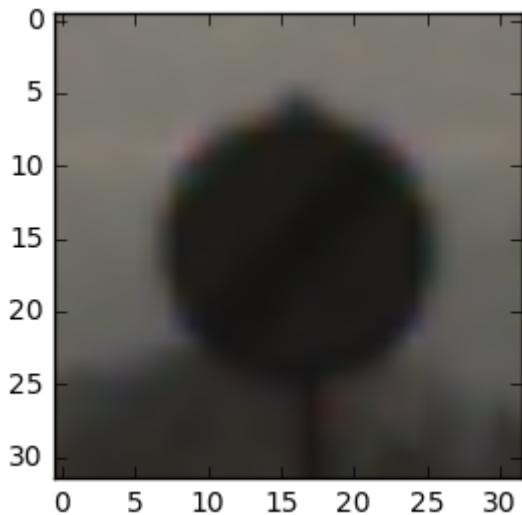


image number 32

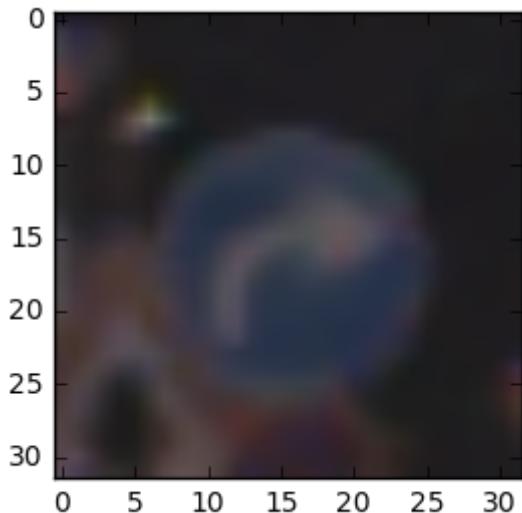


image number 33

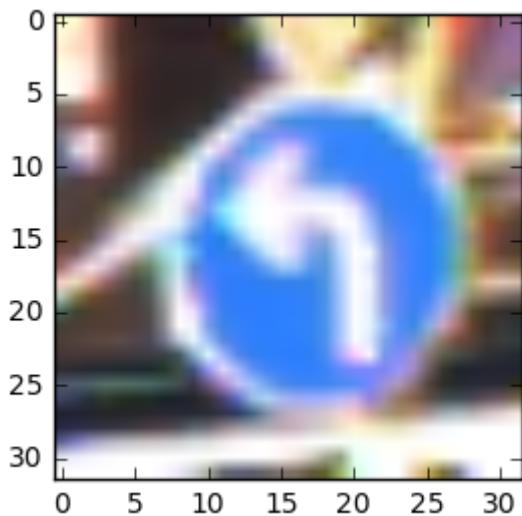


image number 34

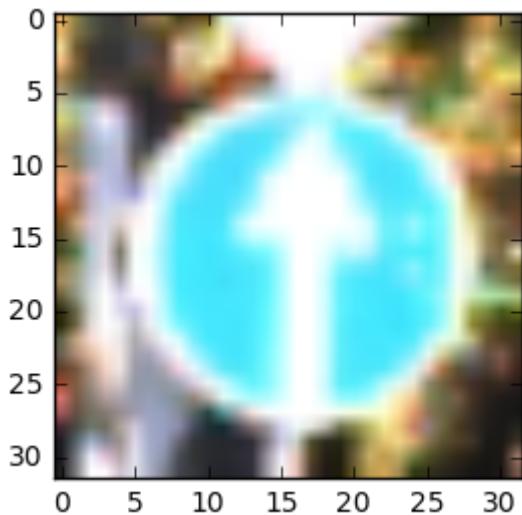


image number 35

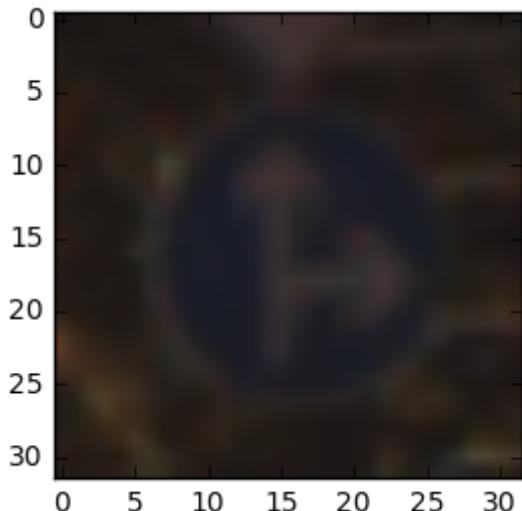


image number 36

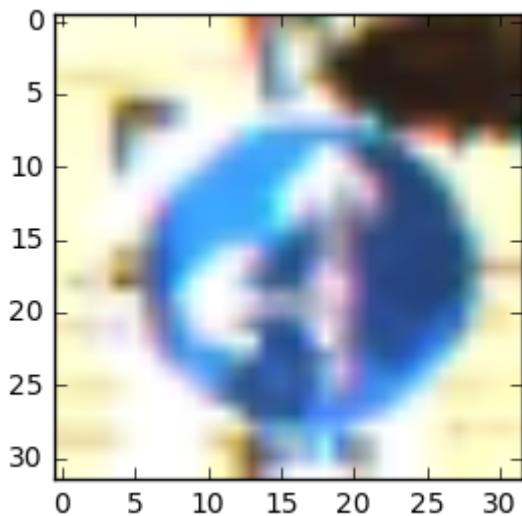


image number 37

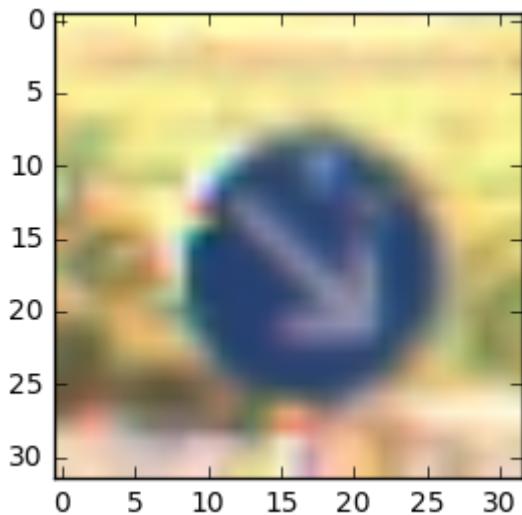


image number 38

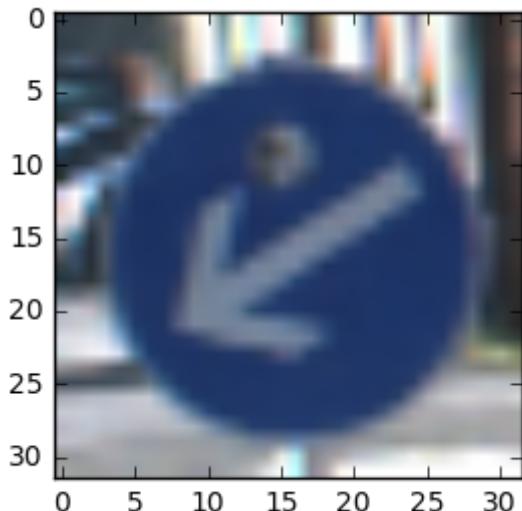


image number 39

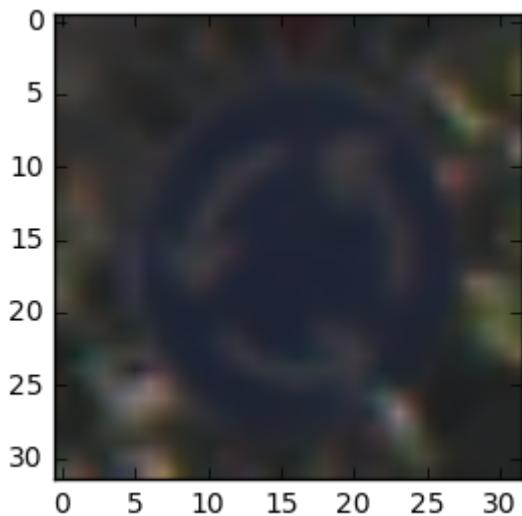


image number 40

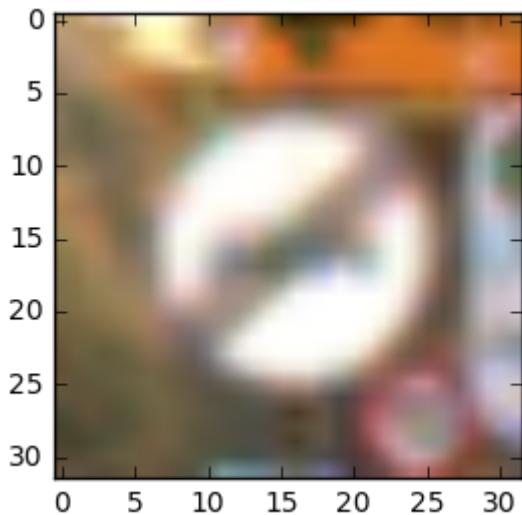


image number 41

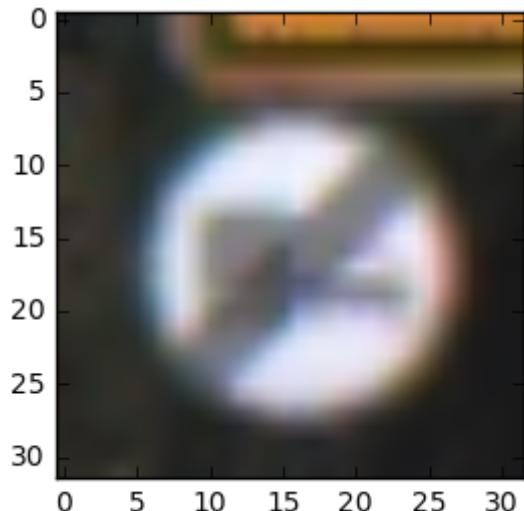


image number 42

exploratory visualization of the dataset

```
In [6]: #Plot a histogram of data
# excell file is included in the director
sign_names = []
n_samples = []

#Open excel file and read its contents for the sign labels(meaning)
with open('signnames.csv', 'r') as file:
    read_file = csv.DictReader(file)
    for i in read_file:
        sign_names.setdefault(i['ClassId'],[]).append(i['SignName'])
#Itirate through the hist

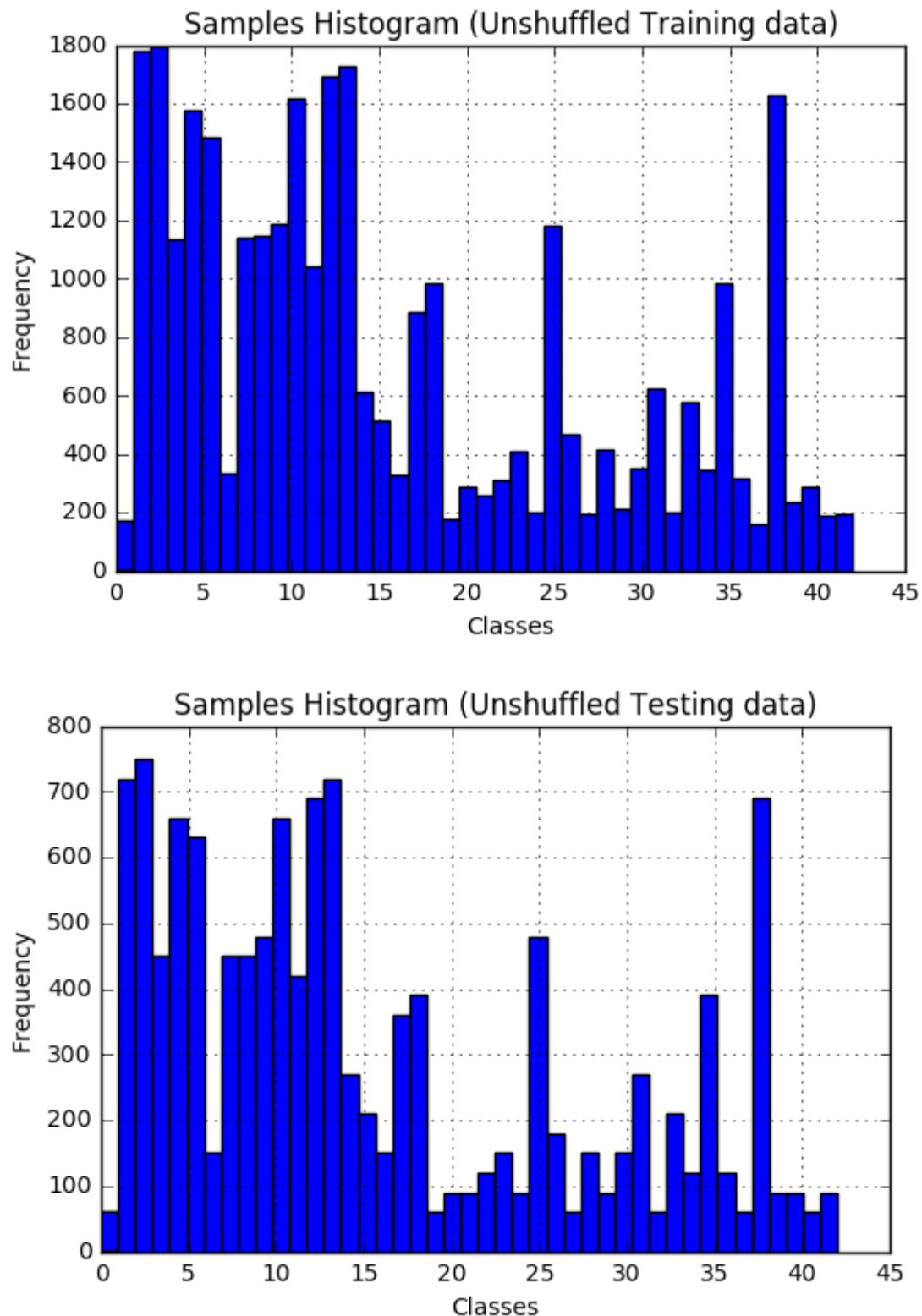
for i in range(0, n_classes):
    n_samples.append(len(train['features'][train['labels'] == i]))
    print("Count of", sign_names[str(i)], "sign is", n_samples[i], "order is",
i)

plt.figure(1)
n_class_list, _, _ = plt.hist(y_train, n_classes)
plt.title("Samples Histogram (Unshuffled Training data)")
plt.xlabel("Classes")
plt.ylabel("Frequency")
plt.grid(True)

plt.figure(2)
plt.hist(y_test, n_classes)
plt.title("Samples Histogram (Unshuffled Testing data)")
plt.xlabel("Classes")
plt.ylabel("Frequency")
plt.grid(True)

plt.show()
#Itirate through the classes using box plot
#fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(9, 4))
# rectangular box plot
#bplot1 = axes[0].boxplot(all_data,
#                         vert=True,    # vertical box alignment
#                         patch_artist=True)  # fill with color
```

```
Count of ['Speed limit (20km/h)'] sign is 210 order is 0
Count of ['Speed limit (30km/h)'] sign is 2220 order is 1
Count of ['Speed limit (50km/h)'] sign is 2250 order is 2
Count of ['Speed limit (60km/h)'] sign is 1410 order is 3
Count of ['Speed limit (70km/h)'] sign is 1980 order is 4
Count of ['Speed limit (80km/h)'] sign is 1860 order is 5
Count of ['End of speed limit (80km/h)'] sign is 420 order is 6
Count of ['Speed limit (100km/h)'] sign is 1440 order is 7
Count of ['Speed limit (120km/h)'] sign is 1410 order is 8
Count of ['No passing'] sign is 1470 order is 9
Count of ['No passing for vehicles over 3.5 metric tons'] sign is 2010 order
is 10
Count of ['Right-of-way at the next intersection'] sign is 1320 order is 11
Count of ['Priority road'] sign is 2100 order is 12
Count of ['Yield'] sign is 2160 order is 13
Count of ['Stop'] sign is 780 order is 14
Count of ['No vehicles'] sign is 630 order is 15
Count of ['Vehicles over 3.5 metric tons prohibited'] sign is 420 order is 16
Count of ['No entry'] sign is 1110 order is 17
Count of ['General caution'] sign is 1200 order is 18
Count of ['Dangerous curve to the left'] sign is 210 order is 19
Count of ['Dangerous curve to the right'] sign is 360 order is 20
Count of ['Double curve'] sign is 330 order is 21
Count of ['Bumpy road'] sign is 390 order is 22
Count of ['Slippery road'] sign is 510 order is 23
Count of ['Road narrows on the right'] sign is 270 order is 24
Count of ['Road work'] sign is 1500 order is 25
Count of ['Traffic signals'] sign is 600 order is 26
Count of ['Pedestrians'] sign is 240 order is 27
Count of ['Children crossing'] sign is 540 order is 28
Count of ['Bicycles crossing'] sign is 270 order is 29
Count of ['Beware of ice/snow'] sign is 450 order is 30
Count of ['Wild animals crossing'] sign is 780 order is 31
Count of ['End of all speed and passing limits'] sign is 240 order is 32
Count of ['Turn right ahead'] sign is 689 order is 33
Count of ['Turn left ahead'] sign is 420 order is 34
Count of ['Ahead only'] sign is 1200 order is 35
Count of ['Go straight or right'] sign is 390 order is 36
Count of ['Go straight or left'] sign is 210 order is 37
Count of ['Keep right'] sign is 2070 order is 38
Count of ['Keep left'] sign is 300 order is 39
Count of ['Roundabout mandatory'] sign is 360 order is 40
Count of ['End of no passing'] sign is 240 order is 41
Count of ['End of no passing by vehicles over 3.5 metric tons'] sign is 240 o
rder is 42
```



Design and Test a Model Architecture

```
In [8]: #=====
#-----step II-----
#=====

### Preprocess the data here.
### Feel free to use as many code cells as needed.

from sklearn.utils import shuffle

x_train, y_train = shuffle(x_train, y_train)

### Generate data additional data (OPTIONAL!)
### and split the data into training/validation/testing sets here.
### Feel free to use as many code cells as needed. Shuffling is required
### for better training the net.
```

Model Architecture

```
In [18]: =====
-----QUESTION 2-----
=====

### Define your architecture here.
### Feel free to use as many code cells as needed.

import tensorflow as tf

EPOCHS = 16
BATCH_SIZE = 150

#Define methods

from tensorflow.contrib.layers import flatten
mu = 0
sigma = 0.1
n_classes=43
layer_depth = {
    'layer_1': 6,
    'layer_2': 16,
    'fully_connected_1': 120,
    'fully_connected_2': 84,
    'fully_connected_logits': 43
}

# Store Layers weight & bias
weights = {
    'layer_1': tf.Variable(tf.truncated_normal(
        [5, 5, 3, layer_depth['layer_1']], mean = mu, stddev = sigma)),
    'layer_2': tf.Variable(tf.truncated_normal(
        [5, 5, layer_depth['layer_1'], layer_depth['layer_2']], mean = mu, stdd
ev = sigma)),
    'fully_connected_1': tf.Variable(tf.truncated_normal(
```

```
[400,layer_depth['fully_connected_1']],mean = mu, stddev = sigma)),  
    'fully_connected_2': tf.Variable(tf.truncated_normal(  
        [layer_depth['fully_connected_1'],layer_depth['fully_connected_2']],me  
an = mu, stddev = sigma)),  
    'out': tf.Variable(tf.truncated_normal(  
        [layer_depth['fully_connected_2'], n_classes],mean = mu, stddev = sigm  
a))  
}  
biases = {  
    'layer_1': tf.Variable(tf.zeros(layer_depth['layer_1'])),  
    'layer_2': tf.Variable(tf.zeros(layer_depth['layer_2'])),  
    'fully_connected_1':  
tf.Variable(tf.zeros(layer_depth['fully_connected_1'])),  
    'fully_connected_2':  
tf.Variable(tf.zeros(layer_depth['fully_connected_2'])),  
    'out': tf.Variable(tf.zeros(n_classes))  
}  
  
def conv2d(x, W, b, strides=1):  
    x = tf.nn.conv2d(x, W, strides=[1, strides, strides, 1], padding='VALID')  
    x = tf.nn.bias_add(x, b)  
    return x  
    #return tf.nn.tanh(x)  
def maxpool2d(x, k=2):  
    return tf.nn.max_pool(  
        x,  
        ksize=[1, k, k, 1],  
        strides=[1, k, k, 1],  
        padding='SAME')
```

```
In [21]: #=====
#-----QUESTION 3-----
#=====

### Train your model here.
### Feel free to use as many code cells as needed.

def LeNet(x):
    # Hyperparameters
    mu = 0
    sigma = 0.1
    dropout=0.55

    # TODO: Layer 1: Convolutional. Input = 32x32x1. Output = 28x28x6.
    conv1 = conv2d(x, weights['layer_1'], biases['layer_1'] )
    # TODO: Activation.
    conv1 = tf.nn.relu(conv1)
    # TODO: Pooling. Input = 28x28x6. Output = 14x14x6.
    conv1=maxpool2d(conv1)
    # TODO: Layer 2: Convolutional. Output = 10x10x16.
    conv2 = conv2d(conv1, weights['layer_2'], biases['layer_2'] )
    # TODO: Activation.
    conv2 = tf.nn.relu(conv2)
    # TODO: Pooling. Input = 10x10x16. Output = 5x5x16.
    conv2=maxpool2d(conv2)
    # TODO: Flatten. Input = 5x5x16. Output = 400.
```

```

        fc1 = flatten(conv2)
# TODO: Layer 3: Fully Connected. Input = 400. Output = 120.

        fc1 = tf.add(
            tf.matmul(fc1, weights['fully_connected_1']),
            biases['fully_connected_1'])
#fc1 = tf.nn.tanh(fc1)
# TODO: Activation.
        fc1 = tf.nn.relu(fc1)
# Drop_out
        fc1 = tf.nn.dropout(fc1, dropout)
# TODO: Layer 4: Fully Connected. Input = 120. Output = 84.
        fc2 = tf.add(
            tf.matmul(fc1, weights['fully_connected_2']),
            biases['fully_connected_2'])
#fc2 = tf.nn.tanh(fc2)
# TODO: Activation.
        fc2 = tf.nn.relu(fc2)
# Drop_out
        fc2 = tf.nn.dropout(fc2, dropout)
# TODO: Layer 5: Fully Connected. Input = 84. Output = 10.
        logits = tf.add(
            tf.matmul(fc2, weights['out']),
            biases['out'])
#logits = tf.nn.tanh(logits)
        return logits
    
```

#Place holder for patches, which is initialized to none in order to accept any patch size

#y stores our labels who come with sparse value

x = tf.placeholder(tf.float32, (None, 32, 32, 3))

y = tf.placeholder(tf.int32, (None))

#one hot encode the labels

one_hot_y = tf.one_hot(y, 43)

#The pipeline is as follows

#how quickly to update the network weights

rate = 0.001

logits = LeNet(x)

###

prediction = tf.nn.softmax(logits) #here i use softmax, later on i will feed the new images using this

*cross_entropy = -tf.reduce_sum(one_hot_y * tf.log(prediction + 1e-6), reduction_indices=1)*

loss = tf.reduce_mean(cross_entropy)

###

#this line will be used for test data without labels

#prediction = tf.nn.softmax(logits)

```
#train the Logits and compare them to the ground truth hot encoded Labels
#cross_entropy = tf.nn.softmax_cross_entropy_with_logits(Logits, one_hot_y)
#Cross-entropy avereging from all images
#Loss_operation = tf.reduce_mean(cross_entropy)
#Adam Optimizer to minimize the loss
optimizer = tf.train.AdamOptimizer(learning_rate = rate)
#path propagation to update the network
#training_operation = optimizer.minimize(loss_operation)

training_operation = optimizer.minimize(loss)

#This section of the code was adopted form the SDC course, traffic sign recognition
#Compare the Logit prediction to the one-hot encoded ground truth Label
correct_prediction = tf.equal(tf.argmax(logits, 1), tf.argmax(one_hot_y, 1))

#Average individual prediction accuraccies to get the overall accuracy
accuracy_operation = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
# create patches and run them
def evaluate(x_data, y_data):
    num_examples = len(x_data)
    total_accuracy = 0
    sess = tf.get_default_session()
    for offset in range(0, num_examples, BATCH_SIZE):
        batch_x, batch_y = x_data[offset:offset+BATCH_SIZE], y_data[offset:offset+BATCH_SIZE]
        accuracy = sess.run(accuracy_operation, feed_dict={x: batch_x, y: batch_y})
        total_accuracy += (accuracy * len(batch_x))
    return total_accuracy / num_examples
```

Train, Validate and Test the Model

```
In [22]: #=====
#-----QUESTION 5-----
#=====

#This section of the code was adopted from the SDC course, traffic sign recognition
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    #sess.run(tf.initialize_all_variables())
    num_examples = len(x_train)

    print("Training...")
    print()
    for i in range(EPOCHS):
        x_train, y_train = shuffle(x_train, y_train)
        for offset in range(0, num_examples, BATCH_SIZE):
            end = offset + BATCH_SIZE
            batch_x, batch_y = x_train[offset:end], y_train[offset:end]
            sess.run(training_operation, feed_dict={x: batch_x, y: batch_y})

            validation_accuracy = evaluate(x_validation, y_validation)
            print("====")
            print("EPOCH {} ...".format(i+1))
            print("Validation Accuracy after Training = {:.3f}".format(validation_accuracy))
            print()

    try:
        saver
    except NameError:
        saver = tf.train.Saver()
    saver.save(sess, './lenet')

    print("Model saved")

#Test the performance of the model on the 20% of traffic signs that we have sp
litted

with tf.Session() as sess:
    loader = tf.train.import_meta_graph('lenet.meta')
    loader.restore(sess, tf.train.latest_checkpoint('./'))

    test_accuracy = evaluate(x_test, y_test)
    len_x_test=len(x_test)
    prediction=test_accuracy * len_x_test
    print("Test Accuracy = {:.3f}".format(test_accuracy))
```

Training...

=====

EPOCH 1 ...

Validation Accuracy after Training = 0.429

=====

EPOCH 2 ...

Validation Accuracy after Training = 0.610

=====

EPOCH 3 ...

Validation Accuracy after Training = 0.732

=====

EPOCH 4 ...

Validation Accuracy after Training = 0.774

=====

EPOCH 5 ...

Validation Accuracy after Training = 0.819

=====

EPOCH 6 ...

Validation Accuracy after Training = 0.844

=====

EPOCH 7 ...

Validation Accuracy after Training = 0.851

=====

EPOCH 8 ...

Validation Accuracy after Training = 0.883

=====

EPOCH 9 ...

Validation Accuracy after Training = 0.872

=====

EPOCH 10 ...

Validation Accuracy after Training = 0.899

=====

EPOCH 11 ...

Validation Accuracy after Training = 0.902

=====

EPOCH 12 ...

Validation Accuracy after Training = 0.915

=====

EPOCH 13 ...

Validation Accuracy after Training = 0.917

=====

EPOCH 14 ...

Validation Accuracy after Training = 0.920

```
=====
EPOCH 15 ...
Validation Accuracy after Training = 0.926

=====
EPOCH 16 ...
Validation Accuracy after Training = 0.930

Model saved
```

```
-----
FailedPreconditionError                                Traceback (most recent call last)
C:\Users\crc52_000\Anaconda3\lib\site-packages\tensorflow\python\client\session.py in _do_call(self, fn, *args)
    1021     try:
-> 1022         return fn(*args)
    1023     except errors.OpError as e:
        1003
-> 1004         feed_dict, fetch_list, target_list,
    1005             status, run_metadata)

C:\Users\crc52_000\Anaconda3\lib\site-packages\tensorflow\python\client\session.py in _run_fn(session, feed_dict, fetch_list, target_list, options, run_metadata)
    1003
-> 1004     feed_dict, fetch_list, target_list,
    1005         status, run_metadata)

C:\Users\crc52_000\Anaconda3\lib\contextlib.py in __exit__(self, type, value, traceback)
    65         try:
---> 66             next(self.gen)
    67         except StopIteration:
        468             compat.as_text(pywrap_tensorflow.TF_Message(status)),
--> 469             pywrap_tensorflow.TF_GetCode(status))
    470     finally:

FailedPreconditionError: Attempting to use uninitialized value Variable_31
      [[Node: Variable_31/read = Identity[T=DT_FLOAT, _class=["loc:@Variable_31"], _device="/job:localhost/replica:0/task:0/cpu:0"](Variable_31)]]
```

During handling of the above exception, another exception occurred:

```
FailedPreconditionError                                Traceback (most recent call last)
<ipython-input-22-f0c0fd8> in <module>()
    39     loader.restore(sess, tf.train.latest_checkpoint('./'))
    40
---> 41     test_accuracy = evaluate(x_test, y_test)
    42     len_x_test=len(x_test)
    43     prediction=test_accuracy * len_x_test

<ipython-input-21-6669622ff6ec> in evaluate(x_data, y_data)
    104     for offset in range(0, num_examples, BATCH_SIZE):
    105         batch_x, batch_y = x_data[offset:offset+BATCH_SIZE], y_data[offset:offset+BATCH_SIZE]
--> 106         accuracy = sess.run(accuracy_operation, feed_dict={x:batch_x, y: batch_y})
    107         total_accuracy += (accuracy * len(batch_x))
    108     return total_accuracy / num_examples

C:\Users\crc52_000\Anaconda3\lib\site-packages\tensorflow\python\client\session.py in run(self, fetches, feed_dict, options, run_metadata)
    765     try:
    766         result = self._run(None, fetches, feed_dict, options_ptr,
--> 767                         run_metadata_ptr)
    768         if run_metadata:
    769             proto_data = tf_session.TF_GetBuffer(run_metadata_ptr)
```

```

C:\Users\crc52_000\Anaconda3\lib\site-packages\tensorflow\python\client\session.py in _run(self, handle, fetches, feed_dict, options, run_metadata)
    963      if final_fetches or final_targets:
    964          results = self._do_run(handle, final_targets, final_fetches,
--> 965                               feed_dict_string, options, run_metadata)
    966      else:
    967          results = []

C:\Users\crc52_000\Anaconda3\lib\site-packages\tensorflow\python\client\session.py in _do_run(self, handle, target_list, fetch_list, feed_dict, options, run_metadata)
   1013      if handle is None:
   1014          return self._do_call(_run_fn, self._session, feed_dict, fetch_l
ist,
-> 1015                               target_list, options, run_metadata)
   1016      else:
   1017          return self._do_call(_prun_fn, self._session, handle, feed_dic
t,

C:\Users\crc52_000\Anaconda3\lib\site-packages\tensorflow\python\client\session.py in _do_call(self, fn, *args)
   1033          except KeyError:
   1034              pass
-> 1035          raise type(e)(node_def, op, message)
   1036
   1037      def _extend_graph(self):

FailedPreconditionError: Attempting to use uninitialized value Variable_31
[[Node: Variable_31/read = Identity[T=DT_FLOAT, _class=["loc:@Variable_31"], _device="/job:localhost/replica:0/task:0/cpu:0"](Variable_31)]]
```

Caused by op 'Variable_31/read', defined at:

```

  File "C:\Users\crc52_000\Anaconda3\lib\runpy.py", line 184, in _run_module_as_main
    "__main__", mod_spec)
  File "C:\Users\crc52_000\Anaconda3\lib\runpy.py", line 85, in _run_code
    exec(code, run_globals)
  File "C:\Users\crc52_000\Anaconda3\lib\site-packages\ipykernel\__main__.py", line 3, in <module>
    app.launch_new_instance()
  File "C:\Users\crc52_000\Anaconda3\lib\site-packages\traitlets\config\application.py", line 653, in launch_instance
    app.start()
  File "C:\Users\crc52_000\Anaconda3\lib\site-packages\ipykernel\kernelapp.py", line 474, in start
    ioloop.IOLoop.instance().start()
  File "C:\Users\crc52_000\Anaconda3\lib\site-packages\zmq\eventloop\ioloop.py", line 162, in start
    super(ZMQIOLoop, self).start()
  File "C:\Users\crc52_000\Anaconda3\lib\site-packages\tornado\ioloop.py", line 887, in start
    handler_func(fd_obj, events)
  File "C:\Users\crc52_000\Anaconda3\lib\site-packages\tornado\stack_context.py", line 275, in null_wrapper
    return fn(*args, **kwargs)
  File "C:\Users\crc52_000\Anaconda3\lib\site-packages\zmq\eventloop\zmqstrea
```

```

m.py", line 440, in _handle_events
    self._handle_recv()
  File "C:\Users\crc52_000\Anaconda3\lib\site-packages\zmq\eventloop\zmqstrea
m.py", line 472, in _handle_recv
    self._run_callback(callback, msg)
  File "C:\Users\crc52_000\Anaconda3\lib\site-packages\zmq\eventloop\zmqstrea
m.py", line 414, in _run_callback
    callback(*args, **kwargs)
  File "C:\Users\crc52_000\Anaconda3\lib\site-packages\tornado\stack_context.
py", line 275, in null_wrapper
    return fn(*args, **kwargs)
  File "C:\Users\crc52_000\Anaconda3\lib\site-packages\ipykernel\kernelbase.p
y", line 276, in dispatcher
    return self.dispatch_shell(stream, msg)
  File "C:\Users\crc52_000\Anaconda3\lib\site-packages\ipykernel\kernelbase.p
y", line 228, in dispatch_shell
    handler(stream, idents, msg)
  File "C:\Users\crc52_000\Anaconda3\lib\site-packages\ipykernel\kernelbase.p
y", line 390, in execute_request
    user_expressions, allow_stdin)
  File "C:\Users\crc52_000\Anaconda3\lib\site-packages\ipykernel\ipkernel.p
y", line 196, in do_execute
    res = shell.run_cell(code, store_history=store_history, silent=silent)
  File "C:\Users\crc52_000\Anaconda3\lib\site-packages\IPython\core\interacti
veshell.py", line 2717, in run_cell
    interactivity=interactivity, compiler=compiler, result=result)
  File "C:\Users\crc52_000\Anaconda3\lib\site-packages\IPython\core\interacti
veshell.py", line 2821, in run_ast_nodes
    if self.run_code(code, result):
  File "C:\Users\crc52_000\Anaconda3\lib\site-packages\IPython\core\interacti
veshell.py", line 2881, in run_code
    exec(code_obj, self.user_global_ns, self.user_ns)
  File "<ipython-input-18-f211c03538be>", line 31, in <module>
    [5, 5, 3, layer_depth['layer_1']], mean = mu, stddev = sigma)),
  File "C:\Users\crc52_000\Anaconda3\lib\site-packages\tensorflow\python\ops
\variables.py", line 226, in __init__
    expected_shape=expected_shape)
  File "C:\Users\crc52_000\Anaconda3\lib\site-packages\tensorflow\python\ops
\variables.py", line 344, in __init_from_args
    self._snapshot = array_ops.identity(self._variable, name="read")
  File "C:\Users\crc52_000\Anaconda3\lib\site-packages\tensorflow\python\ops
\gen_array_ops.py", line 1490, in identity
    result = _op_def_lib.apply_op("Identity", input=input, name=name)
  File "C:\Users\crc52_000\Anaconda3\lib\site-packages\tensorflow\python\fram
ework\op_def_library.py", line 763, in apply_op
    op_def=op_def)
  File "C:\Users\crc52_000\Anaconda3\lib\site-packages\tensorflow\python\fram
ework\ops.py", line 2395, in create_op
    original_op=self._default_original_op, op_def=op_def)
  File "C:\Users\crc52_000\Anaconda3\lib\site-packages\tensorflow\python\fram
ework\ops.py", line 1264, in __init__
    self._traceback = _extract_stack()

```

FailedPreconditionError (see above for traceback): Attempting to use uninitialized

```
lized value Variable_31
[[Node: Variable_31/read = Identity[T=DT_FLOAT, _class=["loc:@Variable_31"], _device="/job:localhost/replica:0/task:0/cpu:0"](Variable_31)]]
```

Test a Model on New Images

```
In [27]: #=====
#-----strep III-----
#=====

import cv2
import matplotlib.pyplot as plt
import numpy as np
### Load the images and plot them here.
### Feel free to use as many code cells as needed.
real_imgs = []
for i in range(1, 11):
    print("Loading Image", i)
    img=cv2.imread('question3/test_sign_{}.jpg'.format(str(i)))
    plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    img = img / 255 - 0.5 #normalize
    resize_img = cv2.resize(img, (32, 32))
    resize_img = np.reshape(np.asarray(resize_img), [32, 32, 3])
    real_imgs.append(resize_img)
    plt.show()

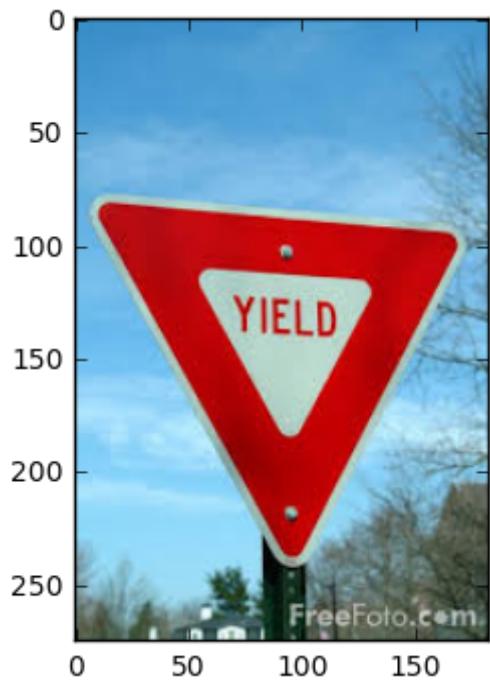
real_imgs = np.asarray(real_imgs)

print("shape of the real image", real_imgs.shape)
```

Loading Image 1



Loading Image 2



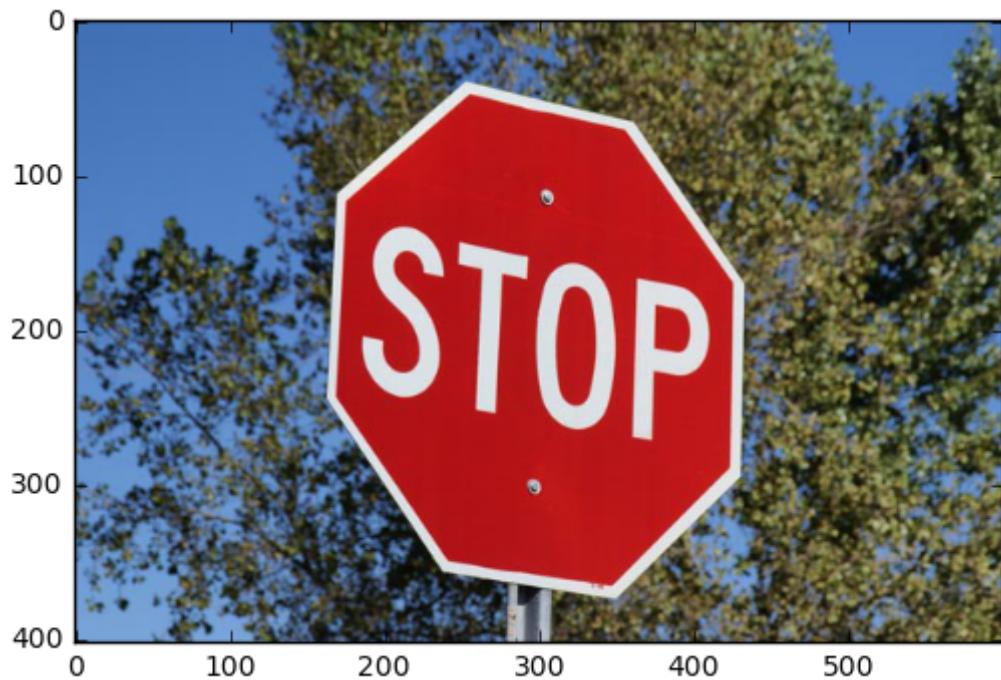
Loading Image 3



Loading Image 4



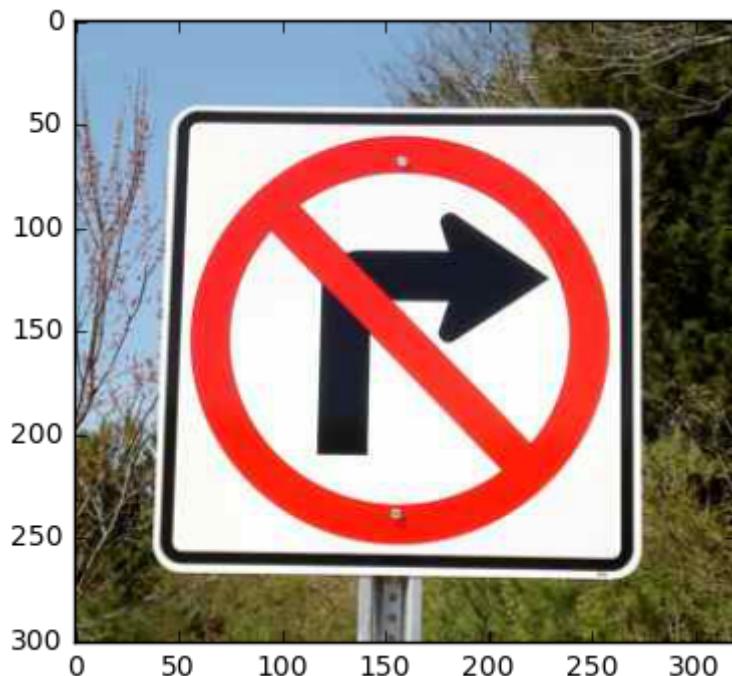
Loading Image 5



Loading Image 6



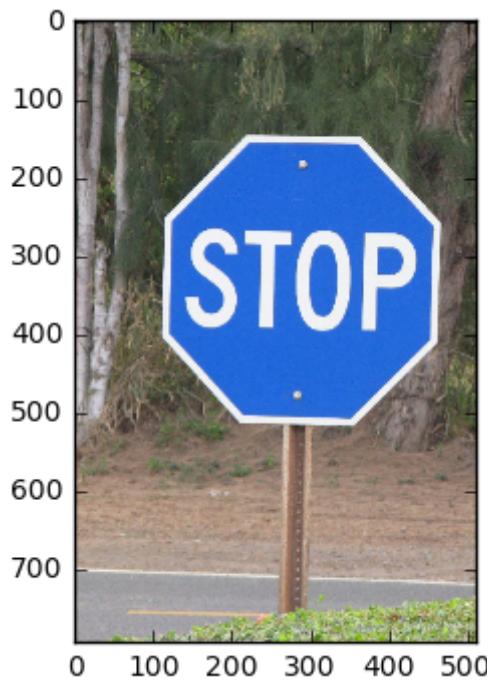
Loading Image 7



Loading Image 8



Loading Image 9



Loading Image 10



shape of the real image (10, 32, 32, 3)

```
In [28]: #=====
#-----QUESTION-----
#=====

### Run the predictions here.
### Feel free to use as many code cells as needed.
new_prediction=tf.placeholder(tf.float32,(5,32,32,3) )

#logits=LeNet(initial_real_presicions)
import tensorflow as tf
with tf.Session() as sess:
    sess.run(tf.initialize_all_variables())
#sess.run(tf.global_variables_initializer())
loader_2 = tf.train.import_meta_graph('lenet.meta')
loader_2.restore(sess, tf.train.latest_checkpoint('./'))

#new_predictions = sess.run(prediction, feed_dict={x: real_imgs})
new_prediction = sess.run(prediction, feed_dict={x: real_imgs})

print(new_prediction)
```

WARNING:tensorflow:From <ipython-input-28-65989051700f>:12: initialize_all_variables (from tensorflow.python.ops.variables) is deprecated and will be removed after 2017-03-02.

Instructions for updating:

Use `tf.global_variables_initializer` instead.

```
[[ 0.03671482  0.02919325  0.02111527  0.01815837  0.01786467  0.0215075
   0.01526415  0.02207803  0.02263631  0.0318367  0.0267596  0.02296243
   0.0235386  0.02500213  0.02266644  0.01728042  0.02959859  0.02080326
   0.01995592  0.01421396  0.03967269  0.02443181  0.01470901  0.02352062
   0.03725915  0.02524541  0.03809617  0.02921389  0.02427999  0.01833985
   0.0175899  0.02133278  0.01987774  0.01616905  0.02011097  0.01722151
   0.02120236  0.02057208  0.02286766  0.01715064  0.02326848  0.02151556
   0.02720231]
[ 0.02172348  0.02990969  0.01544542  0.01855748  0.01585369  0.02544799
   0.02598272  0.02242654  0.02254039  0.01234747  0.02314536  0.02723536
   0.02905569  0.01990118  0.02761975  0.01734525  0.0212754  0.01448636
   0.02868992  0.02149669  0.03154942  0.04288375  0.04045782  0.01816167
   0.02127687  0.01359633  0.02893662  0.02083404  0.01489956  0.02086126
   0.01838446  0.01729498  0.01214816  0.02608911  0.0267519  0.02124858
   0.02362091  0.03667639  0.01920076  0.02274912  0.02862326  0.02775129
   0.02551789]
[ 0.02205866  0.02401705  0.02128186  0.01599018  0.01154993  0.01812373
   0.02027838  0.0190915  0.03098013  0.0200903  0.01901482  0.02224778
   0.01876602  0.02290642  0.02099066  0.02825611  0.02931916  0.02781016
   0.02274844  0.01650122  0.0199861  0.03470442  0.02544828  0.02611506
   0.02294292  0.02096489  0.02388361  0.02666412  0.02290034  0.03500742
   0.02415832  0.0184981  0.01848543  0.01886946  0.01958189  0.03346284
   0.02672811  0.02872574  0.01723764  0.01696059  0.03160389  0.03006786
   0.02498046]
[ 0.02312416  0.02433879  0.01681516  0.01709806  0.02810218  0.02535337
   0.02082839  0.02416729  0.02618995  0.02142031  0.02302695  0.02233303
   0.01597845  0.01852095  0.01945081  0.02474226  0.01760252  0.02150224
   0.02357691  0.01728475  0.02597104  0.02528371  0.03403465  0.0169345
   0.02459055  0.02452289  0.02600527  0.02371883  0.02435894  0.02324088
   0.0214114  0.02338506  0.0221447  0.01720002  0.02624922  0.01901734
   0.03238143  0.02484034  0.01860522  0.03383806  0.03344392  0.02452629
   0.0228393 ]
[ 0.02032223  0.0338054  0.01774942  0.01616865  0.01662964  0.0189884
   0.01630278  0.02275525  0.03414766  0.02410509  0.03006154  0.02606126
   0.01936344  0.01988586  0.02514256  0.01710016  0.02279545  0.01293506
   0.02026235  0.01297824  0.02424275  0.02744365  0.02302392  0.03382124
   0.03208372  0.02190825  0.0283429  0.02605681  0.02229022  0.01892655
   0.01735598  0.02543142  0.02027505  0.02059888  0.02491451  0.01762039
   0.02542531  0.02215214  0.02975757  0.02834638  0.03856668  0.02254314
   0.02131202]
[ 0.0328545  0.03059108  0.01906746  0.01732737  0.02040691  0.01825154
   0.01687696  0.02328756  0.02596114  0.02038932  0.02557414  0.01979947
   0.01976253  0.01659527  0.01731371  0.02358692  0.02905926  0.02045549
   0.02589124  0.01802287  0.02148585  0.0369758  0.02295608  0.03976155
   0.02751008  0.01583521  0.02277765  0.02844103  0.02712323  0.0162875
   0.01650418  0.02125884  0.02197718  0.01813479  0.02007737  0.02148276
   0.02113121  0.02573713  0.02814507  0.02354444  0.03270603  0.02087033
   0.02820199]
[ 0.02623988  0.02498868  0.0186361  0.01963632  0.0227695  0.01523994
   0.02846611  0.01639045  0.04335777  0.01677443  0.02345765  0.0240868
   0.02405738  0.01507945  0.02753465  0.01638576  0.02279029  0.01480512
   0.0261977  0.01456644  0.02904407  0.03421014  0.01832103  0.02213219]
```

```
0.02786063 0.01295854 0.03473607 0.02874875 0.02906569 0.02059055
0.01608308 0.01670258 0.01373026 0.0311435 0.02195485 0.01842888
0.01527858 0.02816405 0.01595499 0.03876207 0.02533769 0.0411868
0.01814451]
[ 0.0259606 0.02057152 0.0243747 0.02460884 0.02559182 0.02114705
0.02395463 0.02834617 0.01985392 0.01708532 0.02218803 0.02504487
0.02393027 0.02476765 0.02393569 0.01891309 0.0245561 0.02329064
0.02426364 0.02037104 0.02046639 0.02396398 0.02823851 0.02091435
0.02147848 0.02334178 0.0358167 0.02482268 0.02036716 0.02128465
0.0194744 0.0222572 0.01998186 0.02505017 0.02325432 0.02430038
0.02152723 0.02211463 0.02319217 0.02348895 0.02500256 0.02207732
0.02482838]
[ 0.02544612 0.02320013 0.02098285 0.02238241 0.0175438 0.02314837
0.02102123 0.01921246 0.02851826 0.02208468 0.02874072 0.02003541
0.02332357 0.01538077 0.02164549 0.02016444 0.02096478 0.02592968
0.02752916 0.02150618 0.02283027 0.02188336 0.02652268 0.02603792
0.02477582 0.02298836 0.02987038 0.02788187 0.02219336 0.0156805
0.01839326 0.0234045 0.01847622 0.02504508 0.02973772 0.01778717
0.02192862 0.02498074 0.01836888 0.01860472 0.03165117 0.03903601
0.02316087]
[ 0.01889665 0.02354892 0.01950277 0.01942849 0.02248789 0.01855355
0.01939961 0.01888656 0.02815671 0.02373307 0.02535506 0.02393035
0.02303661 0.01861029 0.02666681 0.01276425 0.03003335 0.02035228
0.02630307 0.01978101 0.02630044 0.02926389 0.02472917 0.02495425
0.02845035 0.02131256 0.02699539 0.02256627 0.02330752 0.02441871
0.01780707 0.01328203 0.01918865 0.02655832 0.02283906 0.02632534
0.02537676 0.02560897 0.02059293 0.0311961 0.02093327 0.02534322
0.03322249]]
```

```
In [29]: #=====
#----- QUESTION -----
#=====

### Visualize the softmax probabilities here.
### Feel free to use as many code cells as needed.
max_value = np.argmax(new_prediction[0])
print(max_value)

max_value = np.argmax(new_prediction[1])
print(max_value)

max_value = np.argmax(new_prediction[2])
print(max_value)

max_value = np.argmax(new_prediction[3])
print(max_value)

max_value = np.argmax(new_prediction[4])
print(max_value)

max_value = np.argmax(new_prediction[5])
print(max_value)

max_value = np.argmax(new_prediction[6])
print(max_value)
```

```
20
21
29
22
40
23
8
```

```
In [30]: #=====
#-----QUESTION 9-----
#=====

with tf.Session() as sess:
    #probabilities = tf.nn.top_k(new_predictions, k=5)

    #values, indices = sess.run(probabilities)
    values, indices = sess.run(tf.nn.top_k(new_prediction, 5))
    for i in range(0, 5):
        for j in range(0, 7):
            print(sign_names[str(indices[j][i])], values[j][i])
    print("\n")

    #sess.run(tf.initialize_all_variables())
    #print(sess.run(tf.nn.top_k(new_predictions, 5), feed_dict={x: real_imgs}))
```

```
[ 'Dangerous curve to the right'] 0.0396727
[ 'Double curve'] 0.0428837
[ 'Bicycles crossing'] 0.0350074
[ 'Bumpy road'] 0.0340346
[ 'Roundabout mandatory'] 0.0385667
[ 'Slippery road'] 0.0397616
[ 'Speed limit (120km/h)'] 0.0433578

[ 'Traffic signals'] 0.0380962
[ 'Bumpy road'] 0.0404578
[ 'Double curve'] 0.0347044
[ 'Keep left'] 0.0338381
[ 'Speed limit (120km/h)'] 0.0341477
[ 'Double curve'] 0.0369758
[ 'End of no passing'] 0.0411868

[ 'Road narrows on the right'] 0.0372592
[ 'Go straight or left'] 0.0366764
[ 'Ahead only'] 0.0334628
[ 'Roundabout mandatory'] 0.0334439
[ 'Slippery road'] 0.0338212
[ 'Speed limit (20km/h)'] 0.0328545
[ 'Keep left'] 0.0387621

[ 'Speed limit (20km/h)'] 0.0367148
[ 'Dangerous curve to the right'] 0.0315494
[ 'Roundabout mandatory'] 0.0316039
[ 'Go straight or right'] 0.0323814
[ 'Speed limit (30km/h)'] 0.0338054
[ 'Roundabout mandatory'] 0.032706
[ 'Traffic signals'] 0.0347361

[ 'No passing'] 0.0318367
[ 'Speed limit (30km/h)'] 0.0299097
[ 'Speed limit (120km/h)'] 0.0309801
[ 'Speed limit (70km/h)'] 0.0281022
[ 'Road narrows on the right'] 0.0320837
[ 'Speed limit (30km/h)'] 0.0305911
[ 'Double curve'] 0.0342101
```

In []: