Traffic_Sign_Classifier

February 2, 2021

1 Self-Driving Car Engineer Nanodegree

1.1 Deep Learning

1.2 Project: Build a Traffic Sign Recognition Classifier

The goal is to build a classifier with a Deep learning technique - Convolutional Neural Network.

1.3 Step 0: Load The Data

There're three data files:

- Training dataset *train.p*
- Testing dataset *test.p*
- Validation dataset *valid.p*

These files are too big to be submitted to the repository. You can download them from here and put them into the data folder at the project's root.

```
In [76]: # Load pickled data
         import pickle

         training_file = 'data/train.p'
         validation_file= 'data/valid.p'
         testing_file = 'data/test.p'

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

         X_train, y_train = train['features'], train['labels']
         X_valid, y_valid = valid['features'], valid['labels']
         X_test, y_test = test['features'], test['labels']
```

1.4 Step 1: Dataset Summary & Exploration

The pickled data is a dictionary with 4 key/value pairs:

- `'features'` is a 4D array containing raw pixel data of the traffic sign images, (num examples, width, height, channels).
- `'labels'` is a 1D array containing the label/class id of the traffic sign. The file `signnames.csv` contains id -> name mappings for each id.
- `'sizes'` is a list containing tuples, (width, height) representing the original width and height the image.
- `'coords'` is a list containing tuples, (x1, y1, x2, y2) representing coordinates of a bounding box around the sign in the image. **THESE COORDINATES ASSUME THE ORIGINAL IMAGE. THE PICKLED DATA CONTAINS RESIZED VERSIONS (32 by 32) OF THESE IMAGES**

Complete the basic data summary below. Use python, numpy and/or pandas methods to calculate the data summary rather than hard coding the results. For example, the pandas shape method might be useful for calculating some of the summary results.

1.4.1 Basic Summary of the Data Set

```
In [77]: import numpy as np

         # Number of training examples
         n_train = y_train.size

         # Number of validation examples
         n_validation = y_valid.size

         # Number of testing examples
         n_test = y_test.size

         # Shape of an traffic sign image
         image_shape = X_train[0].shape

         # Number of unique classes/labels in the dataset
         n_classes = np.unique(train['labels']).size

         print("Number of training examples =", n_train)
         print("Number of testing examples =", n_test)
         print("Image data shape =", image_shape)
         print("Number of classes =", n_classes)
```

```
Number of training examples = 34799
Number of testing examples = 12630
Image data shape = (32, 32, 3)
Number of classes = 43
```

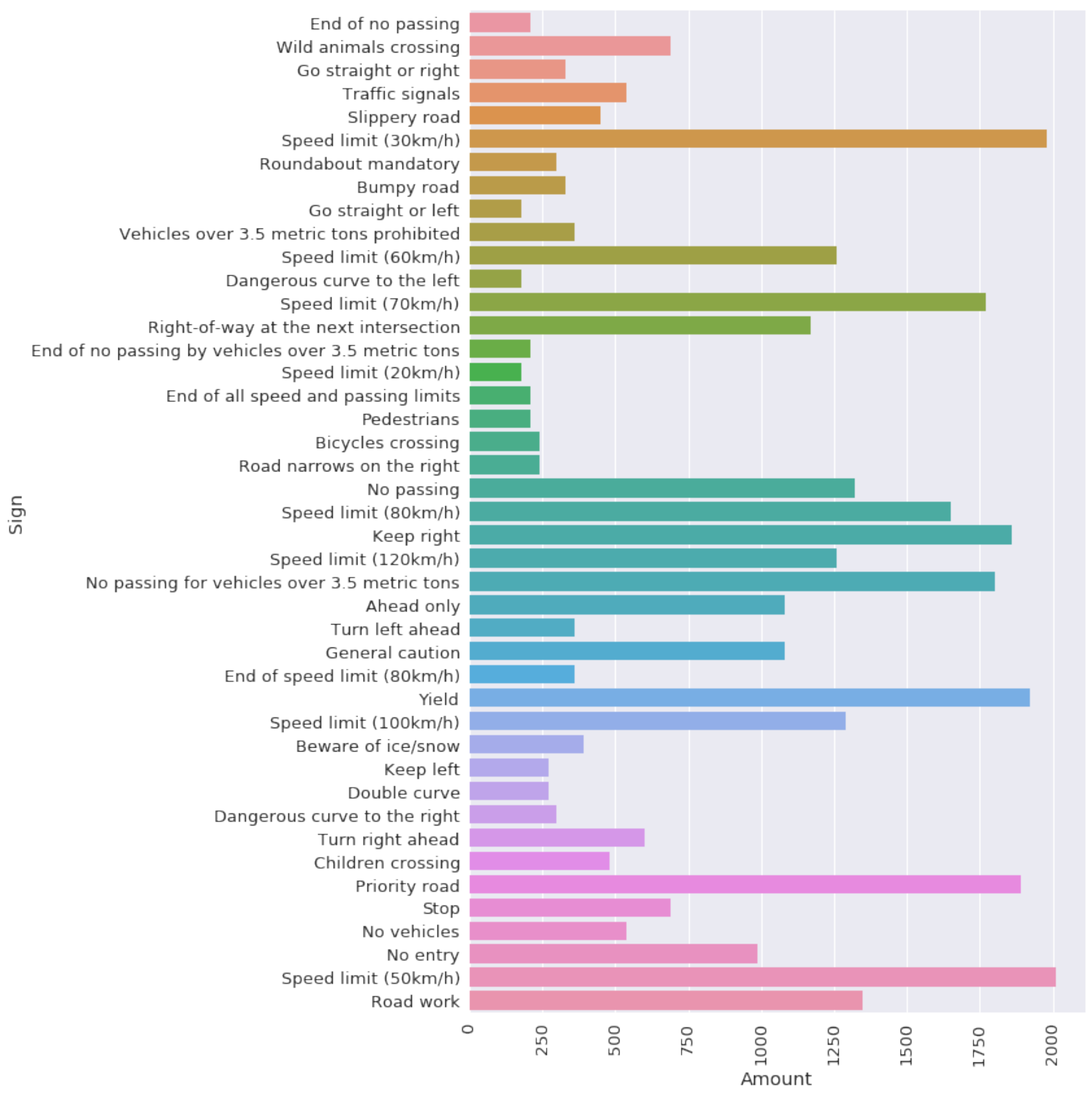
1.4.2 Exploratory visualization of the dataset

Import of needed libraries

```
In [78]: from collections import Counter

         import pandas as pd
         import matplotlib.pyplot as plt
         import seaborn as sb

         %matplotlib inline
```

Distribution of signs in the dataset

```
In [79]: counter = Counter(y_train)
         sign_names = pd.read_csv('./signnames.csv')
         sign_names.set_index('ClassId',inplace=True)

         counts = pd.DataFrame(columns=['sign_label','count'],
                               data=[(label, count) for label, count in counter.items()])
         counts['sign'] = counts.sign_label.apply(lambda l: sign_names.loc[l].SignName)

         plt.figure(figsize=(12, 12))
         sb.set(font_scale=1.3)
         sb.barplot(x='count', y='sign',
                    data=counts, orient='h')
         plt.xticks(rotation=90)
         plt.ylabel('Sign')
         plt.xlabel('Amount')
         plt.tight_layout()
         plt.savefig('output/number_of_signs.png')
```



Examples

```
In [80]: columns = 10
         rows = 10

         signs = []
         for l in range(rows):
             found = 0
             while found < columns:
                 id = np.random.randint(len(y_train))
                 if l == y_train[id]:
                     signs.append(id)
                     found += 1
```

```
plt.figure(figsize=(10, 10))
count = 0
plt.tight_layout()
for s in signs:
    count += 1
    plt.subplot(rows, columns, count)
    plt.imshow(X_train[s])
    plt.axis('off')

plt.savefig('output/examples.png', bbox_inches='tight')
```

1.5 Step 2: Design and Test a Model Architecture

Design and implement a deep learning model that learns to recognize traffic signs. Train and test your model on the German Traffic Sign Dataset.

The LeNet-5 implementation shown in the classroom at the end of the CNN lesson is a solid starting point. You'll have to change the number of classes and possibly the preprocessing, but aside from that it's plug and play!

With the LeNet-5 solution from the lecture, you should expect a validation set accuracy of about 0.89. To meet specifications, the validation set accuracy will need to be at least 0.93. It is possible to get an even higher accuracy, but 0.93 is the minimum for a successful project submission.

There are various aspects to consider when thinking about this problem:

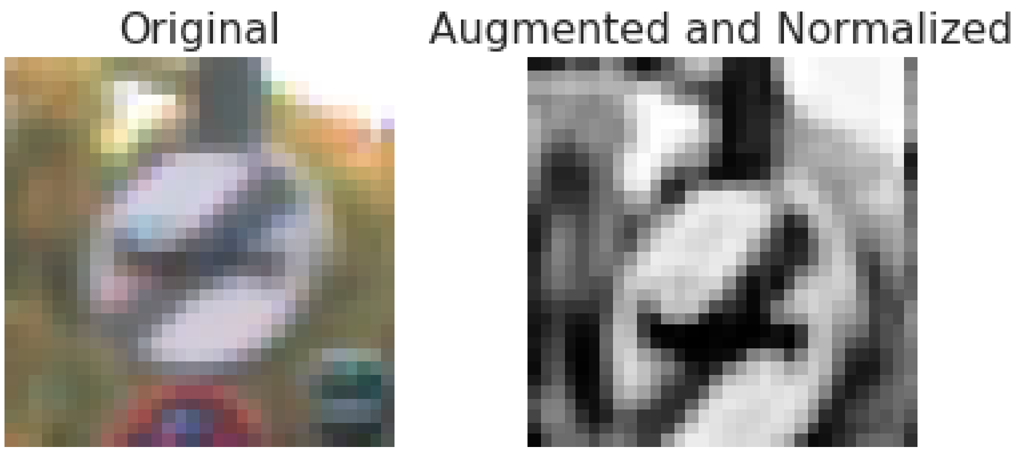
- Neural network architecture (is the network over or underfitting?)
- Play around preprocessing techniques (normalization, rgb to grayscale, etc)
- Number of examples per label (some have more than others).
- Generate fake data.

Here is an example of a published baseline model on this problem. It's not required to be familiar with the approach used in the paper but, it's good practice to try to read papers like these.

1.5.1 Pre-process the Data Set

```
In [81]: import cv2

In [82]: def normalize(img):
             img_in = img.copy()
             gb = cv2.GaussianBlur(img_in, (5,5), 20.0)
             img_in = cv2.addWeighted(img_in, 2, gb, -1, 0)
             img_in = cv2.cvtColor(img_in, cv2.COLOR_RGB2YUV)
             img_in[:,:,0] = cv2.equalizeHist(img_in[:,:,0])
             return img_in[:,:,0]

         def augment(img):
             img = img.copy()

             s = 1.8*np.random.rand() + 0.2
             m = 127.0*(1.0 - s)
             img = cv2.multiply(img, np.array([s]))
             img = cv2.add(img, np.array([m]))

             c_x, c_y = int(img.shape[0]/2), int(img.shape[1]/2)
             a = 30.0*np.random.rand() - 15
             m_rot = cv2.getRotationMatrix2D((c_x, c_y), a, 1.0)
             img = cv2.warpAffine(img, m_rot, img.shape[:2])

             sc_y = 0.4*np.random.rand() + 1.0
             img_s = cv2.resize(img, None, fx=1, fy=sc_y,
                              interpolation=cv2.INTER_CUBIC)
            dy = int((img_s.shape[1] - img.shape[0])/2)
            end = img.shape[1] - dy
            return img_s[dy:end,:,:]
```

```
In [83]: img = X_train[100]
         plt.subplot(1,2,1)
         plt.imshow(img)
         plt.title('Original')
         plt.axis('off')
         plt.subplot(1,2,2)
         plt.axis('off')
         out = normalize(augment(img))
         plt.imshow(out, cmap='gray')
         plt.title('Augmented and Normalized')
         plt.tight_layout()
         plt.savefig('output/augmented.png', bbox_inches='tight')
```



```
In [84]: import os.path
         from sklearn.utils import shuffle

         def prepare_testing_data(augment_ratio):
             file_name = 'data/augmented.p'

             if os.path.isfile(file_name):
                 print('Return stored training data')
                 with open(file_name, mode='rb') as f:
                     train = pickle.load(f)
                 return train['images'], train['labels']
```

```
            print('Prepare augmented data')

            X_augmented = []
            y_augmented = []

            for i in range(X_train.shape[0]):
                img = X_train[i]
                label = y_train[i]
                img_out = normalize(img)
                img_out.shape = img_out.shape + (1,)
                X_augmented.append(img_out)
                y_augmented.append(label)
                for j in range(augment_ratio):
                    img_aug = normalize(augment(img))
                    img_aug.shape = img_aug.shape + (1,)
                    X_augmented.append(img_aug)
                    y_augmented.append(label)

            X_augmented = np.array(X_augmented)
            y_augmented = np.array(y_augmented)

            data = {
                'images': X_augmented,
                'labels': y_augmented
            }

            with open(file_name, 'wb') as f:
                pickle.dump(data, f, protocol=pickle.HIGHEST_PROTOCOL)

            return X_augmented, y_augmented

        X_augmented, y_augmented = prepare_testing_data(5)
        print(f"Number of tesing augmented images: {y_augmented.size}")
```

```
Return stored training data
Number of tesing augmented images: 208794
```

```
In [85]: X_valid_normalized = []
         X_test_normalized = []

         def normalize_data(data):
             out = []
             for i in range(data.shape[0]):
                 img = data[i]
                 img = normalize(img)
                 img.shape = img.shape + (1,)
                 out.append(img)
```

```
                return out

            X_valid = normalize_data(X_valid)
            X_test = normalize_data(X_test)
```

1.5.2 Model Architecture

```
In [86]: import tensorflow as tf
         from tensorflow.contrib.layers import flatten

In [87]: def logits(x):
             def random_weights(s):
                 mu = 0
                 sigma = 0.1
                 return tf.Variable(tf.truncated_normal(shape=(s), mean=mu, stddev=sigma))

             def zeros(n):
                 return tf.Variable(tf.zeros(n))

             # Layer 1: 32x32x1 => 28x28x6
             W1 = random_weights((5, 5, 1, 6))
             b1 = zeros(6)
             x = tf.nn.conv2d(x, W1, strides=[1, 1, 1, 1], padding='VALID')
             x = tf.nn.bias_add(x, b1)
             x = tf.nn.relu(x)

             # Pooling: 28x28x6 => 14x14x6
             x = tf.nn.max_pool(x, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='VALID')

             # Layer 2: 14x14x6 => 10x10x16
             W2 = random_weights((5, 5, 6, 16))
             b2 = zeros(16)
             x = tf.nn.conv2d(x, W2, strides=[1, 1, 1, 1], padding='VALID')
             x = tf.nn.bias_add(x, b2)
             x = tf.nn.relu(x)

             # Pooling: 10x10x16 => 5x5x16
             x = tf.nn.max_pool(x, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='VALID')
             l2 = x

             # Layer 3: 5x5x16 => 1x1x400
             W3 = random_weights((5, 5, 16, 400))
             b3 = zeros(400)
             x = tf.nn.conv2d(x, W3, strides=[1, 1, 1, 1], padding='VALID')
             x = tf.nn.bias_add(x, b3)
             x = tf.nn.relu(x)
             l3 = x
```

```
    # Flatten: 5x5x16 => 400
    l2 = flatten(l2)

    # Flatten: 1x1x400 => 400
    l3 = flatten(l3)

    x = tf.concat([l3, l2], 1)
    x = tf.nn.dropout(x, keep_prob)

    # Layer 4: 800 => 120
    W4 = random_weights((800, 120))
    b4 = zeros(120)
    x = tf.add(tf.matmul(x, W4), b4)
    x = tf.nn.relu(x)

    # Layer 5: 120 => 84
    W5 = random_weights((120, 84))
    b5 = zeros(84)
    x = tf.add(tf.matmul(x, W5), b5)
    x = tf.nn.relu(x)

    # Layer 6: 84 => 43
    W6 = random_weights((84, 43))
    b6 = zeros(43)
    x = tf.add(tf.matmul(x, W6), b6)

    return x

x = tf.placeholder(tf.float32, (None, 32, 32, 1))
y = tf.placeholder(tf.int32, (None))
keep_prob = tf.placeholder(tf.float32)
one_hot_y = tf.one_hot(y, 43)
rate = 0.0001
cnn = logits(x)
cross_entropy = tf.nn.softmax_cross_entropy_with_logits(logits=cnn,
                                                        labels=one_hot_y)
loss_operation = tf.reduce_mean(cross_entropy)
optimizer = tf.train.AdamOptimizer(learning_rate = rate)
training_operation = optimizer.minimize(loss_operation)
predict_operation = tf.argmax(cnn, 1)
predict_proba_operation = tf.nn.softmax(logits=cnn)
correct_prediction = tf.equal(tf.argmax(cnn, 1), tf.argmax(one_hot_y, 1))
accuracy_operation = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))

def evaluate(X_data, y_data):
    num_examples = len(X_data)
    total_accuracy = 0
    sess = tf.get_default_session()
```

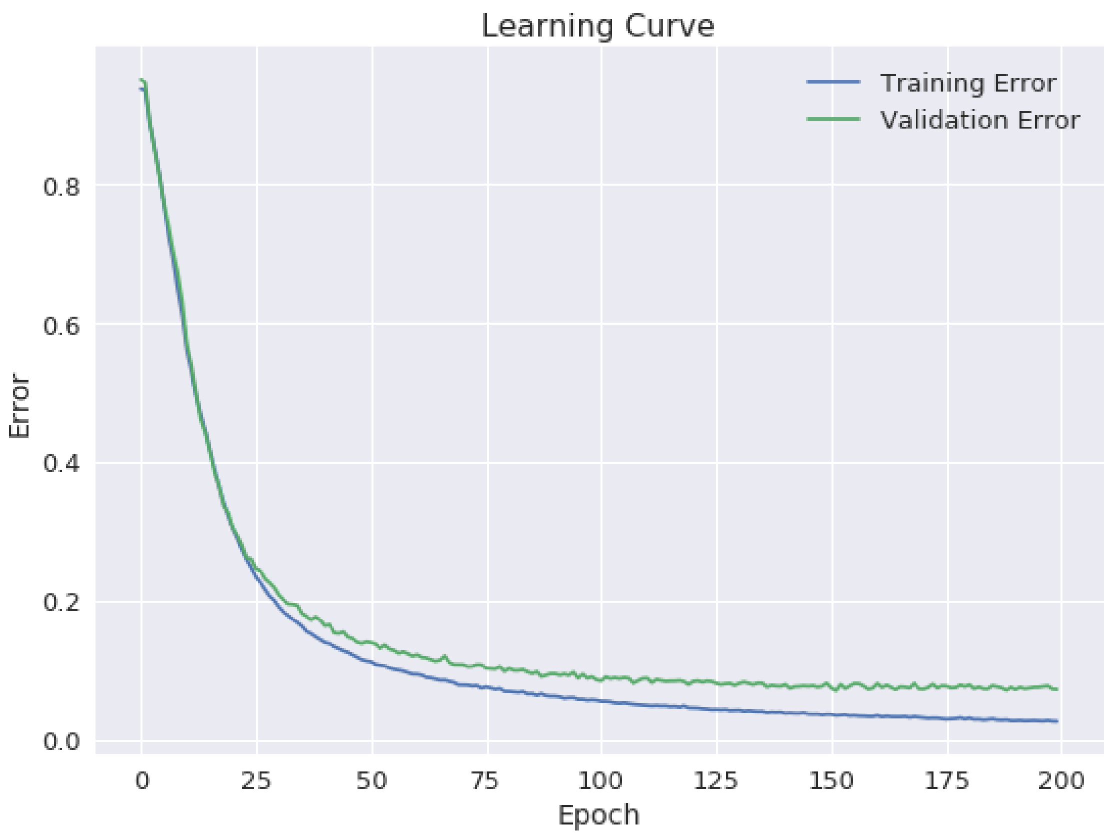
```
        for offset in range(0, num_examples, BATCH_SIZE):
            batch_x, batch_y = X_data[offset:offset+BATCH_SIZE], y_data[offset:offset+BATCH
            accuracy = sess.run(accuracy_operation,
                                feed_dict={
                                    x: batch_x,
                                    y: batch_y,
                                    keep_prob: 1.0
                                })
            total_accuracy += (accuracy * len(batch_x))
        return total_accuracy / num_examples


    def predict(X_data):
        num_examples = len(X_data)
        sess = tf.get_default_session()
        predicted_proba = list()
        for offset in range(0, num_examples, BATCH_SIZE):
            batch_x = X_data[offset:offset+BATCH_SIZE]
            predicted_proba.extend(sess.run(predict_proba_operation,
                                            feed_dict={
                                                x: batch_x,
                                                keep_prob: 1.0
                                            }
                                           )
                                  )
        return predicted_proba

    EPOCHS = 200
    BATCH_SIZE = 128
    dropout = 0.2

    errors = []
    saver = tf.train.Saver()
```

```
In [88]: with tf.Session() as sess:
             sess.run(tf.global_variables_initializer())
             num_examples = len(X_train)
             for i in range(EPOCHS):
                 X_augmented, y_augmented = shuffle(X_augmented, y_augmented)
                 for offset in range(0, num_examples, BATCH_SIZE):
                     end = offset + BATCH_SIZE
                     batch_x, batch_y = X_augmented[offset:end], y_augmented[offset:end]
                     sess.run(training_operation, feed_dict={
                         x: batch_x,
                         y: batch_y,
                         keep_prob: 1 - dropout
                     })
```

```
                training_accuracy = evaluate(X_augmented, y_augmented)
                validation_accuracy = evaluate(X_valid, y_valid)
                errors.append((training_accuracy,validation_accuracy))
                if i % 20 == 0:
                    print(f"Epoch {i}")
                    print(f"Training error = {1 - training_accuracy}")
                    print(f"Validation error = {1 - validation_accuracy}")
            saver.save(sess, './models/lenet')

        print("Training done")
```

```
Epoch 0
Training error = 0.9362529574604634
Validation error = 0.9496598639455782
Epoch 20
Training error = 0.30205848826445536
Validation error = 0.3024943313090439
Epoch 40
Training error = 0.13927124341740016
Validation error = 0.16326530714964915
Epoch 60
Training error = 0.09358985410601772
Validation error = 0.12176870742892998
Epoch 80
Training error = 0.06901539315464555
Validation error = 0.10204081681309918
Epoch 100
Training error = 0.05472379474505973
Validation error = 0.0841269848838685
Epoch 120
Training error = 0.045269500077423364
Validation error = 0.08344671277502502
Epoch 140
Training error = 0.037271185949622754
Validation error = 0.07891156511241892
Epoch 160
Training error = 0.03414849085302041
Validation error = 0.08027210960042175
Epoch 180
Training error = 0.03046543482650621
Validation error = 0.07845805037318987
Training done
```

```
In [89]: plt.figure(figsize=(8,6))
         plt.title('Learning Curve')
         plt.plot([1 - e[0] for e in errors])
         plt.plot([1 - e[1] for e in errors])
plt.legend(['Training Error', 'Validation Error'])
plt.tight_layout()
plt.savefig('output/learning_curve.png')
plt.ylabel('Error')
plt.xlabel('Epoch');
```



```
In [90]: with tf.Session() as sess:
             saver.restore(sess, tf.train.latest_checkpoint('./models'))
             print('Accuracy On Training Images: {:.2f}'.format(evaluate(X_augmented,y_augmented
             print('Accuracy On Validation Images: {:.2f}'.format(evaluate(X_valid,y_valid)))
             print('Accuracy On Test Images: {:.2f}'.format(evaluate(X_test,y_test)))
```

```
INFO:tensorflow:Restoring parameters from ./models/lenet
Accuracy On Training Images: 0.97
Accuracy On Validation Images: 0.93
Accuracy On Test Images: 0.91
```

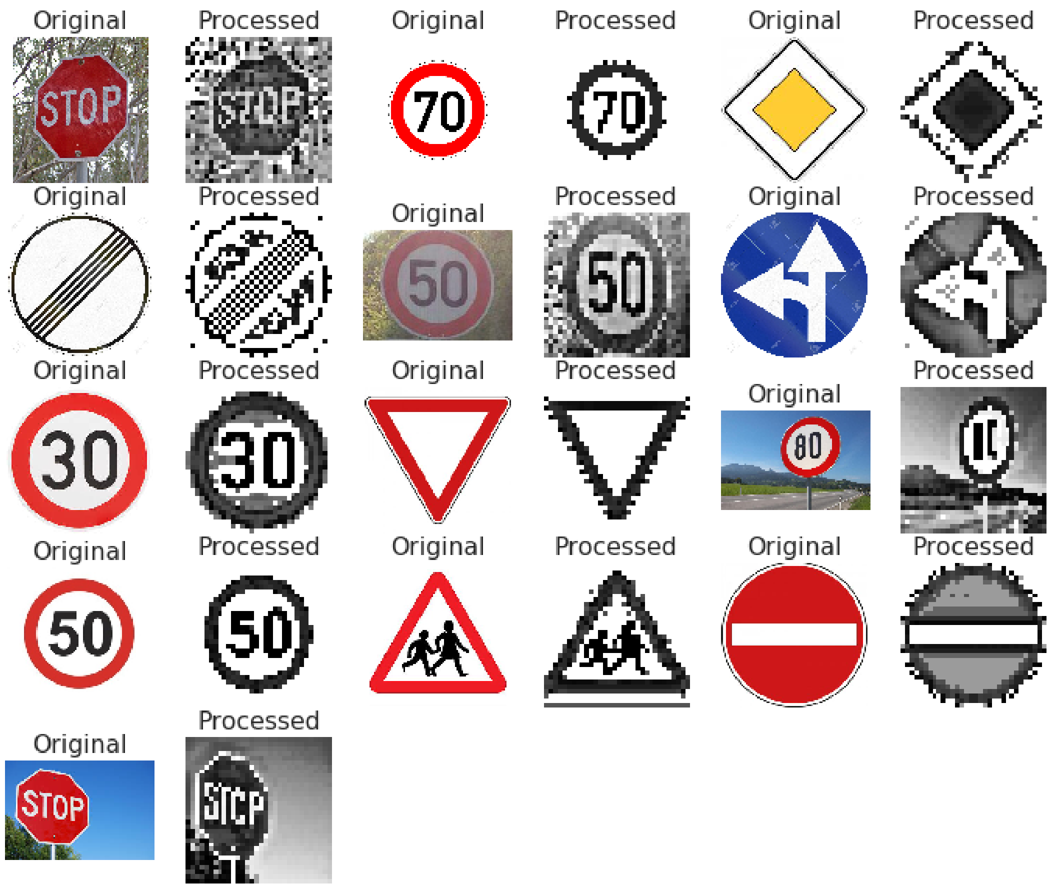
1.6 Step 3: Test a Model on New Images

1.6.1 Load and Output the Images

```
In [116]: from PIL import Image
          import glob
          import os
          from sklearn.metrics import accuracy_score
          import skimage
          from skimage import io
          from skimage import transform
          from skimage.filters import gaussian

          test_images = glob.glob("./test_images/*.*")

          test_labels = {
              './test_images/30.jpg': 1,
              './test_images/50.jpg': 2,
              './test_images/70.jpg': 4,
              './test_images/end_of_speedlimit.jpg': 32,
              './test_images/no_entry.jpg': 17,
              './test_images/priority_road.jpg': 12,
              './test_images/stop.jpg': 14,
              './test_images/yield.jpg': 13,
              './test_images/children_crossing.jpg': 28,
              './test_images/50-Km-limit.jpg': 2,
              './test_images/stop.jpeg': 14,
              './test_images/go_straigth_or_left.jpg': 37,
              './test_images/80-Km-limit.jpg': 5
          }

          def preprocess_test_image(img):
              img = img.resize((32,32))
              return np.asarray(img)

          i = 1
          plt.figure(figsize=(12, 10))
          plt.tight_layout()
          for f in test_images:
              img = Image.open(f)
              img_preprocessed = normalize(preprocess_test_image(img))
              plt.subplot(5, 6, i)
              plt.imshow(img)
              plt.axis('off')
              plt.title('Original')
              i += 1
              plt.subplot(5, 6, i)
              plt.imshow(img_preprocessed, cmap='gray')
```

```
    plt.axis('off')
    plt.title('Processed')
    i += 1

images = []
labels = []
for f in glob.glob('./test_images/*.*'):
    img = io.imread(f)
    img = transform.resize(img, (32,32), order=3, mode='constant',anti_aliasing=True,
    img = gaussian(img, .6, multichannel=True)*255
    img = normalize(img.astype(np.uint8))
    img.shape = (1,) + img.shape + ( 1,)
    images_wild.append(img)
    labels_wild.append(test_labels[el])

images = np.concatenate(images_wild, axis=0)

print ("Test image preprocessing done")
```

```
Test image preprocessing done
```



1.6.2 Predict the Sign Type for Each Image

```
In [117]: with tf.Session() as sess:
              saver.restore(sess, tf.train.latest_checkpoint('./models'))
              predicted_proba = np.vstack(predict(images))
              print('Accuracy Model On Test Images: {}'.format(evaluate(images, labels_wild)))
```

```
INFO:tensorflow:Restoring parameters from ./models/lenet
Accuracy Model On Test Images: 0.2500000298023224
```

1.6.3 Analyze Performance

```
In [118]: def sign_name(label):
              return sign_names.loc[label].SignName

          for true_label,row in zip(labels_wild,predicted_proba):
              top5k = np.argsort(row)[::-1][:5]
              top5p = np.sort(row)[::-1][:5]
              print('Top 5 Labels for image \'{}\':'.format(sign_name(true_label)))
              for k,p in zip(top5k,top5p):
                    print(' - \'{}\' with prob = {:.2f} '.format(sign_name(k),p))
```

```
Top 5 Labels for image 'Stop':
 - 'Stop' with prob = 1.00
 - 'No entry' with prob = 0.00
 - 'Priority road' with prob = 0.00
 - 'Road work' with prob = 0.00
 - 'Turn left ahead' with prob = 0.00
Top 5 Labels for image 'Speed limit (70km/h)':
 - 'Speed limit (70km/h)' with prob = 0.55
 - 'Speed limit (20km/h)' with prob = 0.45
 - 'Speed limit (30km/h)' with prob = 0.00
 - 'Keep right' with prob = 0.00
 - 'Speed limit (120km/h)' with prob = 0.00
Top 5 Labels for image 'Priority road':
 - 'Priority road' with prob = 1.00
 - 'No vehicles' with prob = 0.00
 - 'Road work' with prob = 0.00
 - 'No passing for vehicles over 3.5 metric tons' with prob = 0.00
 - 'No passing' with prob = 0.00
Top 5 Labels for image 'End of all speed and passing limits':
 - 'End of all speed and passing limits' with prob = 1.00
 - 'Priority road' with prob = 0.00
 - 'End of no passing' with prob = 0.00
 - 'End of no passing by vehicles over 3.5 metric tons' with prob = 0.00
 - 'Turn right ahead' with prob = 0.00
Top 5 Labels for image 'Speed limit (50km/h)':
 - 'Speed limit (50km/h)' with prob = 0.92
 - 'Speed limit (30km/h)' with prob = 0.08
 - 'Speed limit (20km/h)' with prob = 0.00
 - 'Speed limit (70km/h)' with prob = 0.00
 - 'Stop' with prob = 0.00
Top 5 Labels for image 'Go straight or left':
 - 'Keep right' with prob = 0.79
 - 'Turn left ahead' with prob = 0.21
 - 'Speed limit (50km/h)' with prob = 0.00
 - 'Speed limit (30km/h)' with prob = 0.00
 - 'Ahead only' with prob = 0.00
Top 5 Labels for image 'Speed limit (30km/h)':
 - 'Speed limit (30km/h)' with prob = 1.00
 - 'Speed limit (80km/h)' with prob = 0.00
 - 'Speed limit (50km/h)' with prob = 0.00
 - 'End of speed limit (80km/h)' with prob = 0.00
 - 'Speed limit (60km/h)' with prob = 0.00
Top 5 Labels for image 'Yield':
 - 'Yield' with prob = 1.00
 - 'Priority road' with prob = 0.00
 - 'End of no passing by vehicles over 3.5 metric tons' with prob = 0.00
 - 'Right-of-way at the next intersection' with prob = 0.00
 - 'Road work' with prob = 0.00
Top 5 Labels for image 'Speed limit (80km/h)':
 - 'Yield' with prob = 1.00
 - 'Keep right' with prob = 0.00
 - 'Stop' with prob = 0.00
 - 'Keep left' with prob = 0.00
 - 'No passing for vehicles over 3.5 metric tons' with prob = 0.00
Top 5 Labels for image 'Speed limit (50km/h)':
 - 'Speed limit (50km/h)' with prob = 1.00
 - 'Speed limit (30km/h)' with prob = 0.00
 - 'Speed limit (80km/h)' with prob = 0.00
 - 'Speed limit (60km/h)' with prob = 0.00
 - 'Keep right' with prob = 0.00
Top 5 Labels for image 'Children crossing':
 - 'Speed limit (30km/h)' with prob = 0.55
 - 'End of speed limit (80km/h)' with prob = 0.38
 - 'Right-of-way at the next intersection' with prob = 0.05
 - 'Speed limit (80km/h)' with prob = 0.01
 - 'Speed limit (60km/h)' with prob = 0.01
Top 5 Labels for image 'No entry':
 - 'No entry' with prob = 1.00
 - 'Keep right' with prob = 0.00
 - 'Stop' with prob = 0.00
 - 'Yield' with prob = 0.00
 - 'End of no passing' with prob = 0.00
Top 5 Labels for image 'Stop':
 - 'Speed limit (30km/h)' with prob = 0.47
 - 'End of speed limit (80km/h)' with prob = 0.38
 - 'Road work' with prob = 0.05
 - 'Stop' with prob = 0.05
 - 'Priority road' with prob = 0.01
Top 5 Labels for image 'Stop':
 - 'Stop' with prob = 1.00
 - 'No entry' with prob = 0.00
 - 'Priority road' with prob = 0.00
 - 'Road work' with prob = 0.00
 - 'Turn left ahead' with prob = 0.00
Top 5 Labels for image 'Stop':
 - 'Speed limit (70km/h)' with prob = 0.55
 - 'Speed limit (20km/h)' with prob = 0.45
 - 'Speed limit (30km/h)' with prob = 0.00
 - 'Keep right' with prob = 0.00
 - 'Speed limit (120km/h)' with prob = 0.00
Top 5 Labels for image 'Stop':
 - 'Priority road' with prob = 1.00
 - 'No vehicles' with prob = 0.00
 - 'Road work' with prob = 0.00
 - 'No passing for vehicles over 3.5 metric tons' with prob = 0.00
 - 'No passing' with prob = 0.00
Top 5 Labels for image 'Stop':
 - 'End of all speed and passing limits' with prob = 1.00
 - 'Priority road' with prob = 0.00
 - 'End of no passing' with prob = 0.00
 - 'End of no passing by vehicles over 3.5 metric tons' with prob = 0.00
 - 'Turn right ahead' with prob = 0.00
Top 5 Labels for image 'Stop':
 - 'Speed limit (50km/h)' with prob = 0.92
 - 'Speed limit (30km/h)' with prob = 0.08
 - 'Speed limit (20km/h)' with prob = 0.00
 - 'Speed limit (70km/h)' with prob = 0.00
 - 'Stop' with prob = 0.00
Top 5 Labels for image 'Stop':
 - 'Keep right' with prob = 0.79
 - 'Turn left ahead' with prob = 0.21
 - 'Speed limit (50km/h)' with prob = 0.00
 - 'Speed limit (30km/h)' with prob = 0.00
 - 'Ahead only' with prob = 0.00
Top 5 Labels for image 'Stop':
 - 'Speed limit (30km/h)' with prob = 1.00
 - 'Speed limit (80km/h)' with prob = 0.00
 - 'Speed limit (50km/h)' with prob = 0.00
 - 'End of speed limit (80km/h)' with prob = 0.00
 - 'Speed limit (60km/h)' with prob = 0.00
Top 5 Labels for image 'Stop':
 - 'Yield' with prob = 1.00
 - 'Priority road' with prob = 0.00
 - 'End of no passing by vehicles over 3.5 metric tons' with prob = 0.00
 - 'Right-of-way at the next intersection' with prob = 0.00
 - 'Road work' with prob = 0.00
Top 5 Labels for image 'Stop':
 - 'Yield' with prob = 1.00
 - 'Keep right' with prob = 0.00
 - 'Stop' with prob = 0.00
 - 'Keep left' with prob = 0.00
 - 'No passing for vehicles over 3.5 metric tons' with prob = 0.00
Top 5 Labels for image 'Stop':
 - 'Speed limit (50km/h)' with prob = 1.00
 - 'Speed limit (30km/h)' with prob = 0.00
 - 'Speed limit (80km/h)' with prob = 0.00
 - 'Speed limit (60km/h)' with prob = 0.00
 - 'Keep right' with prob = 0.00
Top 5 Labels for image 'Stop':
 - 'Speed limit (30km/h)' with prob = 0.55
 - 'End of speed limit (80km/h)' with prob = 0.38
 - 'Right-of-way at the next intersection' with prob = 0.05
 - 'Speed limit (80km/h)' with prob = 0.01
 - 'Speed limit (60km/h)' with prob = 0.01
Top 5 Labels for image 'Stop':
 - 'No entry' with prob = 1.00
 - 'Keep right' with prob = 0.00
 - 'Stop' with prob = 0.00
 - 'Yield' with prob = 0.00
 - 'End of no passing' with prob = 0.00
Top 5 Labels for image 'Stop':
 - 'Speed limit (30km/h)' with prob = 0.47
 - 'End of speed limit (80km/h)' with prob = 0.38
 - 'Road work' with prob = 0.05
 - 'Stop' with prob = 0.05
 - 'Priority road' with prob = 0.01
Top 5 Labels for image 'Stop':
 - 'Stop' with prob = 1.00
 - 'No entry' with prob = 0.00
 - 'Priority road' with prob = 0.00
 - 'Road work' with prob = 0.00
 - 'Turn left ahead' with prob = 0.00
Top 5 Labels for image 'Stop':
 - 'Speed limit (70km/h)' with prob = 0.55
 - 'Speed limit (20km/h)' with prob = 0.45
 - 'Speed limit (30km/h)' with prob = 0.00
 - 'Keep right' with prob = 0.00
 - 'Speed limit (120km/h)' with prob = 0.00
Top 5 Labels for image 'Stop':
 - 'Priority road' with prob = 1.00
 - 'No vehicles' with prob = 0.00
 - 'Road work' with prob = 0.00
 - 'No passing for vehicles over 3.5 metric tons' with prob = 0.00
 - 'No passing' with prob = 0.00
Top 5 Labels for image 'Stop':
 - 'End of all speed and passing limits' with prob = 1.00
 - 'Priority road' with prob = 0.00
 - 'End of no passing' with prob = 0.00
 - 'End of no passing by vehicles over 3.5 metric tons' with prob = 0.00
 - 'Turn right ahead' with prob = 0.00
Top 5 Labels for image 'Stop':
 - 'Speed limit (50km/h)' with prob = 0.92
 - 'Speed limit (30km/h)' with prob = 0.08
 - 'Speed limit (20km/h)' with prob = 0.00
 - 'Speed limit (70km/h)' with prob = 0.00
 - 'Stop' with prob = 0.00
Top 5 Labels for image 'Stop':
 - 'Keep right' with prob = 0.79
 - 'Turn left ahead' with prob = 0.21
 - 'Speed limit (50km/h)' with prob = 0.00
 - 'Speed limit (30km/h)' with prob = 0.00
 - 'Ahead only' with prob = 0.00
Top 5 Labels for image 'Stop':
 - 'Speed limit (30km/h)' with prob = 1.00
 - 'Speed limit (80km/h)' with prob = 0.00
 - 'Speed limit (50km/h)' with prob = 0.00
 - 'End of speed limit (80km/h)' with prob = 0.00
 - 'Speed limit (60km/h)' with prob = 0.00
Top 5 Labels for image 'Stop':
 - 'Yield' with prob = 1.00
 - 'Priority road' with prob = 0.00
 - 'End of no passing by vehicles over 3.5 metric tons' with prob = 0.00
 - 'Right-of-way at the next intersection' with prob = 0.00
 - 'Road work' with prob = 0.00
Top 5 Labels for image 'Stop':
 - 'Yield' with prob = 1.00
 - 'Keep right' with prob = 0.00
 - 'Stop' with prob = 0.00
 - 'Keep left' with prob = 0.00
 - 'No passing for vehicles over 3.5 metric tons' with prob = 0.00
Top 5 Labels for image 'Stop':
 - 'Speed limit (50km/h)' with prob = 1.00
 - 'Speed limit (30km/h)' with prob = 0.00
 - 'Speed limit (80km/h)' with prob = 0.00
 - 'Speed limit (60km/h)' with prob = 0.00
 - 'Keep right' with prob = 0.00
Top 5 Labels for image 'Stop':
 - 'Speed limit (30km/h)' with prob = 0.55
 - 'End of speed limit (80km/h)' with prob = 0.38
 - 'Right-of-way at the next intersection' with prob = 0.05
 - 'Speed limit (80km/h)' with prob = 0.01
 - 'Speed limit (60km/h)' with prob = 0.01
Top 5 Labels for image 'Stop':
 - 'No entry' with prob = 1.00
 - 'Keep right' with prob = 0.00
 - 'Stop' with prob = 0.00
 - 'Yield' with prob = 0.00
 - 'End of no passing' with prob = 0.00
Top 5 Labels for image 'Stop':
 - 'Speed limit (30km/h)' with prob = 0.47
 - 'End of speed limit (80km/h)' with prob = 0.38
 - 'Road work' with prob = 0.05
 - 'Stop' with prob = 0.05
 - 'Priority road' with prob = 0.01
Top 5 Labels for image 'Stop':
 - 'Stop' with prob = 1.00
 - 'No entry' with prob = 0.00
 - 'Yield' with prob = 0.00
 - 'Priority road' with prob = 0.00
 - 'Speed limit (80km/h)' with prob = 0.00
Top 5 Labels for image 'Stop':
 - 'Speed limit (20km/h)' with prob = 0.67
 - 'Speed limit (70km/h)' with prob = 0.33
 - 'Speed limit (30km/h)' with prob = 0.00
 - 'Speed limit (120km/h)' with prob = 0.00
 - 'Speed limit (50km/h)' with prob = 0.00
Top 5 Labels for image 'Stop':
 - 'Priority road' with prob = 1.00
 - 'No vehicles' with prob = 0.00
 - 'No passing for vehicles over 3.5 metric tons' with prob = 0.00
 - 'No passing' with prob = 0.00
 - 'Road work' with prob = 0.00
Top 5 Labels for image 'Stop':
 - 'End of all speed and passing limits' with prob = 1.00
 - 'Priority road' with prob = 0.00
 - 'End of no passing' with prob = 0.00
 - 'End of no passing by vehicles over 3.5 metric tons' with prob = 0.00
 - 'End of speed limit (80km/h)' with prob = 0.00
Top 5 Labels for image 'Stop':
 - 'Speed limit (50km/h)' with prob = 0.98
 - 'Speed limit (30km/h)' with prob = 0.02
 - 'Speed limit (20km/h)' with prob = 0.00
 - 'Speed limit (70km/h)' with prob = 0.00
 - 'Keep right' with prob = 0.00
Top 5 Labels for image 'Stop':
 - 'Turn left ahead' with prob = 1.00
 - 'Ahead only' with prob = 0.00
 - 'Keep right' with prob = 0.00
 - 'Yield' with prob = 0.00
 - 'Speed limit (50km/h)' with prob = 0.00
Top 5 Labels for image 'Stop':
 - 'Speed limit (30km/h)' with prob = 1.00
 - 'End of speed limit (80km/h)' with prob = 0.00
 - 'Speed limit (80km/h)' with prob = 0.00
 - 'Speed limit (50km/h)' with prob = 0.00
 - 'Speed limit (60km/h)' with prob = 0.00
Top 5 Labels for image 'Stop':
 - 'Yield' with prob = 1.00
 - 'Priority road' with prob = 0.00
 - 'End of no passing by vehicles over 3.5 metric tons' with prob = 0.00
 - 'Speed limit (30km/h)' with prob = 0.00
 - 'End of no passing' with prob = 0.00
Top 5 Labels for image 'Stop':
 - 'Stop' with prob = 0.94
 - 'Yield' with prob = 0.05
 - 'No entry' with prob = 0.00
 - 'Keep right' with prob = 0.00
 - 'No passing for vehicles over 3.5 metric tons' with prob = 0.00
Top 5 Labels for image 'Stop':
 - 'Speed limit (50km/h)' with prob = 1.00
 - 'Speed limit (80km/h)' with prob = 0.00
 - 'Speed limit (30km/h)' with prob = 0.00
 - 'Speed limit (60km/h)' with prob = 0.00
 - 'Speed limit (100km/h)' with prob = 0.00
Top 5 Labels for image 'Stop':
 - 'Roundabout mandatory' with prob = 1.00
 - 'Speed limit (30km/h)' with prob = 0.00
 - 'Speed limit (80km/h)' with prob = 0.00
 - 'No passing for vehicles over 3.5 metric tons' with prob = 0.00
 - 'End of no passing by vehicles over 3.5 metric tons' with prob = 0.00
Top 5 Labels for image 'Stop':
 - 'No entry' with prob = 1.00
 - 'Stop' with prob = 0.00
 - 'Yield' with prob = 0.00
 - 'Keep right' with prob = 0.00
 - 'End of no passing' with prob = 0.00
Top 5 Labels for image 'Stop':
 - 'End of speed limit (80km/h)' with prob = 0.87
 - 'Speed limit (80km/h)' with prob = 0.06
 - 'Children crossing' with prob = 0.03
 - 'Dangerous curve to the right' with prob = 0.01
 - 'End of no passing by vehicles over 3.5 metric tons' with prob = 0.01
```

```
In [ ]:
```