Traffic_Sign_Classifier_Final

October 20, 2020

1 Deep Learning Project: Traffic Sign Recognition Classifier using LeNet CNN

This Project is part of Udacity Self-Driving Car Engineer Nanodegree Design and implement a deep learning model that learns to recognize traffic signs. Train and test your model on the German Traffic Sign Dataset. Author: Carolina Hoffmann-Becking 19 Oct 2020

1.1 1. Load the Data

In [1]:
```
# Load pickled data
import pickle

training_file = 'train.p'
validation_file= 'valid.p'
testing_file = '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']
```

1.2 2. Exploration of the dataset using Numpy, Pandas and Matpotlib

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**

1.2.1 2.1 Number of Samples for training, validation and testing

```
In [2]: print(train)
```

```
{'coords': array([[  6,   5,  21,  20],
       [  6,   6,  22,  22],
       [  5,   6,  22,  23],
       ...,
       [ 17,  15, 178, 155],
       [ 17,  15, 183, 160],
       [ 20,  18, 211, 184]], dtype=uint8), 'labels': array([41, 41, 41, ..., 25, 25, 25], dtype
         [ 27,  24,  23],
         [ 27,  24,  22],
         ...,
         [ 32,  28,  24],
         [ 31,  27,  25],
         [ 31,  27,  26]],

        [[ 29,  26,  25],
         [ 27,  25,  23],
         [ 27,  25,  23],
         ...,
         [ 32,  28,  24],
         [ 31,  27,  24],
         [ 30,  27,  25]],

        [[ 28,  26,  26],
         [ 27,  25,  23],
         [ 26,  25,  23],
         ...,
         [ 32,  28,  24],
         [ 31,  27,  24],
         [ 30,  27,  25]],


        ...,
        [[ 27,  24,  23],
         [ 28,  25,  24],
         [ 30,  25,  24],
         ...,
         [ 27,  24,  23],
         [ 28,  24,  22],
         [ 29,  25,  22]],
```

```
 [[ 28,  23,  23],
  [ 29,  24,  24],
  [ 31,  25,  24],
  ...,
  [ 27,  24,  23],
  [ 28,  24,  22],
  [ 28,  24,  21]],

 [[ 29,  23,  23],
  [ 30,  24,  24],
  [ 32,  24,  23],
  ...,
  [ 27,  24,  22],
  [ 27,  23,  21],
  [ 26,  22,  20]]],


[[[ 28,  24,  24],
  [ 26,  23,  23],
  [ 27,  24,  24],
  ...,
  [ 31,  28,  26],
  [ 31,  28,  27],
  [ 32,  28,  27]],

 [[ 27,  24,  24],
  [ 27,  24,  24],
  [ 28,  25,  24],
  ...,
  [ 31,  27,  25],
  [ 31,  27,  26],
  [ 33,  29,  27]],

 [[ 26,  24,  24],
  [ 26,  24,  24],
  [ 27,  24,  23],
  ...,
  [ 31,  26,  25],
  [ 31,  27,  26],
  [ 33,  29,  27]],

 ...,
 [[ 28,  25,  23],
  [ 30,  27,  24],
  [ 30,  27,  24],
  ...,
  [ 27,  24,  22],
  [ 27,  24,  22],
```

```
 [ 28,  24,  22]],

[[ 27,  24,  22],
 [ 29,  26,  23],
 [ 31,  26,  24],
 ...,
 [ 26,  23,  21],
 [ 27,  24,  22],
 [ 28,  25,  23]],

[[ 28,  24,  23],
 [ 28,  24,  22],
 [ 29,  24,  22],
 ...,
 [ 27,  23,  22],
 [ 27,  24,  23],
 [ 29,  26,  25]]],


[[[ 29,  25,  25],
  [ 29,  26,  26],
  [ 30,  27,  27],
  ...,
  [ 31,  27,  24],
  [ 31,  28,  25],
  [ 32,  29,  27]],

 [[ 27,  24,  24],
  [ 27,  25,  25],
  [ 28,  26,  26],
  ...,
  [ 31,  27,  23],
  [ 32,  28,  25],
  [ 33,  30,  27]],

 [[ 27,  24,  24],
  [ 28,  26,  26],
  [ 29,  27,  27],
  ...,
  [ 32,  28,  24],
  [ 32,  28,  24],
  [ 33,  29,  26]],

 ...,
 [[ 28,  26,  22],
  [ 29,  26,  21],
  [ 31,  26,  22],
  ...,
```

```
  [ 29,  24,  21],
  [ 28,  23,  20],
  [ 28,  23,  22]],

 [[ 27,  26,  23],
  [ 28,  25,  21],
  [ 30,  25,  22],
  ...,
  [ 28,  23,  21],
  [ 27,  22,  20],
  [ 28,  24,  22]],

 [[ 29,  26,  23],
  [ 28,  24,  21],
  [ 29,  24,  21],
  ...,
  [ 29,  25,  23],
  [ 28,  24,  22],
  [ 30,  26,  24]]],


...,
[[[ 51,  67,  86],
  [ 55,  59,  71],
  [ 75,  81,  92],
  ...,
  [250, 248, 243],
  [207, 212, 233],
  [121, 116, 140]],

 [[ 35,  42,  49],
  [ 48,  47,  51],
  [ 91,  96, 113],
  ...,
  [220, 224, 226],
  [169, 177, 187],
  [ 84,  87, 100]],

 [[ 27,  26,  29],
  [ 41,  38,  39],
  [ 55,  64,  78],
  ...,
  [122, 143, 160],
  [ 97, 104, 129],
  [ 59,  59,  56]],

 ...,
 [[ 24,  23,  27],
```

```
 [ 21,  20,  27],
 [ 20,  19,  22],
 ...,
 [ 76,  79,  83],
 [ 54,  64,  77],
 [ 45,  51,  65]],

[[ 31,  31,  33],
 [ 22,  23,  29],
 [ 20,  18,  21],
 ...,
 [ 66,  67,  84],
 [ 56,  65,  74],
 [ 45,  55,  73]],

[[ 28,  28,  30],
 [ 22,  21,  25],
 [ 19,  18,  19],
 ...,
 [ 67,  63,  76],
 [ 39,  45,  55],
 [ 32,  37,  47]]],


[[[ 82,  78,  96],
  [120, 126, 148],
  [112, 125, 146],
  ...,
  [185, 182, 177],
  [204, 210, 215],
  [132, 114, 121]],

 [[ 74,  77,  93],
  [171, 174, 185],
  [137, 164, 184],
  ...,
  [180, 181, 187],
  [198, 200, 213],
  [ 79,  85,  85]],

 [[ 54,  50,  56],
  [100, 106, 118],
  [117, 132, 158],
  ...,
  [157, 153, 160],
  [176, 183, 199],
  [ 88,  80,  82]],
```

```
 ...,
 [[ 22,  21,  22],
  [ 20,  19,  19],
  [ 18,  16,  19],
  ...,
  [ 50,  50,  60],
  [ 37,  43,  52],
  [ 30,  41,  59]],

 [[ 18,  16,  18],
  [ 19,  17,  18],
  [ 19,  18,  21],
  ...,
  [ 36,  44,  60],
  [ 33,  36,  48],
  [ 36,  43,  61]],

 [[ 18,  17,  20],
  [ 21,  20,  23],
  [ 24,  22,  25],
  ...,
  [ 32,  34,  41],
  [ 45,  42,  48],
  [ 41,  43,  52]]],


[[[ 69,  79,  96],
  [ 24,  26,  28],
  [ 40,  42,  45],
  ...,
  [225, 234, 237],
  [151, 161, 166],
  [164, 162, 169]],

 [[ 99, 110, 125],
  [ 36,  41,  47],
  [ 63,  56,  62],
  ...,
  [197, 216, 224],
  [154, 163, 169],
  [164, 163, 159]],

 [[104, 107, 113],
  [ 34,  37,  40],
  [ 72,  70,  77],
  ...,
  [223, 237, 235],
  [181, 192, 198],
```

```
          [166, 167, 159]],

         ...,
         [[ 21,  20,  23],
          [ 23,  24,  30],
          [ 19,  20,  24],
          ...,
          [ 45,  47,  54],
          [ 58,  62,  70],
          [ 58,  70,  82]],

         [[ 18,  17,  21],
          [ 19,  19,  24],
          [ 18,  18,  23],
          ...,
          [ 36,  36,  40],
          [ 58,  59,  70],
          [ 61,  69,  81]],

         [[ 17,  16,  19],
          [ 16,  15,  18],
          [ 16,  15,  18],
          ...,
          [ 40,  40,  44],
          [ 57,  62,  73],
          [ 57,  68,  80]]]], dtype=uint8), 'sizes': array([[ 26,  25],
        [ 27,  27],
        [ 27,  28],
        ...,
        [194, 169],
        [201, 175],
        [230, 201]], dtype=uint8)}
```

In [3]:
```python
train.keys()
```

Out[3]:
```
dict_keys(['coords', 'labels', 'features', 'sizes'])
```

In [4]:
```python
n_train = len(train['labels'])
n_validation = len(valid['labels'])
n_test = len(test['labels'])

print('Number of Training Labels', n_train)
print('Number of Validation Labels', n_validation)
print('Number of Testing Labels', n_test)
```

```
Number of Training Labels 34799
Number of Validation Labels 4410
Number of Testing Labels 12630
```

1.2.2 2.2 Number of unique labels (classes)

```
In [5]: import pandas as pd
        import numpy as np

        df = pd.DataFrame.from_dict(train,orient='index').transpose()
        df.head()
```

```
Out[5]:                                               coords  \
        0  [[6, 5, 21, 20], [6, 6, 22, 22], [5, 6, 22, 23...

                                                      labels  \
        0  [41, 41, 41, 41, 41, 41, 41, 41, 41, 41, 41, 4...

                                                    features  \
        0  [[[[28 25 24], [27 24 23], [27 24 22], [27 24 ...

                                                       sizes
        0  [[26, 25], [27, 27], [27, 28], [27, 28], [29, ...
```

```
In [6]: Labels = list(df['labels'])[0]
        uniqueLabels = np.unique(Labels)
        n_classes = len(uniqueLabels)
        print(n_classes)
        print(uniqueLabels)
```

```
43
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42]
```

1.2.3 2.3 Shape of Images

```
In [7]: Features = list(df['features'])[0]
        print(Features)
```

```
[[[[ 28  25  24]
   [ 27  24  23]
   [ 27  24  22]
   ...,
   [ 32  28  24]
   [ 31  27  25]
   [ 31  27  26]]

  [[ 29  26  25]
   [ 27  25  23]
   [ 27  25  23]
   ...,
   [ 32  28  24]
```

```
  [ 31  27  24]
  [ 30  27  25]]

 [[ 28  26  26]
  [ 27  25  23]
  [ 26  25  23]
  ...,
  [ 32  28  24]
  [ 31  27  24]
  [ 30  27  25]]

 ...,
 [[ 27  24  23]
  [ 28  25  24]
  [ 30  25  24]
  ...,
  [ 27  24  23]
  [ 28  24  22]
  [ 29  25  22]]

 [[ 28  23  23]
  [ 29  24  24]
  [ 31  25  24]
  ...,
  [ 27  24  23]
  [ 28  24  22]
  [ 28  24  21]]

 [[ 29  23  23]
  [ 30  24  24]
  [ 32  24  23]
  ...,
  [ 27  24  22]
  [ 27  23  21]
  [ 26  22  20]]]


[[[ 28  24  24]
  [ 26  23  23]
  [ 27  24  24]
  ...,
  [ 31  28  26]
  [ 31  28  27]
  [ 32  28  27]]

 [[ 27  24  24]
  [ 27  24  24]
  [ 28  25  24]
```

```
  ...,
  [ 31  27  25]
  [ 31  27  26]
  [ 33  29  27]]

 [[ 26  24  24]
  [ 26  24  24]
  [ 27  24  23]
  ...,
  [ 31  26  25]
  [ 31  27  26]
  [ 33  29  27]]

 ...,
 [[ 28  25  23]
  [ 30  27  24]
  [ 30  27  24]
  ...,
  [ 27  24  22]
  [ 27  24  22]
  [ 28  24  22]]

 [[ 27  24  22]
  [ 29  26  23]
  [ 31  26  24]
  ...,
  [ 26  23  21]
  [ 27  24  22]
  [ 28  25  23]]

 [[ 28  24  23]
  [ 28  24  22]
  [ 29  24  22]
  ...,
  [ 27  23  22]
  [ 27  24  23]
  [ 29  26  25]]]


[[[ 29  25  25]
  [ 29  26  26]
  [ 30  27  27]
  ...,
  [ 31  27  24]
  [ 31  28  25]
  [ 32  29  27]]

 [[ 27  24  24]
```

```
  [ 27  25  25]
  [ 28  26  26]
  ...,
  [ 31  27  23]
  [ 32  28  25]
  [ 33  30  27]]

 [[ 27  24  24]
  [ 28  26  26]
  [ 29  27  27]
  ...,
  [ 32  28  24]
  [ 32  28  24]
  [ 33  29  26]]

 ...,
 [[ 28  26  22]
  [ 29  26  21]
  [ 31  26  22]
  ...,
  [ 29  24  21]
  [ 28  23  20]
  [ 28  23  22]]

 [[ 27  26  23]
  [ 28  25  21]
  [ 30  25  22]
  ...,
  [ 28  23  21]
  [ 27  22  20]
  [ 28  24  22]]

 [[ 29  26  23]
  [ 28  24  21]
  [ 29  24  21]
  ...,
  [ 29  25  23]
  [ 28  24  22]
  [ 30  26  24]]]


...,
[[[ 51  67  86]
  [ 55  59  71]
  [ 75  81  92]
  ...,
  [250 248 243]
  [207 212 233]
```

```
  [121 116 140]]

 [[ 35  42  49]
  [ 48  47  51]
  [ 91  96 113]
  ...,
  [220 224 226]
  [169 177 187]
  [ 84  87 100]]

 [[ 27  26  29]
  [ 41  38  39]
  [ 55  64  78]
  ...,
  [122 143 160]
  [ 97 104 129]
  [ 59  59  56]]

 ...,
 [[ 24  23  27]
  [ 21  20  27]
  [ 20  19  22]
  ...,
  [ 76  79  83]
  [ 54  64  77]
  [ 45  51  65]]

 [[ 31  31  33]
  [ 22  23  29]
  [ 20  18  21]
  ...,
  [ 66  67  84]
  [ 56  65  74]
  [ 45  55  73]]

 [[ 28  28  30]
  [ 22  21  25]
  [ 19  18  19]
  ...,
  [ 67  63  76]
  [ 39  45  55]
  [ 32  37  47]]]


[[[ 82  78  96]
  [120 126 148]
  [112 125 146]
  ...,
```

```
  [185 182 177]
  [204 210 215]
  [132 114 121]]

 [[ 74  77  93]
  [171 174 185]
  [137 164 184]
  ...,
  [180 181 187]
  [198 200 213]
  [ 79  85  85]]

 [[ 54  50  56]
  [100 106 118]
  [117 132 158]
  ...,
  [157 153 160]
  [176 183 199]
  [ 88  80  82]]

 ...,
 [[ 22  21  22]
  [ 20  19  19]
  [ 18  16  19]
  ...,
  [ 50  50  60]
  [ 37  43  52]
  [ 30  41  59]]

 [[ 18  16  18]
  [ 19  17  18]
  [ 19  18  21]
  ...,
  [ 36  44  60]
  [ 33  36  48]
  [ 36  43  61]]

 [[ 18  17  20]
  [ 21  20  23]
  [ 24  22  25]
  ...,
  [ 32  34  41]
  [ 45  42  48]
  [ 41  43  52]]]


[[[ 69  79  96]
  [ 24  26  28]
```

```
  [ 40  42  45]
  ...,
  [225 234 237]
  [151 161 166]
  [164 162 169]]

 [[ 99 110 125]
  [ 36  41  47]
  [ 63  56  62]
  ...,
  [197 216 224]
  [154 163 169]
  [164 163 159]]

 [[104 107 113]
  [ 34  37  40]
  [ 72  70  77]
  ...,
  [223 237 235]
  [181 192 198]
  [166 167 159]]


 ...,
 [[ 21  20  23]
  [ 23  24  30]
  [ 19  20  24]
  ...,
  [ 45  47  54]
  [ 58  62  70]
  [ 58  70  82]]

 [[ 18  17  21]
  [ 19  19  24]
  [ 18  18  23]
  ...,
  [ 36  36  40]
  [ 58  59  70]
  [ 61  69  81]]

 [[ 17  16  19]
  [ 16  15  18]
  [ 16  15  18]
  ...,
  [ 40  40  44]
  [ 57  62  73]
  [ 57  68  80]]]]
```

```
In [8]: image_shape = Features[0].shape
        print(image_shape)
```

```
(32, 32, 3)
```

1.2.4 2.4 Visualisation of sample traffic sign

```
In [9]: import matplotlib.pyplot as plt
        %matplotlib inline
```

```
In [10]: print(X_train)
```

```
[[[[ 28  25  24]
   [ 27  24  23]
   [ 27  24  22]
   ...,
   [ 32  28  24]
   [ 31  27  25]
   [ 31  27  26]]

  [[ 29  26  25]
   [ 27  25  23]
   [ 27  25  23]
   ...,
   [ 32  28  24]
   [ 31  27  24]
   [ 30  27  25]]

  [[ 28  26  26]
   [ 27  25  23]
   [ 26  25  23]
   ...,
   [ 32  28  24]
   [ 31  27  24]
   [ 30  27  25]]


  ...,
  [[ 27  24  23]
   [ 28  25  24]
   [ 30  25  24]
   ...,
   [ 27  24  23]
   [ 28  24  22]
   [ 29  25  22]]

  [[ 28  23  23]
   [ 29  24  24]
   [ 31  25  24]
```

```
  ...,
  [ 27  24  23]
  [ 28  24  22]
  [ 28  24  21]]

 [[ 29  23  23]
  [ 30  24  24]
  [ 32  24  23]
  ...,
  [ 27  24  22]
  [ 27  23  21]
  [ 26  22  20]]]


[[[ 28  24  24]
  [ 26  23  23]
  [ 27  24  24]
  ...,
  [ 31  28  26]
  [ 31  28  27]
  [ 32  28  27]]

 [[ 27  24  24]
  [ 27  24  24]
  [ 28  25  24]
  ...,
  [ 31  27  25]
  [ 31  27  26]
  [ 33  29  27]]

 [[ 26  24  24]
  [ 26  24  24]
  [ 27  24  23]
  ...,
  [ 31  26  25]
  [ 31  27  26]
  [ 33  29  27]]

 ...,
 [[ 28  25  23]
  [ 30  27  24]
  [ 30  27  24]
  ...,
  [ 27  24  22]
  [ 27  24  22]
  [ 28  24  22]]

 [[ 27  24  22]
```

```
  [ 29  26  23]
  [ 31  26  24]
  ...,
  [ 26  23  21]
  [ 27  24  22]
  [ 28  25  23]]

 [[ 28  24  23]
  [ 28  24  22]
  [ 29  24  22]
  ...,
  [ 27  23  22]
  [ 27  24  23]
  [ 29  26  25]]]


[[[ 29  25  25]
  [ 29  26  26]
  [ 30  27  27]
  ...,
  [ 31  27  24]
  [ 31  28  25]
  [ 32  29  27]]

 [[ 27  24  24]
  [ 27  25  25]
  [ 28  26  26]
  ...,
  [ 31  27  23]
  [ 32  28  25]
  [ 33  30  27]]

 [[ 27  24  24]
  [ 28  26  26]
  [ 29  27  27]
  ...,
  [ 32  28  24]
  [ 32  28  24]
  [ 33  29  26]]

 ...,
 [[ 28  26  22]
  [ 29  26  21]
  [ 31  26  22]
  ...,
  [ 29  24  21]
  [ 28  23  20]
  [ 28  23  22]]
```

```
 [[ 27  26  23]
  [ 28  25  21]
  [ 30  25  22]
  ...,
  [ 28  23  21]
  [ 27  22  20]
  [ 28  24  22]]

 [[ 29  26  23]
  [ 28  24  21]
  [ 29  24  21]
  ...,
  [ 29  25  23]
  [ 28  24  22]
  [ 30  26  24]]]


...,
[[[ 51  67  86]
  [ 55  59  71]
  [ 75  81  92]
  ...,
  [250 248 243]
  [207 212 233]
  [121 116 140]]

 [[ 35  42  49]
  [ 48  47  51]
  [ 91  96 113]
  ...,
  [220 224 226]
  [169 177 187]
  [ 84  87 100]]

 [[ 27  26  29]
  [ 41  38  39]
  [ 55  64  78]
  ...,
  [122 143 160]
  [ 97 104 129]
  [ 59  59  56]]

 ...,
 [[ 24  23  27]
  [ 21  20  27]
  [ 20  19  22]
  ...,
```

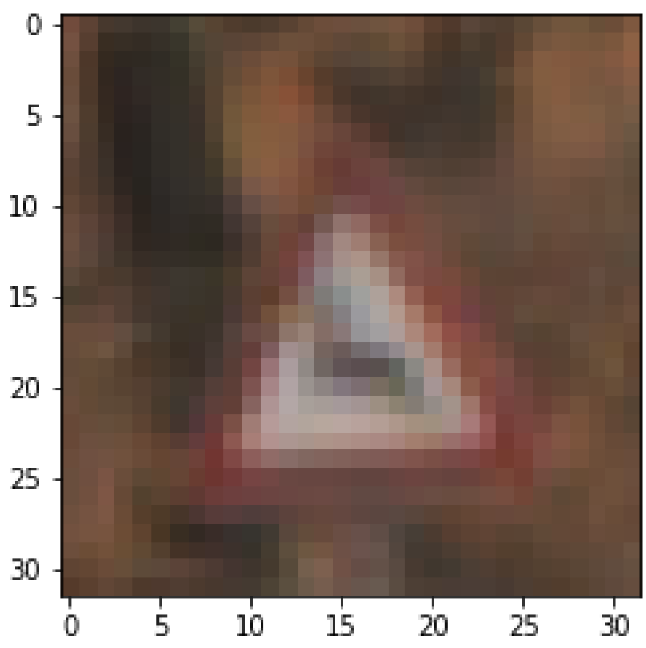
```
  [ 76  79  83]
  [ 54  64  77]
  [ 45  51  65]]

 [[ 31  31  33]
  [ 22  23  29]
  [ 20  18  21]
  ...,
  [ 66  67  84]
  [ 56  65  74]
  [ 45  55  73]]

 [[ 28  28  30]
  [ 22  21  25]
  [ 19  18  19]
  ...,
  [ 67  63  76]
  [ 39  45  55]
  [ 32  37  47]]]


[[[ 82  78  96]
  [120 126 148]
  [112 125 146]
  ...,
  [185 182 177]
  [204 210 215]
  [132 114 121]]

 [[ 74  77  93]
  [171 174 185]
  [137 164 184]
  ...,
  [180 181 187]
  [198 200 213]
  [ 79  85  85]]

 [[ 54  50  56]
  [100 106 118]
  [117 132 158]
  ...,
  [157 153 160]
  [176 183 199]
  [ 88  80  82]]

 ...,
 [[ 22  21  22]
  [ 20  19  19]
```

```
  [ 18  16  19]
  ...,
  [ 50  50  60]
  [ 37  43  52]
  [ 30  41  59]]

 [[ 18  16  18]
  [ 19  17  18]
  [ 19  18  21]
  ...,
  [ 36  44  60]
  [ 33  36  48]
  [ 36  43  61]]

 [[ 18  17  20]
  [ 21  20  23]
  [ 24  22  25]
  ...,
  [ 32  34  41]
  [ 45  42  48]
  [ 41  43  52]]]


[[[ 69  79  96]
  [ 24  26  28]
  [ 40  42  45]
  ...,
  [225 234 237]
  [151 161 166]
  [164 162 169]]

 [[ 99 110 125]
  [ 36  41  47]
  [ 63  56  62]
  ...,
  [197 216 224]
  [154 163 169]
  [164 163 159]]

 [[104 107 113]
  [ 34  37  40]
  [ 72  70  77]
  ...,
  [223 237 235]
  [181 192 198]
  [166 167 159]]

 ...,
```

```
[[ 21  20  23]
 [ 23  24  30]
 [ 19  20  24]
 ...,
 [ 45  47  54]
 [ 58  62  70]
 [ 58  70  82]]

[[ 18  17  21]
 [ 19  19  24]
 [ 18  18  23]
 ...,
 [ 36  36  40]
 [ 58  59  70]
 [ 61  69  81]]

[[ 17  16  19]
 [ 16  15  18]
 [ 16  15  18]
 ...,
 [ 40  40  44]
 [ 57  62  73]
 [ 57  68  80]]]]
```

In [11]:
```
image_sample = X_train[300]
plt.imshow(image_sample)
```

Out[11]:
```
<matplotlib.image.AxesImage at 0x7f05c0beebe0>
```



1.2.5 2.5 Analysis of labels distribution

In [12]:
```
#31 - wild animal crossing
print(y_train[300])
```

```
31
```

In [13]:
```
values, train_counts = np.unique(y_train, return_counts=True)
values, valid_counts = np.unique(y_valid, return_counts=True)
values, test_counts = np.unique(y_test, return_counts=True)
```

In [14]:
```
df_plot = pd.DataFrame(values,index=None, columns=['values'])
df_plot = df_plot.set_index('values')
df_plot['train_counts'] = train_counts
df_plot['valid_counts'] = valid_counts
df_plot['test_counts'] = test_counts
df_plot = df_plot.sort_values(by='train_counts')
df_plot
```

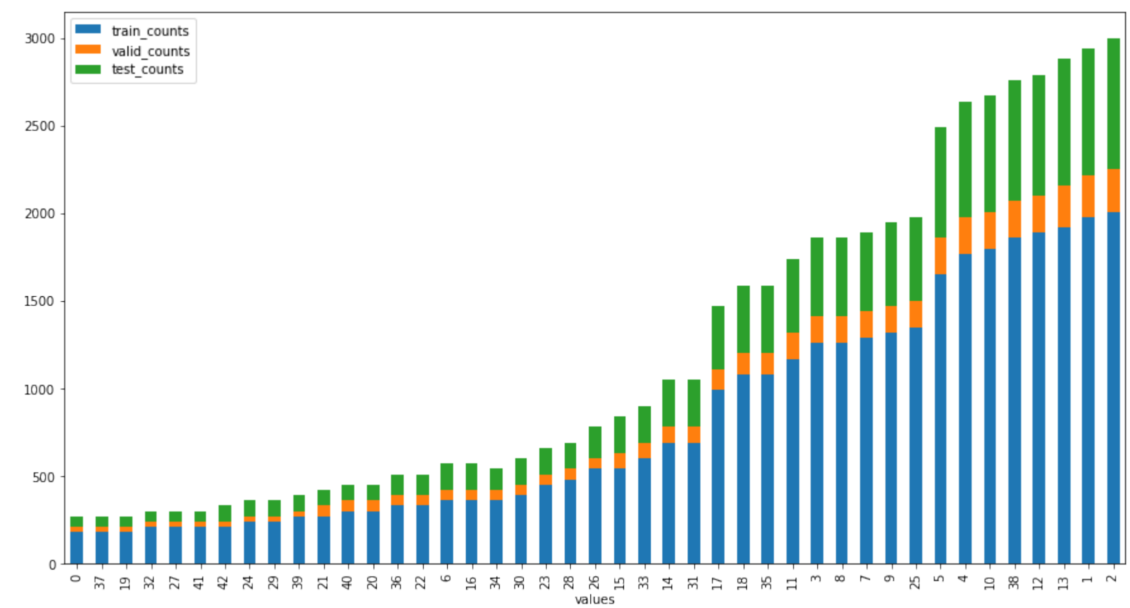
Out[14]:

values	train_counts	valid_counts	test_counts
0	180	30	60
37	180	30	60
19	180	30	60
32	210	30	60
27	210	30	60
41	210	30	60
42	210	30	90
24	240	30	90
29	240	30	90
39	270	30	90
21	270	60	90
40	300	60	90
20	300	60	90
36	330	60	120
22	330	60	120
6	360	60	150
16	360	60	150
34	360	60	120
30	390	60	150
23	450	60	150
28	480	60	150
26	540	60	180
15	540	90	210
33	599	90	210

14	690	90	270
31	690	90	270
17	990	120	360
18	1080	120	390
35	1080	120	390
11	1170	150	420
3	1260	150	450
8	1260	150	450
7	1290	150	450
9	1320	150	480
25	1350	150	480
5	1650	210	630
4	1770	210	660
10	1800	210	660
38	1860	210	690
12	1890	210	690
13	1920	240	720
1	1980	240	720
2	2010	240	750

```
In [15]: df_plot.plot.bar(stacked=True, figsize=(15, 8))
```

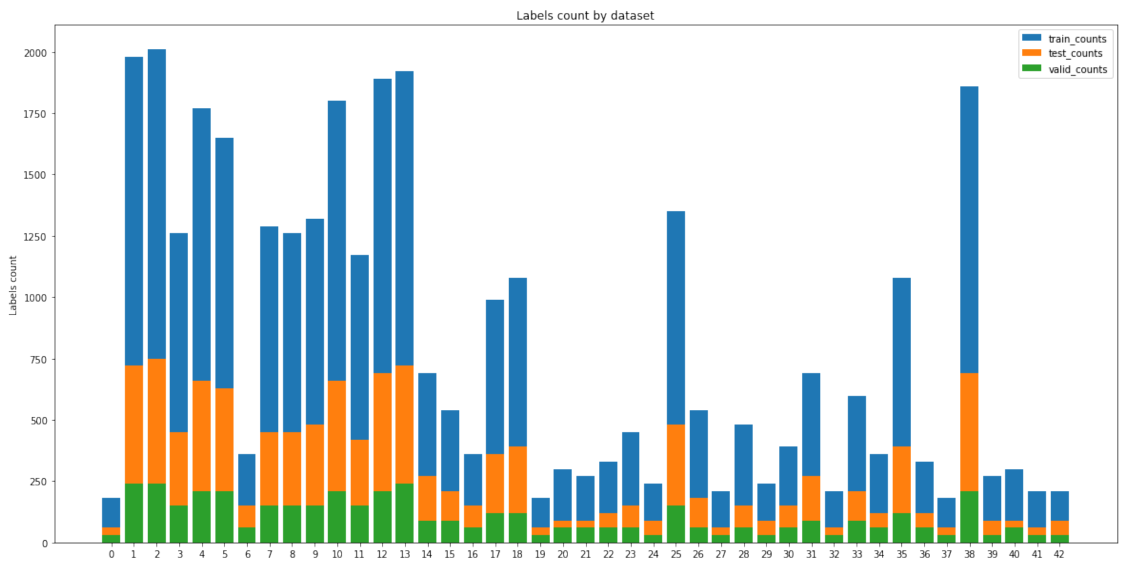
```
Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x7f05c2c2b908>
```




```
In [16]: x = np.arange(len(values))
         width = 0.8
```

```
fig, ax = plt.subplots(figsize=(20, 10))
rects1 = ax.bar(x, train_counts, width, label='train_counts')
rects2 = ax.bar(x, test_counts, width, label='test_counts')
rects3 = ax.bar(x, valid_counts, width, label='valid_counts')

# Add some text for labels, title and custom x-axis tick labels, etc.
ax.set_ylabel('Labels count')
ax.set_title('Labels count by dataset')
ax.set_xticks(x)
ax.set_xticklabels(values)
ax.legend()
```

Out[16]: <matplotlib.legend.Legend at 0x7f05c0742978>



1.3 3. Pre-process the Data Set

In [17]:
```
X_train.shape
```

Out[17]: (34799, 32, 32, 3)

In [18]:
```
print(X_train[0].shape)
X_train[0]
```

```
(32, 32, 3)
```

Out[18]:
```
array([[[28, 25, 24],
        [27, 24, 23],
        [27, 24, 22],
```

```
        ...,
        [32, 28, 24],
        [31, 27, 25],
        [31, 27, 26]],

       [[29, 26, 25],
        [27, 25, 23],
        [27, 25, 23],
        ...,
        [32, 28, 24],
        [31, 27, 24],
        [30, 27, 25]],

       [[28, 26, 26],
        [27, 25, 23],
        [26, 25, 23],
        ...,
        [32, 28, 24],
        [31, 27, 24],
        [30, 27, 25]],

       ...,
       [[27, 24, 23],
        [28, 25, 24],
        [30, 25, 24],
        ...,
        [27, 24, 23],
        [28, 24, 22],
        [29, 25, 22]],

       [[28, 23, 23],
        [29, 24, 24],
        [31, 25, 24],
        ...,
        [27, 24, 23],
        [28, 24, 22],
        [28, 24, 21]],

       [[29, 23, 23],
        [30, 24, 24],
        [32, 24, 23],
        ...,
        [27, 24, 22],
        [27, 23, 21],
        [26, 22, 20]]], dtype=uint8)
```

```
In [19]: print(X_train[0][0].shape)
         X_train[0][0]
```

```
(32, 3)
```

```
Out[19]: array([[28, 25, 24],
                [27, 24, 23],
                [27, 24, 22],
                [27, 24, 22],
                [27, 25, 23],
                [29, 27, 25],
                [49, 39, 37],
                [53, 33, 31],
                [49, 28, 28],
                [54, 41, 42],
                [80, 75, 78],
                [92, 91, 96],
                [72, 76, 83],
                [68, 74, 83],
                [81, 87, 94],
                [91, 97, 94],
                [78, 81, 67],
                [65, 65, 58],
                [53, 50, 50],
                [49, 43, 47],
                [59, 49, 52],
                [76, 55, 57],
                [65, 32, 34],
                [63, 32, 35],
                [60, 37, 38],
                [51, 34, 33],
                [41, 29, 24],
                [36, 28, 24],
                [34, 28, 24],
                [32, 28, 24],
                [31, 27, 25],
                [31, 27, 26]], dtype=uint8)
```

```
In [20]: X_train[34798][31][31][2]
```

```
Out[20]: 80
```

1.3.1 3.1 Normalize the data

Minimally, the image data should be normalized so that the data has mean zero and equal variance. For image data, `(pixel - 128)/ 128` is a quick way to approximately normalize the data and can be used in this project.

```
In [21]: X_train[0][0][0][0]
```

```
Out[21]: 28
```

```
In [22]: x = (X_train[0][0][0][0]-128)/128
         x
```

```
Out[22]: -0.78125
```

```
In [23]: X_train_norm = []

         for i in range(0,n_train):
             for j in range(0,32):
                 for k in range(0,32):
                     for l in range(0,3):
                         X_train_norm.append((X_train[i][j][k][l]-128)/128)

         X_train_norm = np.asarray(X_train_norm)
         X_train_norm = X_train_norm.reshape(34799, 32, 32, 3)
         print(X_train_norm.shape)
         X_train_norm
```

```
(34799, 32, 32, 3)
```

```
Out[23]: array([[[[-0.78125  , -0.8046875, -0.8125   ],
                  [-0.7890625, -0.8125   , -0.8203125],
                  [-0.7890625, -0.8125   , -0.828125 ],
                  ...,
                  [-0.75     , -0.78125  , -0.8125   ],
                  [-0.7578125, -0.7890625, -0.8046875],
                  [-0.7578125, -0.7890625, -0.796875 ]],

                 [[-0.7734375, -0.796875 , -0.8046875],
                  [-0.7890625, -0.8046875, -0.8203125],
                  [-0.7890625, -0.8046875, -0.8203125],
                  ...,
                  [-0.75     , -0.78125  , -0.8125   ],
                  [-0.7578125, -0.7890625, -0.8125   ],
                  [-0.765625 , -0.7890625, -0.8046875]],

                 [[-0.78125  , -0.796875 , -0.796875 ],
                  [-0.7890625, -0.8046875, -0.8203125],
                  [-0.796875 , -0.8046875, -0.8203125],
                  ...,
                  [-0.75     , -0.78125  , -0.8125   ],
                  [-0.7578125, -0.7890625, -0.8125   ],
                  [-0.765625 , -0.7890625, -0.8046875]],

                 ...,
                 [[-0.7890625, -0.8125   , -0.8203125],
                  [-0.78125  , -0.8046875, -0.8125   ],
                  [-0.765625 , -0.8046875, -0.8125   ],
```

```
 ...,
 [-0.7890625, -0.8125   , -0.8203125],
 [-0.78125  , -0.8125   , -0.828125 ],
 [-0.7734375, -0.8046875, -0.828125 ]],

[[-0.78125  , -0.8203125, -0.8203125],
 [-0.7734375, -0.8125   , -0.8125   ],
 [-0.7578125, -0.8046875, -0.8125   ],
 ...,
 [-0.7890625, -0.8125   , -0.8203125],
 [-0.78125  , -0.8125   , -0.828125 ],
 [-0.78125  , -0.8125   , -0.8359375]],

[[-0.7734375, -0.8203125, -0.8203125],
 [-0.765625 , -0.8125   , -0.8125   ],
 [-0.75     , -0.8125   , -0.8203125],
 ...,
 [-0.7890625, -0.8125   , -0.828125 ],
 [-0.7890625, -0.8203125, -0.8359375],
 [-0.796875 , -0.828125 , -0.84375  ]]],


[[[-0.78125  , -0.8125   , -0.8125   ],
  [-0.796875 , -0.8203125, -0.8203125],
  [-0.7890625, -0.8125   , -0.8125   ],
  ...,
  [-0.7578125, -0.78125  , -0.796875 ],
  [-0.7578125, -0.78125  , -0.7890625],
  [-0.75     , -0.78125  , -0.7890625]],

 [[-0.7890625, -0.8125   , -0.8125   ],
  [-0.7890625, -0.8125   , -0.8125   ],
  [-0.78125  , -0.8046875, -0.8125   ],
  ...,
  [-0.7578125, -0.7890625, -0.8046875],
  [-0.7578125, -0.7890625, -0.796875 ],
  [-0.7421875, -0.7734375, -0.7890625]],

 [[-0.796875 , -0.8125   , -0.8125   ],
  [-0.796875 , -0.8125   , -0.8125   ],
  [-0.7890625, -0.8125   , -0.8203125],
  ...,
  [-0.7578125, -0.796875 , -0.8046875],
  [-0.7578125, -0.7890625, -0.796875 ],
  [-0.7421875, -0.7734375, -0.7890625]],

 ...,
 [[-0.78125  , -0.8046875, -0.8203125],
```

```
 [-0.765625 , -0.7890625, -0.8125   ],
 [-0.765625 , -0.7890625, -0.8125   ],
 ...,
 [-0.7890625, -0.8125   , -0.828125 ],
 [-0.7890625, -0.8125   , -0.828125 ],
 [-0.78125  , -0.8125   , -0.828125 ]],

[[-0.7890625, -0.8125   , -0.828125 ],
 [-0.7734375, -0.796875 , -0.8203125],
 [-0.7578125, -0.796875 , -0.8125   ],
 ...,
 [-0.796875 , -0.8203125, -0.8359375],
 [-0.7890625, -0.8125   , -0.828125 ],
 [-0.78125  , -0.8046875, -0.8203125]],

[[-0.78125  , -0.8125   , -0.8203125],
 [-0.78125  , -0.8125   , -0.828125 ],
 [-0.7734375, -0.8125   , -0.828125 ],
 ...,
 [-0.7890625, -0.8203125, -0.828125 ],
 [-0.7890625, -0.8125   , -0.8203125],
 [-0.7734375, -0.796875 , -0.8046875]]],


[[[-0.7734375, -0.8046875, -0.8046875],
  [-0.7734375, -0.796875 , -0.796875 ],
  [-0.765625 , -0.7890625, -0.7890625],
  ...,
  [-0.7578125, -0.7890625, -0.8125   ],
  [-0.7578125, -0.78125  , -0.8046875],
  [-0.75     , -0.7734375, -0.7890625]],

 [[-0.7890625, -0.8125   , -0.8125   ],
  [-0.7890625, -0.8046875, -0.8046875],
  [-0.78125  , -0.796875 , -0.796875 ],
  ...,
  [-0.7578125, -0.7890625, -0.8203125],
  [-0.75     , -0.78125  , -0.8046875],
  [-0.7421875, -0.765625 , -0.7890625]],

 [[-0.7890625, -0.8125   , -0.8125   ],
  [-0.78125  , -0.796875 , -0.796875 ],
  [-0.7734375, -0.7890625, -0.7890625],
  ...,
  [-0.75     , -0.78125  , -0.8125   ],
  [-0.75     , -0.78125  , -0.8125   ],
  [-0.7421875, -0.7734375, -0.796875 ]],
```

```
 ...,
 [[-0.78125  , -0.796875 , -0.828125 ],
  [-0.7734375, -0.796875 , -0.8359375],
  [-0.7578125, -0.796875 , -0.828125 ],
  ...,
  [-0.7734375, -0.8125   , -0.8359375],
  [-0.78125  , -0.8203125, -0.84375  ],
  [-0.78125  , -0.8203125, -0.828125 ]],

 [[-0.7890625, -0.796875 , -0.8203125],
  [-0.78125  , -0.8046875, -0.8359375],
  [-0.765625 , -0.8046875, -0.828125 ],
  ...,
  [-0.78125  , -0.8203125, -0.8359375],
  [-0.7890625, -0.828125 , -0.84375  ],
  [-0.78125  , -0.8125   , -0.828125 ]],

 [[-0.7734375, -0.796875 , -0.8203125],
  [-0.78125  , -0.8125   , -0.8359375],
  [-0.7734375, -0.8125   , -0.8359375],
  ...,
  [-0.7734375, -0.8046875, -0.8203125],
  [-0.78125  , -0.8125   , -0.828125 ],
  [-0.765625 , -0.796875 , -0.8125   ]]],


...,
[[[-0.6015625, -0.4765625, -0.328125 ],
  [-0.5703125, -0.5390625, -0.4453125],
  [-0.4140625, -0.3671875, -0.28125  ],
  ...,
  [ 0.953125 ,  0.9375   ,  0.8984375],
  [ 0.6171875,  0.65625  ,  0.8203125],
  [-0.0546875, -0.09375  ,  0.09375  ]],

 [[-0.7265625, -0.671875 , -0.6171875],
  [-0.625    , -0.6328125, -0.6015625],
  [-0.2890625, -0.25     , -0.1171875],
  ...,
  [ 0.71875  ,  0.75     ,  0.765625 ],
  [ 0.3203125,  0.3828125,  0.4609375],
  [-0.34375  , -0.3203125, -0.21875  ]],

 [[-0.7890625, -0.796875 , -0.7734375],
  [-0.6796875, -0.703125 , -0.6953125],
  [-0.5703125, -0.5      , -0.390625 ],
  ...,
  [-0.046875 ,  0.1171875,  0.25     ],
```

```
  [-0.2421875, -0.1875   ,  0.0078125],
  [-0.5390625, -0.5390625, -0.5625   ]],

 ...,
 [[-0.8125   , -0.8203125, -0.7890625],
  [-0.8359375, -0.84375  , -0.7890625],
  [-0.84375  , -0.8515625, -0.828125 ],
  ...,
  [-0.40625  , -0.3828125, -0.3515625],
  [-0.578125 , -0.5      , -0.3984375],
  [-0.6484375, -0.6015625, -0.4921875]],

 [[-0.7578125, -0.7578125, -0.7421875],
  [-0.828125 , -0.8203125, -0.7734375],
  [-0.84375  , -0.859375 , -0.8359375],
  ...,
  [-0.484375 , -0.4765625, -0.34375  ],
  [-0.5625   , -0.4921875, -0.421875 ],
  [-0.6484375, -0.5703125, -0.4296875]],

 [[-0.78125  , -0.78125  , -0.765625 ],
  [-0.828125 , -0.8359375, -0.8046875],
  [-0.8515625, -0.859375 , -0.8515625],
  ...,
  [-0.4765625, -0.5078125, -0.40625  ],
  [-0.6953125, -0.6484375, -0.5703125],
  [-0.75     , -0.7109375, -0.6328125]]],


[[[-0.359375 , -0.390625 , -0.25     ],
  [-0.0625   , -0.015625 ,  0.15625  ],
  [-0.125    , -0.0234375,  0.140625 ],
  ...,
  [ 0.4453125,  0.421875 ,  0.3828125],
  [ 0.59375  ,  0.640625 ,  0.6796875],
  [ 0.03125  , -0.109375 , -0.0546875]],

 [[-0.421875 , -0.3984375, -0.2734375],
  [ 0.3359375,  0.359375 ,  0.4453125],
  [ 0.0703125,  0.28125  ,  0.4375   ],
  ...,
  [ 0.40625  ,  0.4140625,  0.4609375],
  [ 0.546875 ,  0.5625   ,  0.6640625],
  [-0.3828125, -0.3359375, -0.3359375]],

 [[-0.578125 , -0.609375 , -0.5625   ],
  [-0.21875  , -0.171875 , -0.078125 ],
  [-0.0859375,  0.03125  ,  0.234375 ],
```

```
 ...,
 [ 0.2265625,  0.1953125,  0.25     ],
 [ 0.375    ,  0.4296875,  0.5546875],
 [-0.3125   , -0.375    , -0.359375 ]],

...,
[[-0.828125 , -0.8359375, -0.828125 ],
 [-0.84375  , -0.8515625, -0.8515625],
 [-0.859375 , -0.875    , -0.8515625],
 ...,
 [-0.609375 , -0.609375 , -0.53125  ],
 [-0.7109375, -0.6640625, -0.59375  ],
 [-0.765625 , -0.6796875, -0.5390625]],

[[-0.859375 , -0.875    , -0.859375 ],
 [-0.8515625, -0.8671875, -0.859375 ],
 [-0.8515625, -0.859375 , -0.8359375],
 ...,
 [-0.71875  , -0.65625  , -0.53125  ],
 [-0.7421875, -0.71875  , -0.625    ],
 [-0.71875  , -0.6640625, -0.5234375]],

[[-0.859375 , -0.8671875, -0.84375  ],
 [-0.8359375, -0.84375  , -0.8203125],
 [-0.8125   , -0.828125 , -0.8046875],
 ...,
 [-0.75     , -0.734375 , -0.6796875],
 [-0.6484375, -0.671875 , -0.625    ],
 [-0.6796875, -0.6640625, -0.59375  ]]],


[[[-0.4609375, -0.3828125, -0.25     ],
 [-0.8125   , -0.796875 , -0.78125  ],
 [-0.6875   , -0.671875 , -0.6484375],
 ...,
 [ 0.7578125,  0.828125 ,  0.8515625],
 [ 0.1796875,  0.2578125,  0.296875 ],
 [ 0.28125  ,  0.265625 ,  0.3203125]],

[[-0.2265625, -0.140625 , -0.0234375],
 [-0.71875  , -0.6796875, -0.6328125],
 [-0.5078125, -0.5625   , -0.515625 ],
 ...,
 [ 0.5390625,  0.6875   ,  0.75     ],
 [ 0.203125 ,  0.2734375,  0.3203125],
 [ 0.28125  ,  0.2734375,  0.2421875]],

[[-0.1875   , -0.1640625, -0.1171875],
```

```
                [-0.734375 , -0.7109375, -0.6875   ],
                [-0.4375   , -0.453125 , -0.3984375],
                ...,
                [ 0.7421875,  0.8515625,  0.8359375],
                [ 0.4140625,  0.5      ,  0.546875 ],
                [ 0.296875 ,  0.3046875,  0.2421875]],

               ...,
               [[-0.8359375, -0.84375  , -0.8203125],
                [-0.8203125, -0.8125   , -0.765625 ],
                [-0.8515625, -0.84375  , -0.8125   ],
                ...,
                [-0.6484375, -0.6328125, -0.578125 ],
                [-0.546875 , -0.515625 , -0.453125 ],
                [-0.546875 , -0.453125 , -0.359375 ]],

               [[-0.859375 , -0.8671875, -0.8359375],
                [-0.8515625, -0.8515625, -0.8125   ],
                [-0.859375 , -0.859375 , -0.8203125],
                ...,
                [-0.71875  , -0.71875  , -0.6875   ],
                [-0.546875 , -0.5390625, -0.453125 ],
                [-0.5234375, -0.4609375, -0.3671875]],

               [[-0.8671875, -0.875    , -0.8515625],
                [-0.875    , -0.8828125, -0.859375 ],
                [-0.875    , -0.8828125, -0.859375 ],
                ...,
                [-0.6875   , -0.6875   , -0.65625  ],
                [-0.5546875, -0.515625 , -0.4296875],
                [-0.5546875, -0.46875  , -0.375    ]]]])
```

In [24]:
```
print(X_train_norm.min())
print(X_train_norm.max())
```

```
-1.0
0.9921875
```

In [25]:
```
X_valid_norm = []

for i in range(0,n_validation):
    for j in range(0,32):
        for k in range(0,32):
            for l in range(0,3):
                X_valid_norm.append((X_valid[i][j][k][l]-128)/128)

X_valid_norm = np.asarray(X_valid_norm)
```

```python
X_valid_norm = X_valid_norm.reshape(4410, 32, 32, 3)
print(X_valid_norm.shape)
X_valid_norm
```

```
(4410, 32, 32, 3)
```

```
Out[25]: array([[[[-0.8984375, -0.90625  , -0.90625  ],
                 [-0.90625  , -0.9140625, -0.90625  ],
                 [-0.8984375, -0.9140625, -0.9140625],
                 ...,
                 [-0.890625 , -0.90625  , -0.9140625],
                 [-0.8984375, -0.90625  , -0.9140625],
                 [-0.90625  , -0.90625  , -0.9140625]],

                [[-0.8984375, -0.90625  , -0.8984375],
                 [-0.90625  , -0.9140625, -0.90625  ],
                 [-0.8984375, -0.90625  , -0.90625  ],
                 ...,
                 [-0.8984375, -0.90625  , -0.9140625],
                 [-0.8984375, -0.90625  , -0.9140625],
                 [-0.90625  , -0.90625  , -0.9140625]],

                [[-0.8984375, -0.90625  , -0.8984375],
                 [-0.90625  , -0.9140625, -0.90625  ],
                 [-0.90625  , -0.90625  , -0.90625  ],
                 ...,
                 [-0.8984375, -0.90625  , -0.9140625],
                 [-0.8984375, -0.90625  , -0.9140625],
                 [-0.90625  , -0.90625  , -0.9140625]],


                ...,
                [[-0.875    , -0.8828125, -0.8828125],
                 [-0.8828125, -0.890625 , -0.890625 ],
                 [-0.8828125, -0.8828125, -0.8828125],
                 ...,
                 [-0.8984375, -0.90625  , -0.90625  ],
                 [-0.890625 , -0.8984375, -0.8984375],
                 [-0.890625 , -0.8984375, -0.8984375]],

                [[-0.859375 , -0.875    , -0.875    ],
                 [-0.8671875, -0.8828125, -0.8828125],
                 [-0.859375 , -0.875    , -0.875    ],
                 ...,
                 [-0.90625  , -0.9140625, -0.9140625],
                 [-0.8984375, -0.90625  , -0.90625  ],
                 [-0.890625 , -0.8984375, -0.8984375]],
```

```
 [[-0.8515625, -0.8671875, -0.8671875],
  [-0.859375 , -0.875    , -0.875    ],
  [-0.8515625, -0.8671875, -0.8671875],
  ...,
  [-0.90625  , -0.90625  , -0.90625  ],
  [-0.8984375, -0.8984375, -0.8984375],
  [-0.8828125, -0.890625 , -0.890625 ]]],


[[[-0.890625 , -0.90625  , -0.90625  ],
  [-0.90625  , -0.90625  , -0.90625  ],
  [-0.90625  , -0.8984375, -0.8984375],
  ...,
  [-0.8828125, -0.90625  , -0.9140625],
  [-0.890625 , -0.8984375, -0.90625  ],
  [-0.8984375, -0.8984375, -0.90625  ]],

 [[-0.8984375, -0.90625  , -0.90625  ],
  [-0.90625  , -0.90625  , -0.8984375],
  [-0.8984375, -0.8984375, -0.8984375],
  ...,
  [-0.890625 , -0.8984375, -0.9140625],
  [-0.8828125, -0.890625 , -0.8984375],
  [-0.8984375, -0.8984375, -0.90625  ]],

 [[-0.90625  , -0.90625  , -0.90625  ],
  [-0.90625  , -0.90625  , -0.8984375],
  [-0.90625  , -0.90625  , -0.90625  ],
  ...,
  [-0.8984375, -0.90625  , -0.9140625],
  [-0.890625 , -0.8984375, -0.90625  ],
  [-0.8984375, -0.90625  , -0.90625  ]],

 ...,
 [[-0.8828125, -0.890625 , -0.890625 ],
  [-0.8828125, -0.890625 , -0.890625 ],
  [-0.890625 , -0.8984375, -0.8984375],
  ...,
  [-0.890625 , -0.890625 , -0.890625 ],
  [-0.890625 , -0.8828125, -0.890625 ],
  [-0.8984375, -0.8984375, -0.90625  ]],

 [[-0.875    , -0.8828125, -0.8828125],
  [-0.875    , -0.8828125, -0.8828125],
  [-0.875    , -0.8828125, -0.8828125],
  ...,
  [-0.8984375, -0.8984375, -0.90625  ],
  [-0.8984375, -0.890625 , -0.8984375],
```

```
 [-0.8984375, -0.8984375, -0.90625  ]],

[[-0.875    , -0.875    , -0.875    ],
 [-0.8671875, -0.875    , -0.875    ],
 [-0.859375 , -0.875    , -0.875    ],
 ...,
 [-0.90625  , -0.90625  , -0.9140625],
 [-0.8984375, -0.8984375, -0.90625  ],
 [-0.890625 , -0.890625 , -0.90625  ]]],


[[[-0.8984375, -0.9140625, -0.9140625],
 [-0.90625  , -0.9140625, -0.90625  ],
 [-0.90625  , -0.90625  , -0.90625  ],
 ...,
 [-0.8828125, -0.90625  , -0.90625  ],
 [-0.90625  , -0.90625  , -0.9140625],
 [-0.90625  , -0.8984375, -0.90625  ]],

[[-0.8984375, -0.90625  , -0.90625  ],
 [-0.90625  , -0.9140625, -0.90625  ],
 [-0.8984375, -0.90625  , -0.90625  ],
 ...,
 [-0.890625 , -0.90625  , -0.90625  ],
 [-0.90625  , -0.90625  , -0.9140625],
 [-0.8984375, -0.8984375, -0.9140625]],

[[-0.90625  , -0.9140625, -0.9140625],
 [-0.8984375, -0.90625  , -0.90625  ],
 [-0.890625 , -0.90625  , -0.90625  ],
 ...,
 [-0.90625  , -0.90625  , -0.90625  ],
 [-0.9140625, -0.90625  , -0.9140625],
 [-0.8984375, -0.8984375, -0.90625  ]],


...,
[[-0.8984375, -0.8984375, -0.8984375],
 [-0.890625 , -0.890625 , -0.890625 ],
 [-0.890625 , -0.890625 , -0.890625 ],
 ...,
 [-0.90625  , -0.90625  , -0.8984375],
 [-0.890625 , -0.890625 , -0.8984375],
 [-0.8984375, -0.8984375, -0.90625  ]],

[[-0.875    , -0.875    , -0.8828125],
 [-0.875    , -0.875    , -0.8828125],
 [-0.8828125, -0.875    , -0.8828125],
 ...,
```

```
  [-0.9140625, -0.9140625, -0.9140625],
  [-0.90625  , -0.90625  , -0.90625  ],
  [-0.890625 , -0.890625 , -0.8984375]],

 [[-0.8515625, -0.859375 , -0.8671875],
  [-0.8671875, -0.8671875, -0.875    ],
  [-0.8671875, -0.8671875, -0.875    ],
  ...,
  [-0.90625  , -0.90625  , -0.90625  ],
  [-0.8984375, -0.8984375, -0.8984375],
  [-0.8984375, -0.8984375, -0.90625  ]]],


...,
[[[-0.515625 , -0.5      , -0.4921875],
  [-0.5234375, -0.515625 , -0.484375 ],
  [-0.5078125, -0.5234375, -0.515625 ],
  ...,
  [-0.6171875, -0.6328125, -0.625    ],
  [-0.6328125, -0.640625 , -0.6171875],
  [-0.6328125, -0.6328125, -0.6171875]],

 [[-0.546875 , -0.53125  , -0.515625 ],
  [-0.5546875, -0.5390625, -0.5078125],
  [-0.5390625, -0.5390625, -0.515625 ],
  ...,
  [-0.6015625, -0.6328125, -0.609375 ],
  [-0.6171875, -0.640625 , -0.6171875],
  [-0.625    , -0.640625 , -0.6171875]],

 [[-0.5625   , -0.5703125, -0.5390625],
  [-0.5703125, -0.578125 , -0.546875 ],
  [-0.5703125, -0.578125 , -0.5546875],
  ...,
  [-0.59375  , -0.609375 , -0.6015625],
  [-0.609375 , -0.6328125, -0.609375 ],
  [-0.625    , -0.640625 , -0.6171875]],

 ...,
 [[-0.609375 , -0.6171875, -0.59375  ],
  [-0.5703125, -0.609375 , -0.6015625],
  [-0.609375 , -0.640625 , -0.6328125],
  ...,
  [-0.6484375, -0.6640625, -0.6640625],
  [-0.640625 , -0.65625  , -0.6640625],
  [-0.640625 , -0.6640625, -0.6640625]],

 [[-0.5546875, -0.6171875, -0.59375  ],
```

```
 [-0.5625   , -0.6171875, -0.609375 ],
 [-0.5625   , -0.6015625, -0.5859375],
 ...,
 [-0.6484375, -0.671875 , -0.6640625],
 [-0.6484375, -0.6640625, -0.65625  ],
 [-0.6484375, -0.6640625, -0.65625  ]],

[[-0.53125  , -0.609375 , -0.6015625],
 [-0.546875 , -0.6328125, -0.6171875],
 [-0.546875 , -0.6328125, -0.625    ],
 ...,
 [-0.65625  , -0.6640625, -0.65625  ],
 [-0.640625 , -0.65625  , -0.65625  ],
 [-0.640625 , -0.65625  , -0.65625  ]]],


[[[-0.4921875, -0.484375 , -0.46875  ],
  [-0.4921875, -0.5078125, -0.4609375],
  [-0.4765625, -0.4921875, -0.4765625],
  ...,
  [-0.6640625, -0.671875 , -0.65625  ],
  [-0.671875 , -0.671875 , -0.65625  ],
  [-0.671875 , -0.6796875, -0.671875 ]],

 [[-0.4296875, -0.4375   , -0.4140625],
  [-0.4296875, -0.4375   , -0.4140625],
  [-0.421875 , -0.4296875, -0.421875 ],
  ...,
  [-0.671875 , -0.671875 , -0.65625  ],
  [-0.6640625, -0.671875 , -0.65625  ],
  [-0.671875 , -0.6796875, -0.671875 ]],

 [[-0.453125 , -0.453125 , -0.4453125],
  [-0.453125 , -0.4453125, -0.4296875],
  [-0.453125 , -0.453125 , -0.453125 ],
  ...,
  [-0.671875 , -0.671875 , -0.65625  ],
  [-0.671875 , -0.6796875, -0.65625  ],
  [-0.6796875, -0.6875   , -0.671875 ]],

 ...,
 [[-0.640625 , -0.6484375, -0.6328125],
  [-0.640625 , -0.65625  , -0.640625 ],
  [-0.640625 , -0.65625  , -0.6484375],
  ...,
  [-0.65625  , -0.671875 , -0.671875 ],
  [-0.65625  , -0.671875 , -0.6640625],
  [-0.6640625, -0.6796875, -0.6796875]],
```

```
 [[-0.6328125, -0.6484375, -0.640625 ],
  [-0.6484375, -0.6640625, -0.640625 ],
  [-0.640625 , -0.6640625, -0.6484375],
  ...,
  [-0.65625  , -0.671875 , -0.671875 ],
  [-0.65625  , -0.6796875, -0.671875 ],
  [-0.65625  , -0.6875   , -0.6796875]],

 [[-0.59375  , -0.625    , -0.625    ],
  [-0.640625 , -0.65625  , -0.6484375],
  [-0.640625 , -0.6640625, -0.640625 ],
  ...,
  [-0.65625  , -0.671875 , -0.671875 ],
  [-0.65625  , -0.6796875, -0.671875 ],
  [-0.671875 , -0.6875   , -0.6796875]]],


[[[-0.484375 , -0.4921875, -0.484375 ],
  [-0.5      , -0.515625 , -0.5078125],
  [-0.5      , -0.5      , -0.4921875],
  ...,
  [-0.7109375, -0.7109375, -0.703125 ],
  [-0.7109375, -0.7109375, -0.703125 ],
  [-0.703125 , -0.7109375, -0.6953125]],

 [[-0.546875 , -0.546875 , -0.546875 ],
  [-0.53125  , -0.5625   , -0.5625   ],
  [-0.5703125, -0.5703125, -0.5625   ],
  ...,
  [-0.7109375, -0.7109375, -0.703125 ],
  [-0.7109375, -0.71875  , -0.703125 ],
  [-0.71875  , -0.71875  , -0.7109375]],

 [[-0.5      , -0.5      , -0.4921875],
  [-0.4921875, -0.4921875, -0.484375 ],
  [-0.4921875, -0.5      , -0.4765625],
  ...,
  [-0.7109375, -0.7109375, -0.7109375],
  [-0.7109375, -0.71875  , -0.703125 ],
  [-0.7109375, -0.71875  , -0.7109375]],

 ...,
 [[-0.671875 , -0.6796875, -0.6640625],
  [-0.671875 , -0.6796875, -0.671875 ],
  [-0.6796875, -0.6875   , -0.671875 ],
  ...,
  [-0.7265625, -0.734375 , -0.7265625],
```

```
                [-0.734375 , -0.7421875, -0.734375 ],
                [-0.734375 , -0.7421875, -0.734375 ]],

               [[-0.6640625, -0.671875 , -0.6640625],
                [-0.6640625, -0.671875 , -0.671875 ],
                [-0.671875 , -0.6796875, -0.671875 ],
                ...,
                [-0.734375 , -0.7421875, -0.7421875],
                [-0.734375 , -0.734375 , -0.7265625],
                [-0.7265625, -0.7421875, -0.734375 ]],

               [[-0.65625  , -0.671875 , -0.65625  ],
                [-0.6640625, -0.671875 , -0.6640625],
                [-0.6640625, -0.671875 , -0.671875 ],
                ...,
                [-0.734375 , -0.7421875, -0.734375 ],
                [-0.734375 , -0.734375 , -0.734375 ],
                [-0.734375 , -0.734375 , -0.7265625]]]])
```

```
In [26]: print(X_valid_norm.min())
         print(X_valid_norm.max())
```

```
-1.0
0.9921875
```

```
In [27]: X_test_norm = []

         for i in range(0,n_test):
             for j in range(0,32):
                 for k in range(0,32):
                     for l in range(0,3):
                         X_test_norm.append((X_test[i][j][k][l]-128)/128)

         X_test_norm = np.asarray(X_test_norm)
         X_test_norm = X_test_norm.reshape(n_test, 32, 32, 3)
         print(X_test_norm.shape)
         X_test_norm
```

```
(12630, 32, 32, 3)
```

```
Out[27]: array([[[[-0.09375  ,  0.0859375,  0.359375 ],
                 [-0.09375  ,  0.0703125,  0.3359375],
                 [-0.078125 ,  0.078125 ,  0.34375  ],
                 ...,
                 [-0.234375 , -0.109375 ,  0.1171875],
                 [-0.2421875, -0.0546875,  0.1484375],
                 [-0.3359375, -0.1796875,  0.015625 ]],
```

```
 [[-0.1015625,  0.109375 ,  0.375    ],
  [-0.1015625,  0.09375  ,  0.359375 ],
  [-0.0859375,  0.1015625,  0.3671875],
  ...,
  [-0.0703125,  0.1171875,  0.375    ],
  [-0.0625   ,  0.1015625,  0.3515625],
  [-0.0625   ,  0.0859375,  0.3359375]],

 [[-0.0859375,  0.1015625,  0.359375 ],
  [-0.0859375,  0.109375 ,  0.3671875],
  [-0.109375 ,  0.09375  ,  0.34375  ],
  ...,
  [-0.0625   ,  0.125    ,  0.3984375],
  [-0.046875 ,  0.125    ,  0.3984375],
  [-0.0625   ,  0.1171875,  0.3828125]],


 ...,
 [[-0.0859375,  0.0703125,  0.3046875],
  [-0.1015625,  0.046875 ,  0.28125  ],
  [-0.0859375,  0.0546875,  0.28125  ],
  ...,
  [-0.1015625,  0.0703125,  0.3203125],
  [-0.09375  ,  0.0625   ,  0.3125   ],
  [-0.078125 ,  0.0859375,  0.34375  ]],

 [[-0.09375  ,  0.0625   ,  0.296875 ],
  [-0.109375 ,  0.046875 ,  0.3125   ],
  [-0.1171875,  0.0234375,  0.296875 ],
  ...,
  [-0.1171875,  0.0546875,  0.3046875],
  [-0.0859375,  0.0625   ,  0.2890625],
  [-0.1015625,  0.0859375,  0.3046875]],

 [[-0.125    ,  0.0546875,  0.3046875],
  [-0.140625 ,  0.046875 ,  0.2890625],
  [-0.0703125,  0.0546875,  0.2890625],
  ...,
  [-0.1015625,  0.078125 ,  0.3046875],
  [-0.0859375,  0.078125 ,  0.3046875],
  [-0.109375 ,  0.09375  ,  0.328125 ]]],


[[[-0.5390625, -0.453125 , -0.5234375],
  [-0.3203125, -0.375    , -0.5078125],
  [-0.28125  , -0.3671875, -0.5078125],
  ...,
  [-0.3984375, -0.5      , -0.4765625],
```

```
 [-0.4453125, -0.4921875, -0.4765625],
 [-0.421875 , -0.4609375, -0.46875  ]],

[[-0.5390625, -0.46875  , -0.546875 ],
 [-0.265625 , -0.3515625, -0.4921875],
 [-0.2578125, -0.375    , -0.515625 ],
 ...,
 [-0.3984375, -0.5      , -0.484375 ],
 [-0.28125  , -0.4609375, -0.453125 ],
 [-0.3359375, -0.46875  , -0.46875  ]],

[[-0.5625   , -0.46875  , -0.546875 ],
 [-0.296875 , -0.34375  , -0.4765625],
 [-0.2578125, -0.359375 , -0.5      ],
 ...,
 [-0.4375   , -0.4921875, -0.484375 ],
 [-0.375    , -0.4765625, -0.4609375],
 [-0.3828125, -0.4609375, -0.453125 ]],

...,
[[-0.625    , -0.6328125, -0.6640625],
 [-0.3828125, -0.46875  , -0.5546875],
 [-0.375    , -0.484375 , -0.5625   ],
 ...,
 [-0.46875  , -0.5078125, -0.5078125],
 [-0.390625 , -0.4921875, -0.4921875],
 [-0.40625  , -0.484375 , -0.46875  ]],

[[-0.6171875, -0.640625 , -0.6484375],
 [-0.390625 , -0.484375 , -0.5234375],
 [-0.390625 , -0.4921875, -0.546875 ],
 ...,
 [-0.53125  , -0.53125  , -0.5234375],
 [-0.46875  , -0.4921875, -0.4921875],
 [-0.4296875, -0.4765625, -0.4765625]],

[[-0.6015625, -0.6484375, -0.6484375],
 [-0.390625 , -0.484375 , -0.5234375],
 [-0.3828125, -0.484375 , -0.546875 ],
 ...,
 [-0.515625 , -0.5390625, -0.515625 ],
 [-0.4296875, -0.4765625, -0.453125 ],
 [-0.4140625, -0.453125 , -0.453125 ]]],


[[[-0.59375  , -0.6875   , -0.703125 ],
  [-0.6015625, -0.6953125, -0.7109375],
  [-0.609375 , -0.703125 , -0.7109375],
```

```
  ...,
  [-0.5859375, -0.65625  , -0.640625 ],
  [-0.578125 , -0.6484375, -0.625    ],
  [-0.5703125, -0.6484375, -0.625    ]],

 [[-0.5390625, -0.6484375, -0.671875 ],
  [-0.5390625, -0.65625  , -0.6796875],
  [-0.5625   , -0.65625  , -0.6875   ],
  ...,
  [-0.59375  , -0.6796875, -0.671875 ],
  [-0.578125 , -0.65625  , -0.6484375],
  [-0.5703125, -0.640625 , -0.6484375]],

 [[-0.4765625, -0.6171875, -0.65625  ],
  [-0.4921875, -0.609375 , -0.6484375],
  [-0.4921875, -0.6171875, -0.6484375],
  ...,
  [-0.5546875, -0.671875 , -0.6953125],
  [-0.5703125, -0.671875 , -0.6875   ],
  [-0.5625   , -0.6640625, -0.6875   ]],


 ...,
 [[-0.78125  , -0.8125   , -0.796875 ],
  [-0.7734375, -0.8046875, -0.7890625],
  [-0.7734375, -0.7890625, -0.765625 ],
  ...,
  [-0.7890625, -0.8125   , -0.8125   ],
  [-0.78125  , -0.8046875, -0.8125   ],
  [-0.7734375, -0.7890625, -0.7890625]],

 [[-0.7578125, -0.78125  , -0.7734375],
  [-0.7109375, -0.75     , -0.734375 ],
  [-0.6796875, -0.7265625, -0.7109375],
  ...,
  [-0.734375 , -0.7578125, -0.78125  ],
  [-0.671875 , -0.6875   , -0.703125 ],
  [-0.6328125, -0.6484375, -0.6484375]],

 [[-0.75     , -0.7734375, -0.765625 ],
  [-0.7421875, -0.78125  , -0.78125  ],
  [-0.65625  , -0.734375 , -0.75     ],
  ...,
  [-0.6796875, -0.6875   , -0.703125 ],
  [-0.65625  , -0.6640625, -0.6640625],
  [-0.6328125, -0.640625 , -0.65625  ]]],


...,
```

```
[[[-0.8125    , -0.7890625, -0.734375 ],
  [-0.8203125, -0.8046875, -0.75     ],
  [-0.8125    , -0.8046875, -0.75     ],
  ...,
  [-0.796875 , -0.7890625, -0.7421875],
  [-0.796875 , -0.7890625, -0.734375 ],
  [-0.796875 , -0.7890625, -0.734375 ]],

 [[-0.8046875, -0.796875 , -0.7421875],
  [-0.8046875, -0.796875 , -0.734375 ],
  [-0.8046875, -0.796875 , -0.7421875],
  ...,
  [-0.796875 , -0.7890625, -0.734375 ],
  [-0.7890625, -0.7890625, -0.7265625],
  [-0.78125  , -0.78125  , -0.71875  ]],

 [[-0.796875 , -0.796875 , -0.75     ],
  [-0.8046875, -0.8046875, -0.75     ],
  [-0.8046875, -0.8046875, -0.7578125],
  ...,
  [-0.796875 , -0.7890625, -0.7421875],
  [-0.7890625, -0.7890625, -0.7421875],
  [-0.7890625, -0.796875 , -0.734375 ]],


 ...,
 [[-0.875    , -0.8671875, -0.8359375],
  [-0.8828125, -0.875    , -0.8359375],
  [-0.890625 , -0.875    , -0.8359375],
  ...,
  [-0.8515625, -0.8359375, -0.765625 ],
  [-0.84375  , -0.828125 , -0.75     ],
  [-0.8515625, -0.8359375, -0.765625 ]],

 [[-0.875    , -0.8671875, -0.828125 ],
  [-0.8828125, -0.875    , -0.8359375],
  [-0.890625 , -0.875    , -0.8359375],
  ...,
  [-0.8515625, -0.8359375, -0.765625 ],
  [-0.84375  , -0.828125 , -0.75     ],
  [-0.8515625, -0.828125 , -0.75     ]],

 [[-0.8828125, -0.875    , -0.8359375],
  [-0.8828125, -0.875    , -0.8359375],
  [-0.890625 , -0.875    , -0.8359375],
  ...,
  [-0.859375 , -0.8359375, -0.765625 ],
  [-0.8515625, -0.828125 , -0.75     ],
  [-0.859375 , -0.828125 , -0.7578125]]],
```

```
[[[-0.6328125, -0.546875 , -0.4296875],
  [-0.6171875, -0.5859375, -0.4765625],
  [-0.5390625, -0.5546875, -0.5078125],
  ...,
  [-0.6953125, -0.71875  , -0.65625  ],
  [-0.90625  , -0.921875 , -0.90625  ],
  [-0.921875 , -0.9296875, -0.90625  ]],

 [[-0.5234375, -0.46875  , -0.34375  ],
  [-0.5234375, -0.546875 , -0.4375   ],
  [-0.53125  , -0.5546875, -0.4765625],
  ...,
  [-0.796875 , -0.8203125, -0.7578125],
  [-0.90625  , -0.921875 , -0.9375   ],
  [-0.9140625, -0.9296875, -0.9140625]],

 [[-0.484375 , -0.4140625, -0.2734375],
  [-0.4375   , -0.3828125, -0.2265625],
  [-0.4765625, -0.4296875, -0.265625 ],
  ...,
  [-0.71875  , -0.71875  , -0.703125 ],
  [-0.8046875, -0.8359375, -0.890625 ],
  [-0.8984375, -0.9296875, -0.921875 ]],


 ...,
 [[-0.875    , -0.875    , -0.8515625],
  [-0.8828125, -0.8828125, -0.859375 ],
  [-0.890625 , -0.8984375, -0.8671875],
  ...,
  [-0.75     , -0.7578125, -0.7109375],
  [-0.71875  , -0.7109375, -0.65625  ],
  [-0.8125   , -0.8125   , -0.8203125]],

 [[-0.8984375, -0.90625  , -0.8828125],
  [-0.8984375, -0.90625  , -0.890625 ],
  [-0.90625  , -0.9140625, -0.8984375],
  ...,
  [-0.796875 , -0.78125  , -0.7421875],
  [-0.75     , -0.7265625, -0.6953125],
  [-0.78125  , -0.7890625, -0.8125   ]],

 [[-0.90625  , -0.9140625, -0.890625 ],
  [-0.90625  , -0.9140625, -0.890625 ],
  [-0.90625  , -0.921875 , -0.90625  ],
  ...,
  [-0.78125  , -0.75     , -0.6875   ],
```

```
 [-0.71875  , -0.7109375, -0.6875   ],
 [-0.7734375, -0.7890625, -0.828125 ]]],


[[[-0.921875 , -0.921875 , -0.8984375],
  [-0.921875 , -0.921875 , -0.8984375],
  [-0.921875 , -0.9296875, -0.90625  ],
  ...,
  [-0.8984375, -0.90625  , -0.8828125],
  [-0.8984375, -0.90625  , -0.8828125],
  [-0.8984375, -0.90625  , -0.875    ]],

 [[-0.90625  , -0.9140625, -0.890625 ],
  [-0.9296875, -0.9296875, -0.90625  ],
  [-0.9296875, -0.9296875, -0.90625  ],
  ...,
  [-0.8984375, -0.90625  , -0.890625 ],
  [-0.8984375, -0.90625  , -0.8828125],
  [-0.8984375, -0.90625  , -0.875    ]],

 [[-0.9140625, -0.9296875, -0.9140625],
  [-0.9296875, -0.9375   , -0.9140625],
  [-0.9296875, -0.9296875, -0.90625  ],
  ...,
  [-0.8984375, -0.90625  , -0.890625 ],
  [-0.8984375, -0.90625  , -0.8828125],
  [-0.8984375, -0.90625  , -0.875    ]],


 ...,
 [[-0.9140625, -0.9296875, -0.8984375],
  [-0.921875 , -0.9296875, -0.90625  ],
  [-0.9140625, -0.921875 , -0.8984375],
  ...,
  [-0.9140625, -0.921875 , -0.8984375],
  [-0.921875 , -0.9296875, -0.90625  ],
  [-0.90625  , -0.9140625, -0.890625 ]],

 [[-0.9140625, -0.921875 , -0.890625 ],
  [-0.9140625, -0.921875 , -0.890625 ],
  [-0.9140625, -0.921875 , -0.890625 ],
  ...,
  [-0.9296875, -0.9296875, -0.90625  ],
  [-0.9296875, -0.9296875, -0.9140625],
  [-0.90625  , -0.9140625, -0.8984375]],

 [[-0.921875 , -0.921875 , -0.8984375],
  [-0.921875 , -0.9296875, -0.90625  ],
  [-0.9140625, -0.9296875, -0.90625  ],
```

```
           ...,
           [-0.921875 , -0.921875 , -0.8984375],
           [-0.9296875, -0.921875 , -0.90625  ],
           [-0.9140625, -0.9140625, -0.8984375]]]])
```

```
In [28]: print(X_test_norm.min())
         print(X_test_norm.max())
```

```
-1.0
0.9921875
```

1.3.2 3.2 Shuffle the data

By shuffling your data, you ensure that each data point creates an "independent" change on the model, without being biased by the same points before them. Suppose data is sorted in a specified order. For example a data set which is sorted base on their class.

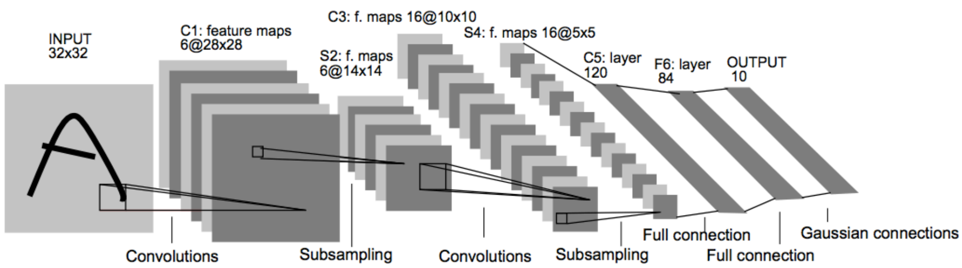
```
In [29]: from sklearn.utils import shuffle
         X_train_norm, y_train = shuffle(X_train_norm, y_train)
         X_valid_norm, y_valid = shuffle(X_valid_norm, y_valid)
         X_test_norm, y_test = shuffle(X_test_norm, y_test)
```

1.4 4. Design and Test the Deep Learning Architecture

There are various aspects to consider when thinking about the Deep Learning Architecture:

- 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.

Example of a published baseline model on this problem.



Source: Yan LeCun

```
In [30]: import tensorflow as tf
         from tensorflow.contrib.layers import flatten
```

1.4.1 4.1 Epochs and Batch Size

- The EPOCH and BATCH_SIZE values affect the training speed and model accuracy
- Epochs are a single forward and backward pass of the whole dataset during TRAINING
- The larger the batch size the faster the model will train, however memory limitations
- Batch_SIZE Is the number of datapoints per batch. Number of batches = number of images / batch_size
- Batch x = images, batch y = Labels

```
In [31]: EPOCHS = 50
         BATCH_SIZE = 128
```

1.4.2 4.2 Placeholders

- input = tf.float32, [bacth_size, image_height, image_width, color_channels]
- batch size is set to NONE for placeholder variable, which allows later to accept a batch of any size

```
In [32]: x = tf.placeholder(tf.float32, (None, 32, 32, 3))
         y = tf.placeholder(tf.int32, (None))
         one_hot_y = tf.one_hot(y, n_classes)
         keep_prob = tf.placeholder(tf.float32)
```

1.4.3 4.3 Model architecture

4.3.1 Weights and Biases - weight = tf.Variable(tf.random_normal([filter_size_height, filter_size_width, color_channels, k_output])) - biases = tf.Variable(tf.random_normal([k_output]))

4.3.2 Strides and Padding - Stride represents at what steps the filter is run over the image - Padding is if the filter overlaps the edges of the image so that zero padding is required - If Padding is Same then the image size will remain the same due to zero padding - IF padding is valid the image size will be smaller, i.e. 32x32 > 28x28 for stride = 1 and valid padding - stride for each dimension (batch_size, height, width, depth) - For both ksize and strides, the batch_size and channel_depth dimensions are typically set to 1. - padding is either 'VALID' or 'SAME'.

4.3.3 Graph Operations - Convolutions - convolution is running the filter over the image at stride - tf.nn.conv2d(x, weights, strides=[1, 1, 1, 1], padding='VALID') + biases - Activation function: tf.nn.relu(conv2d) - Maxpooling - tf.nn.max_pool(x, ksize=[1, k, k, 1], strides=[1, k, k, 1], padding='SAME') - for k = 2 the output image size is half from 28x28 > 14x14 for example - Flattening - flatten(conv2d) - Matrix multiplication - fc = fully connected layer - tf.add(tf.matmul(fc, weights['weights']), biases['biases']) - Activation function: tf.nn.relu(fc) - Optimization/Regularization: tf.nn.dropout(fc, dropout) with dropout being the probability to keep units

4.3.4 Regularization The network thats just the right size for your data is very very hard to optimize. In practive, we always try networks that are way too big for our data and then we try our best to prevent them from overfitting.

Note that Regularization only applies to the fully-connected region of your convnet. If you add dropout between conv layers. It'll only degrade the performance further since conv layers are already very sparse. For conv layers instead you can insert batch normalization between your convolutions. This will regularize your model, as well as make your model more stable during training.

Dropout Regularization - The values that go from one layer to the next are called activations - Randomly, for every example you train your network on, set a set number of the activations to 0, i.e. keep_prob = 0.5 then set half of the activations to 0 - At the same time factor the remaining activations by a factor of 1/keep_prob - Take the consenus ye by averaging the activations yt - TensorFlow provides the **tf.nn.dropout()** function, which you can use to implement dropout.

During training, a good starting value for keep_prob is 0.5.

During testing, use a keep_prob value of 1.0 to keep all units and maximize the power of the model.

Code The tf.nn.dropout() function takes in two parameters:

- hidden_layer: the tensor to which you would like to apply dropout
- keep_prob: the probability of keeping (i.e. not dropping) any given unit
- keep_prob allows you to adjust the number of units to drop. In order to compensate for dropped units, tf.nn.dropout() multiplies all units that are kept (i.e. not dropped) by 1/keep_prob.

Batch Normalization Batch normalization is another method to regularize a convolutional network. On top of a regularizing effect, batch normalization also gives your convolutional network a resistance to vanishing gradient during training. This can decrease training time and result in better performance. https://www.kdnuggets.com/2018/09/dropout-convolutional-networks.html

```
In [33]: def LeNet(x):
             # Arguments used for tf.truncated_normal, randomly defines variables for the weight
             # mu and sigma define how we initialize our weights, they can be adjusted as additi
             mu = 0
             sigma = 0.1

             # SOLUTION: Layer 1: Convolutional. Input = 32x32x3. Output = 28x28x6.
             conv1_W = tf.Variable(tf.truncated_normal(shape=(5, 5, 3, 6), mean = mu, stddev = s
             conv1_b = tf.Variable(tf.zeros(6))
             conv1   = tf.nn.conv2d(x, conv1_W, strides=[1, 1, 1, 1], padding='VALID') + conv1_b

             # SOLUTION: Activation.
             conv1 = tf.nn.relu(conv1)

             # SOLUTION: Pooling. Input = 28x28x6. Output = 14x14x6.
             # Same result using padding = 'SAME' due to aymmetric matrix
             conv1 = tf.nn.max_pool(conv1, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='VA

             # SOLUTION: Layer 2: Convolutional. Output = 10x10x16.
             conv2_W = tf.Variable(tf.truncated_normal(shape=(5, 5, 6, 16), mean = mu, stddev =
             conv2_b = tf.Variable(tf.zeros(16))
             conv2   = tf.nn.conv2d(conv1, conv2_W, strides=[1, 1, 1, 1], padding='VALID') + con

             # SOLUTION: Activation.
             conv2 = tf.nn.relu(conv2)

             # SOLUTION: Pooling. Input = 10x10x16. Output = 5x5x16.
```

```
    # Same result using padding = 'SAME' due to aymmetric matrix
    conv2 = tf.nn.max_pool(conv2, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='VA

    # SOLUTION: Flatten. Input = 5x5x16. Output = 400.
    fc0   = flatten(conv2)

    # SOLUTION: Layer 3: Fully Connected. Input = 400. Output = 120.
    fc1_W = tf.Variable(tf.truncated_normal(shape=(400, 120), mean = mu, stddev = sigma
    fc1_b = tf.Variable(tf.zeros(120))
    fc1   = tf.matmul(fc0, fc1_W) + fc1_b

    # SOLUTION: Activation.
    fc1    = tf.nn.relu(fc1)
    fc1    = tf.nn.dropout(fc1, keep_prob)

    # SOLUTION: Layer 4: Fully Connected. Input = 120. Output = 84.
    fc2_W  = tf.Variable(tf.truncated_normal(shape=(120, 84), mean = mu, stddev = sigma
    fc2_b  = tf.Variable(tf.zeros(84))
    fc2    = tf.matmul(fc1, fc2_W) + fc2_b

    # SOLUTION: Activation.
    fc2    = tf.nn.relu(fc2)
    fc2    = tf.nn.dropout(fc2, keep_prob)

    # SOLUTION: Layer 5: Fully Connected. Input = 84. Output = 43.
    fc3_W  = tf.Variable(tf.truncated_normal(shape=(84, 43), mean = mu, stddev = sigma)
    fc3_b  = tf.Variable(tf.zeros(43))
    logits = tf.matmul(fc2, fc3_W) + fc3_b

    return logits, conv1, conv2
```

Formulas Convolution Parameters

1. For Valid Padding
   - out_height = ceil(float(in_height - filter_height + 1) / float(strides))
   - out_width = ceil(float(in_width - filter_width + 1) / float(strides))
2. For Same Padding
   - out_height = ceil(float(in_height / float(strides))
   - out_width = ceil(float(in_width / float(strides))

ceil = rounding up

In [34]:
```
# SOLUTION: Layer 1: Convolutional. Input = 32x32x3. Output = 28x28x6.
Input = [32,32,3]
Output = [28,28,6]
#Formula
#out_height_width = (float(in_height_width - filter_height_width + 1) / float(strides))
```

```
#Output[0] = (Input[0] - filter_height_width + 1)/1
filter_height_width = Input[0]-Output[0]+1

print(filter_height_width)
```

```
5
```

```
In [35]: # SOLUTION: Layer 2: Convolutional. Input = 32x32x6. Output = 28x28x16.
         Input = [14,14,6]
         Output = [10,10,16]

         filter_height_width = Input[0]-Output[0]+1

         print(filter_height_width)
```

```
5
```

1.4.4 4.4 Loss function, Optimizer & Accuracy evaluation

4.4.1 Cross entropy loss function

- Cross entropy minimises difference of softmax generated probabilities (logits) to one hot encoded labels
- rf.reduce_mean averages the difference from logits to ground truth labels

```
In [36]: logits, conv1, conv2 = LeNet(x)
         cross_entropy = tf.nn.softmax_cross_entropy_with_logits(labels=one_hot_y, logits=logits
         loss_operation = tf.reduce_mean(cross_entropy)
```

4.4.2 Learning rate

- For Optimizer (i.e. Gradient Descent, Adam Optimizer) to update weights and bias during training. New weights and bias deduct learning rate * derivative of weights and bias.
- Learning rate defines how quickly the network updates its weights with 0.001 being a good default value
- Stay calm and decrease your learning rate for better accuracy

```
In [37]: rate = 0.001
```

4.4.3 Adam Optimizer

- AdamOptimizer is more sophisticated than stochastic gradient descent and a good default optimizer Adaptive Gradient Algorithm (AdaGrad) that maintains a per-parameter learning rate that improves performance on problems with sparse gradients (e.g. natural language and computer vision problems). The method computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients.
- optimizer.minimze uses back propagation to update the network and minimize the training loss

```
In [38]: optimizer = tf.train.AdamOptimizer(learning_rate = rate)
         training_operation = optimizer.minimize(loss_operation)
```

4.4.4 Accuracy evaluation

- tf.argmax(logits, 1) outputs the correct label, which is the label with the max probability across the logits
- tf.argmax(one_hot_y, 1) outputs the actual true values
- tf.equal compares the two tensors and returns a list of booleans [True, False, True...] for all the predictions
- Convert (cast) the list of booleans into a list of binary value [0,1,0,..] and calculate the accuracy mean of each batch

```
In [39]: correct_prediction = tf.equal(tf.argmax(logits, 1), tf.argmax(one_hot_y, 1))
         accuracy_operation = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
```

1.4.5 4.5 Train, Validate and Test the Model

Training: - loss_operation = tf.reduce_mean(cross_entropy) - optimizer = tf.train.AdamOptimizer(learning_rate = rate) - training_operation = optimizer.minimize(loss_operation) - sess.run(training_operation, feed_dict={x: X_train, y: y_train})

Validation: - correct_prediction = tf.equal(tf.argmax(logits, 1), tf.argmax(one_hot_y, 1)) - accuracy_operation = tf.reduce_mean(tf.cast(correct_prediction, tf.float32)) - accuracy = sess.run(accuracy_operation, feed_dict={x: X_validation, y: y_validation})

Testing: - Same as Validation (6.2) with the only difference feeding testing dataset into the session - correct_prediction = tf.equal(tf.argmax(logits, 1), tf.argmax(one_hot_y, 1)) - accuracy_operation = tf.reduce_mean(tf.cast(correct_prediction, tf.float32)) - accuracy = sess.run(accuracy_operation, feed_dict={x: X_test, y: y_test})

Conclusion: Hence validation and testing runs the same code (def evaluate accuracy operation) just on different datasets, wereby training runs training_operation on training dataset, there is no accuracy in training, only training itself

```
In [40]: #keep prob required to implement dropout regularization


         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
                 total_accuracy += (accuracy * len(batch_x))
             return total_accuracy / num_examples
```

4.5.1 Train and validate the model

Initialize values & Run session Initializing weights and biases using tf.truncan, tf.random or tf.zeros

- tf.zeros is only for simplicity as it doesnt provide any randomness and is hence not a great choice

- tf.random: tf. truncated_normal() selects random numbers from a normal distribution whose mean is close to 0 and values are close to 0. For example, from -0.1 to 0.1
- tf.truncated: the generated values follow a normal distribution with specified mean and standard deviation, except that values whose magnitude is more than 2 standard deviations from the mean are dropped and re-picked. It's called truncated because your cutting off the tails from a normal distribution

```
In [41]: saver = tf.train.Saver()
```

```
In [42]: #Create lists for visualisation
         batches = (range(0, EPOCHS))
         loss_batch = []
         train_acc_batch = []
         valid_acc_batch = []
```

Training and Validation Operation - training operation feeds in its training data - after the training validation accuracy is calculated based on validation data - keep prob required to implement dropout regularization

```
In [43]: with tf.Session() as sess:
             sess.run(tf.global_variables_initializer())

             print("Training...")
             print()

             for i in range(EPOCHS):
                 X_train_norm, y_train = shuffle(X_train_norm, y_train)
                 for offset in range(0, n_train, BATCH_SIZE):
                     batch_x, batch_y = X_train_norm[offset:offset + BATCH_SIZE], y_train[offset
                     sess.run(training_operation, feed_dict={x: batch_x, y: batch_y, keep_prob:

                 loss = sess.run(loss_operation, feed_dict={x: batch_x, y: batch_y, keep_prob: 0
                 loss_batch.append(loss)

                 training_accuracy = evaluate(X_train_norm, y_train)
                 train_acc_batch.append(training_accuracy)
                 print("EPOCH {} ...".format(i+1))
                 print("Training Accuracy = {:.3f}".format(training_accuracy))
                 print()

                 validation_accuracy = evaluate(X_valid_norm, y_valid)
                 valid_acc_batch.append(validation_accuracy)
                 print("Validation Accuracy = {:.3f}".format(validation_accuracy))
                 print()

             saver.save(sess, './lenet')

             print("Model saved")
```

```
Training...

EPOCH 1 ...
Training Accuracy = 0.575

Validation Accuracy = 0.512

EPOCH 2 ...
Training Accuracy = 0.815

Validation Accuracy = 0.750

EPOCH 3 ...
Training Accuracy = 0.888

Validation Accuracy = 0.826

EPOCH 4 ...
Training Accuracy = 0.935

Validation Accuracy = 0.882

EPOCH 5 ...
Training Accuracy = 0.955

Validation Accuracy = 0.899

EPOCH 6 ...
Training Accuracy = 0.963

Validation Accuracy = 0.903

EPOCH 7 ...
Training Accuracy = 0.973

Validation Accuracy = 0.920

EPOCH 8 ...
Training Accuracy = 0.979

Validation Accuracy = 0.921

EPOCH 9 ...
Training Accuracy = 0.982

Validation Accuracy = 0.937

EPOCH 10 ...
```

```
Training Accuracy = 0.982

Validation Accuracy = 0.941

EPOCH 11 ...
Training Accuracy = 0.987

Validation Accuracy = 0.946

EPOCH 12 ...
Training Accuracy = 0.987

Validation Accuracy = 0.944

EPOCH 13 ...
Training Accuracy = 0.989

Validation Accuracy = 0.946

EPOCH 14 ...
Training Accuracy = 0.991

Validation Accuracy = 0.944

EPOCH 15 ...
Training Accuracy = 0.992

Validation Accuracy = 0.949

EPOCH 16 ...
Training Accuracy = 0.991

Validation Accuracy = 0.947

EPOCH 17 ...
Training Accuracy = 0.992

Validation Accuracy = 0.948

EPOCH 18 ...
Training Accuracy = 0.993

Validation Accuracy = 0.955

EPOCH 19 ...
Training Accuracy = 0.994

Validation Accuracy = 0.951
```

```
EPOCH 20 ...
Training Accuracy = 0.994

Validation Accuracy = 0.955

EPOCH 21 ...
Training Accuracy = 0.996

Validation Accuracy = 0.958

EPOCH 22 ...
Training Accuracy = 0.995

Validation Accuracy = 0.952

EPOCH 23 ...
Training Accuracy = 0.996

Validation Accuracy = 0.953

EPOCH 24 ...
Training Accuracy = 0.996

Validation Accuracy = 0.952

EPOCH 25 ...
Training Accuracy = 0.997

Validation Accuracy = 0.959

EPOCH 26 ...
Training Accuracy = 0.996

Validation Accuracy = 0.962

EPOCH 27 ...
Training Accuracy = 0.997

Validation Accuracy = 0.951

EPOCH 28 ...
Training Accuracy = 0.997

Validation Accuracy = 0.959

EPOCH 29 ...
Training Accuracy = 0.998
```

```
Validation Accuracy = 0.958

EPOCH 30 ...
Training Accuracy = 0.997

Validation Accuracy = 0.954

EPOCH 31 ...
Training Accuracy = 0.997

Validation Accuracy = 0.954

EPOCH 32 ...
Training Accuracy = 0.998

Validation Accuracy = 0.956

EPOCH 33 ...
Training Accuracy = 0.998

Validation Accuracy = 0.959

EPOCH 34 ...
Training Accuracy = 0.998

Validation Accuracy = 0.958

EPOCH 35 ...
Training Accuracy = 0.998

Validation Accuracy = 0.962

EPOCH 36 ...
Training Accuracy = 0.997

Validation Accuracy = 0.958

EPOCH 37 ...
Training Accuracy = 0.997

Validation Accuracy = 0.957

EPOCH 38 ...
Training Accuracy = 0.998

Validation Accuracy = 0.957
```

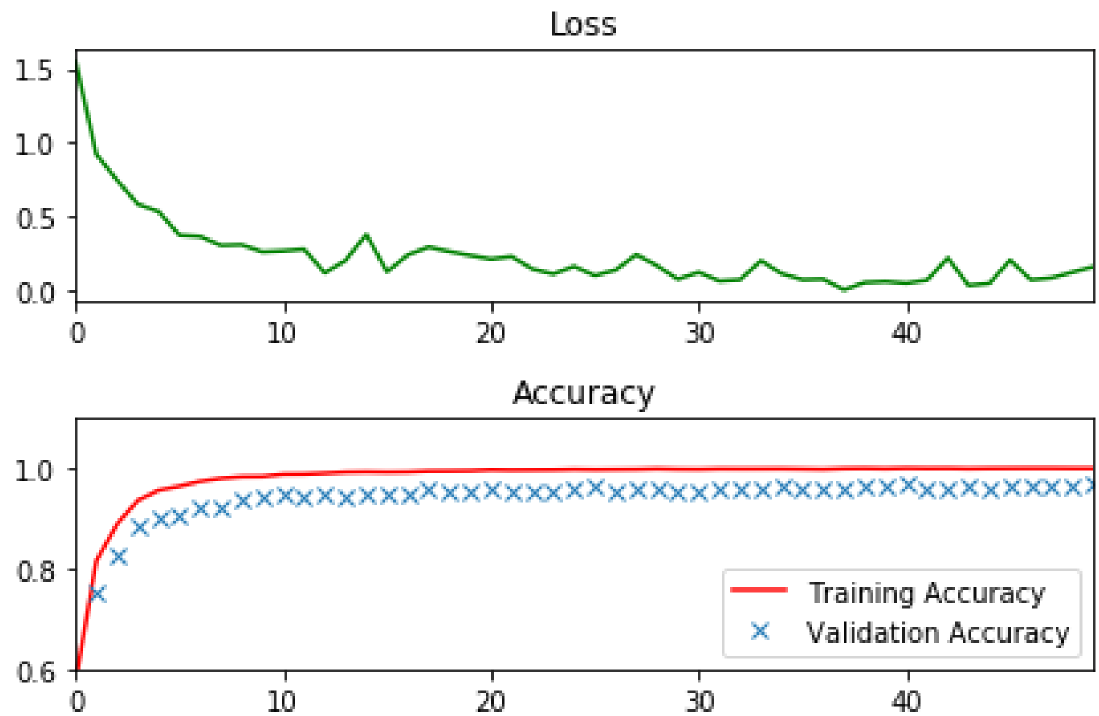
```
EPOCH 39 ...
Training Accuracy = 0.999

Validation Accuracy = 0.962

EPOCH 40 ...
Training Accuracy = 0.998

Validation Accuracy = 0.965

EPOCH 41 ...
Training Accuracy = 0.999

Validation Accuracy = 0.966

EPOCH 42 ...
Training Accuracy = 0.999

Validation Accuracy = 0.957

EPOCH 43 ...
Training Accuracy = 0.999

Validation Accuracy = 0.959

EPOCH 44 ...
Training Accuracy = 0.998

Validation Accuracy = 0.961

EPOCH 45 ...
Training Accuracy = 0.999

Validation Accuracy = 0.958

EPOCH 46 ...
Training Accuracy = 0.999

Validation Accuracy = 0.961

EPOCH 47 ...
Training Accuracy = 0.999

Validation Accuracy = 0.961

EPOCH 48 ...
Training Accuracy = 0.999
```

```
Validation Accuracy = 0.962

EPOCH 49 ...
Training Accuracy = 0.999

Validation Accuracy = 0.961

EPOCH 50 ...
Training Accuracy = 0.999

Validation Accuracy = 0.967

Model saved
```

1.4.6 4.6 Interpretation of accuracy results in training and validation

Visualisation of loss and accuracy over training epochs

```
In [44]: loss_plot = plt.subplot(211)
         loss_plot.set_title('Loss')
         loss_plot.plot(batches, loss_batch, 'g')
         loss_plot.set_xlim([batches[0], batches[-1]])

         acc_plot = plt.subplot(212)
         acc_plot.set_title('Accuracy')
         acc_plot.plot(batches, train_acc_batch, 'r', label='Training Accuracy')
         acc_plot.plot(batches, valid_acc_batch, 'x', label='Validation Accuracy')
         acc_plot.set_ylim([0.6, 1.1])
         acc_plot.set_xlim([batches[0], batches[-1]])
         acc_plot.legend(loc=4)

         plt.tight_layout()
         plt.show()
```



A validation set can be used to assess how well the model is performing. A low accuracy on the training and validation sets imply underfitting. A high accuracy on the training set but low accuracy on the validation set implies overfitting.

Conclusion: Hence the model seems to be slightly overfitting as validation accuracy is always lower than training accuracy. To avoid overfitting images can as an example be converted to grayscale images.

1.4.7 4.7 Final Model Evaluation on Test dataset

Once you are completely satisfied with your model, evaluate the performance of the model on the test set. Be sure to only do this once! If you were to measure the performance of your trained model on the test set, then improve your model, and then measure the performance of your model on the test set again, that would invalidate your test results. You wouldn't get a true measure of how well your model would perform against real data.

```
In [45]: #test_accuracy feeds in its own testing data for Testing

         with tf.Session() as sess:
             saver.restore(sess, tf.train.latest_checkpoint('.'))
             test_accuracy = evaluate(X_test_norm, y_test)
             print("Test Accuracy = {:.3f}".format(test_accuracy))
```

```
INFO:tensorflow:Restoring parameters from ./lenet
Test Accuracy = 0.950
```

1.5 5. Test the model on new images

To give yourself more insight into how your model is working, download at least five pictures of German traffic signs from the web and use your model to predict the traffic sign type.

You may find `signnames.csv` useful as it contains mappings from the class id (integer) to the actual sign name.

```
In [46]: import matplotlib.pyplot as plt
         import matplotlib.image as mpimg
         import cv2
         import numpy as np
         import tensorflow as tf
```

1.5.1 5.1 Load and Output the New Images

```
In [47]: Trafficsign1 = mpimg.imread('Trafficsign1.jpg')
         Trafficsign2 = mpimg.imread('Trafficsign2.jpg')
         Trafficsign3 = mpimg.imread('Trafficsign3.jpg')
         Trafficsign4 = mpimg.imread('Trafficsign4.jpg')
         Trafficsign5 = mpimg.imread('Trafficsign5.jpg')
```

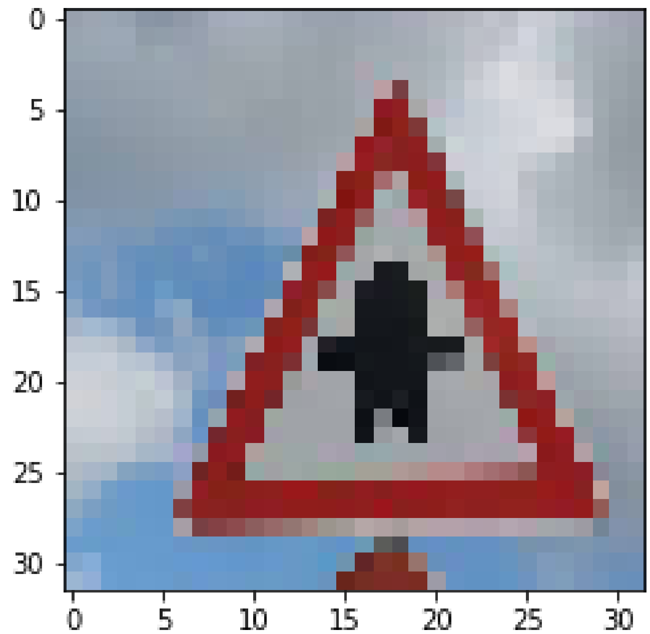
```
In [48]: f, (ax1, ax2, ax3, ax4, ax5) = plt.subplots(1, 5, figsize=(20,10))
         ax1.imshow(Trafficsign1)
         ax2.imshow(Trafficsign2)
         ax3.imshow(Trafficsign3)
         ax4.imshow(Trafficsign4)
         ax5.imshow(Trafficsign5)
```

```
Out[48]: <matplotlib.image.AxesImage at 0x7f0534084eb8>
```


```
In [49]: Trafficsign1 = cv2.resize(Trafficsign1, (32,32))
         plt.imshow(Trafficsign1)
         Trafficsign1
```

```
Out[49]: array([[[152, 160, 173],
                 [157, 165, 178],
                 [154, 166, 180],
                 ...,
                 [171, 173, 188],
                 [164, 166, 181],
```

```
 [160, 162, 177]],

[[153, 164, 176],
 [156, 167, 179],
 [156, 168, 182],
 ...,
 [165, 172, 182],
 [163, 170, 180],
 [156, 163, 173]],

[[151, 165, 176],
 [152, 166, 177],
 [154, 166, 180],
 ...,
 [162, 171, 178],
 [159, 168, 175],
 [155, 164, 171]],

...,
[[106, 155, 204],
 [103, 157, 201],
 [102, 154, 204],
 ...,
 [119, 147, 176],
 [124, 145, 174],
 [123, 144, 173]],

[[129, 166, 204],
 [133, 173, 216],
 [105, 157, 207],
 ...,
 [114, 144, 179],
 [111, 142, 178],
 [112, 143, 180]],

[[155, 182, 210],
 [145, 174, 215],
 [107, 159, 209],
 ...,
 [114, 148, 186],
 [105, 144, 186],
 [104, 143, 186]]], dtype=uint8)
```



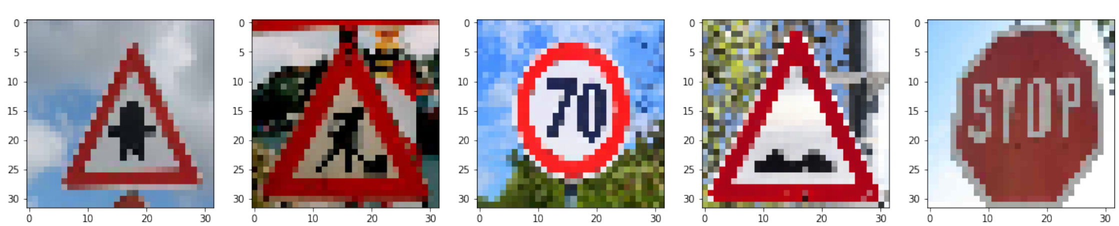
1.5.2 5.2 Preprocess new images

5.2.1 Resize images to 32x32 pixels

In [50]:
```
Trafficsign2 = cv2.resize(Trafficsign2, (32,32))
Trafficsign3 = cv2.resize(Trafficsign3, (32,32))
Trafficsign4 = cv2.resize(Trafficsign4, (32,32))
Trafficsign5 = cv2.resize(Trafficsign5, (32,32))
```

In [51]:
```
f, (ax1, ax2, ax3, ax4, ax5) = plt.subplots(1, 5, figsize=(20,10))
ax1.imshow(Trafficsign1)
ax2.imshow(Trafficsign2)
ax3.imshow(Trafficsign3)
ax4.imshow(Trafficsign4)
ax5.imshow(Trafficsign5)
```

Out[51]: <matplotlib.image.AxesImage at 0x7f052c1a7198>



5.2.2 Create a list of 32x32 normalised new images

In [52]:
```
new_images = []
new_images.append(Trafficsign1)
new_images.append(Trafficsign2)
new_images.append(Trafficsign3)
new_images.append(Trafficsign4)
new_images.append(Trafficsign5)
new_images
```

Out[52]:
```
[array([[[152, 160, 173],
         [157, 165, 178],
         [154, 166, 180],
         ...,
         [171, 173, 188],
         [164, 166, 181],
         [160, 162, 177]],

        [[153, 164, 176],
         [156, 167, 179],
         [156, 168, 182],
         ...,
         [165, 172, 182],
         [163, 170, 180],
         [156, 163, 173]],

        [[151, 165, 176],
         [152, 166, 177],
         [154, 166, 180],
         ...,
         [162, 171, 178],
         [159, 168, 175],
         [155, 164, 171]],

        ...,
        [[106, 155, 204],
         [103, 157, 201],
         [102, 154, 204],
         ...,
         [119, 147, 176],
         [124, 145, 174],
         [123, 144, 173]],

        [[129, 166, 204],
         [133, 173, 216],
         [105, 157, 207],
```

```
 ...,
 [114, 144, 179],
 [111, 142, 178],
 [112, 143, 180]],

[[155, 182, 210],
 [145, 174, 215],
 [107, 159, 209],
 ...,
 [114, 148, 186],
 [105, 144, 186],
 [104, 143, 186]]], dtype=uint8), array([[[157,  19,  13],
 [152,   5,   3],
 [160,   1,   3],
 ...,
 [179,  47,  51],
 [186,  84,  80],
 [177, 102,  89]],

[[154,   9,   2],
 [160,   5,   0],
 [173,   0,   0],
 ...,
 [224, 211, 210],
 [229, 213, 215],
 [228, 214, 217]],

[[157, 150, 141],
 [153, 146, 137],
 [172, 162, 166],
 ...,
 [231, 220, 224],
 [231, 220, 222],
 [232, 220, 220]],

...,
[[ 84,  64,  21],
 [131, 106,  75],
 [110,   3,   0],
 ...,
 [152,   0,   0],
 [ 66,  75,  66],
 [ 59,  93,  94]],

[[ 69,  63,   3],
 [117,  90,  43],
 [ 50,  11,   0],
 ...,
```

```
 [ 76,  10,   4],
 [ 72,  95,  93],
 [ 57,  91,  92]],

[[ 62,  50,   1],
 [ 79,  71,  11],
 [118,  94,  38],
 ...,
 [  1,   5,   6],
 [ 58,  90,  87],
 [ 61,  93,  92]]], dtype=uint8), array([[[ 80, 143, 255, 255],
 [144, 186, 247, 255],
 [164, 196, 251, 255],
 ...,
 [129, 182, 251, 255],
 [137, 186, 255, 255],
 [118, 183, 255, 255]],

[[107, 161, 248, 255],
 [144, 188, 243, 255],
 [130, 180, 246, 255],
 ...,
 [153, 199, 255, 255],
 [129, 185, 253, 255],
 [ 94, 158, 247, 255]],

[[105, 165, 245, 255],
 [123, 174, 251, 255],
 [113, 173, 252, 255],
 ...,
 [134, 186, 255, 255],
 [112, 173, 255, 255],
 [ 75, 133, 211, 255]],

...,
[[154, 195, 246, 255],
 [136, 189, 246, 255],
 [123, 190, 243, 255],
 ...,
 [ 80,  64,  21, 255],
 [ 76,  88,  28, 255],
 [122, 144,  77, 255]],

[[ 96, 122,  72, 255],
 [103, 142, 130, 255],
 [126, 191, 246, 255],
 ...,
 [ 66,  94,  13, 255],
```

```
 [ 56,  80,   9, 255],
 [ 71,  87,  19, 255]],

[[ 63,  85,  39, 255],
 [ 63,  81,  39, 255],
 [ 71,  88,  35, 255],
 ...,
 [ 57,  75,   0, 255],
 [127, 144,  60, 255],
 [ 60,  71,  16, 255]]], dtype=uint8), array([[[168, 177,  88],
 [186, 199, 156],
 [120, 129, 112],
 ...,
 [242, 245, 249],
 [249, 251, 253],
 [250, 251, 255]],

[[185, 192, 127],
 [114, 122,  59],
 [158, 174,  41],
 ...,
 [238, 241, 248],
 [249, 250, 254],
 [248, 250, 254]],

[[136, 148, 153],
 [131, 140,  90],
 [ 50,  48,  26],
 ...,
 [238, 240, 247],
 [242, 246, 250],
 [252, 252, 255]],

...,
[[196, 179,  80],
 [193,   8,  16],
 [173,   5,  22],
 ...,
 [173,   4,  24],
 [220, 211, 215],
 [162, 159, 157]],

[[165, 183, 128],
 [236, 236, 241],
 [186,   3,  15],
 ...,
 [172,   4,  23],
 [227, 225, 230],
```

```
 [155, 154, 155]],

[[168, 181,  78],
 [142, 155,  52],
 [155, 161,  73],
 ...,
 [199, 197, 199],
 [ 50,  50,  53],
 [245, 248, 252]]], dtype=uint8), array([[[152, 204, 254],
 [155, 207, 255],
 [162, 209, 253],
 ...,
 [ 67,  85,  66],
 [106, 125,  97],
 [183, 200, 181]],

[[155, 207, 255],
 [158, 211, 253],
 [161, 214, 255],
 ...,
 [105, 124, 131],
 [208, 228, 226],
 [171, 189, 202]],

[[158, 211, 253],
 [158, 211, 253],
 [165, 212, 255],
 ...,
 [176, 200, 205],
 [237, 255, 255],
 [ 41,  48,  38]],

...,
[[234, 247, 255],
 [243, 247, 255],
 [243, 248, 252],
 ...,
 [255, 255, 255],
 [255, 255, 255],
 [255, 255, 255]],

[[239, 243, 255],
 [243, 248, 252],
 [243, 248, 252],
 ...,
 [255, 255, 255],
 [255, 255, 255],
 [255, 255, 255]],
```

```
[[254, 254, 254],
 [253, 253, 253],
 [253, 253, 253],
 ...,
 [255, 255, 255],
 [255, 255, 255],
 [255, 255, 255]]], dtype=uint8)]
```

5.2.3 Normalise pixel values of new images

```
In [53]: X_new_norm = []

         for i in range(0,5):
             for j in range(0,32):
                 for k in range(0,32):
                     for l in range(0,3):
                         X_new_norm.append((new_images[i][j][k][l]-128)/128)

         X_new_norm  = np.asarray(X_new_norm)
         X_new_norm  = X_new_norm.reshape(5, 32, 32, 3)
         print(X_new_norm.shape)
         print(X_new_norm.dtype)
         X_new_norm
```

```
(5, 32, 32, 3)
float64
```

```
Out[53]: array([[[[ 0.1875   ,  0.25     ,  0.3515625],
                  [ 0.2265625,  0.2890625,  0.390625 ],
                  [ 0.203125 ,  0.296875 ,  0.40625  ],
                  ...,
                  [ 0.3359375,  0.3515625,  0.46875  ],
                  [ 0.28125  ,  0.296875 ,  0.4140625],
                  [ 0.25     ,  0.265625 ,  0.3828125]],

                 [[ 0.1953125,  0.28125  ,  0.375    ],
                  [ 0.21875  ,  0.3046875,  0.3984375],
                  [ 0.21875  ,  0.3125   ,  0.421875 ],
                  ...,
                  [ 0.2890625,  0.34375  ,  0.421875 ],
                  [ 0.2734375,  0.328125 ,  0.40625  ],
                  [ 0.21875  ,  0.2734375,  0.3515625]],

                 [[ 0.1796875,  0.2890625,  0.375    ],
                  [ 0.1875   ,  0.296875 ,  0.3828125],
                  [ 0.203125 ,  0.296875 ,  0.40625  ],
                  ...,
```

```
  [ 0.265625 ,  0.3359375,  0.390625 ],
  [ 0.2421875,  0.3125   ,  0.3671875],
  [ 0.2109375,  0.28125  ,  0.3359375]],

 ...,
 [[-0.171875 ,  0.2109375,  0.59375  ],
  [-0.1953125,  0.2265625,  0.5703125],
  [-0.203125 ,  0.203125 ,  0.59375  ],
  ...,
  [-0.0703125,  0.1484375,  0.375    ],
  [-0.03125  ,  0.1328125,  0.359375 ],
  [-0.0390625,  0.125    ,  0.3515625]],

 [[ 0.0078125,  0.296875 ,  0.59375  ],
  [ 0.0390625,  0.3515625,  0.6875   ],
  [-0.1796875,  0.2265625,  0.6171875],
  ...,
  [-0.109375 ,  0.125    ,  0.3984375],
  [-0.1328125,  0.109375 ,  0.390625 ],
  [-0.125    ,  0.1171875,  0.40625  ]],

 [[ 0.2109375,  0.421875 ,  0.640625 ],
  [ 0.1328125,  0.359375 ,  0.6796875],
  [-0.1640625,  0.2421875,  0.6328125],
  ...,
  [-0.109375 ,  0.15625  ,  0.453125 ],
  [-0.1796875,  0.125    ,  0.453125 ],
  [-0.1875   ,  0.1171875,  0.453125 ]]],


[[[ 0.2265625, -0.8515625, -0.8984375],
  [ 0.1875   , -0.9609375, -0.9765625],
  [ 0.25     , -0.9921875, -0.9765625],
  ...,
  [ 0.3984375, -0.6328125, -0.6015625],
  [ 0.453125 , -0.34375  , -0.375    ],
  [ 0.3828125, -0.203125 , -0.3046875]],

 [[ 0.203125 , -0.9296875, -0.984375 ],
  [ 0.25     , -0.9609375, -1.       ],
  [ 0.3515625, -1.       , -1.       ],
  ...,
  [ 0.75     ,  0.6484375,  0.640625 ],
  [ 0.7890625,  0.6640625,  0.6796875],
  [ 0.78125  ,  0.671875 ,  0.6953125]],

 [[ 0.2265625,  0.171875 ,  0.1015625],
  [ 0.1953125,  0.140625 ,  0.0703125],
```

```
  [ 0.34375  ,  0.265625 ,  0.296875 ],
  ...,
  [ 0.8046875,  0.71875  ,  0.75     ],
  [ 0.8046875,  0.71875  ,  0.734375 ],
  [ 0.8125   ,  0.71875  ,  0.71875  ]],

 ...,
 [[-0.34375  , -0.5      , -0.8359375],
  [ 0.0234375, -0.171875 , -0.4140625],
  [-0.140625 , -0.9765625, -1.       ],
  ...,
  [ 0.1875   , -1.       , -1.       ],
  [-0.484375 , -0.4140625, -0.484375 ],
  [-0.5390625, -0.2734375, -0.265625 ]],

 [[-0.4609375, -0.5078125, -0.9765625],
  [-0.0859375, -0.296875 , -0.6640625],
  [-0.609375 , -0.9140625, -1.       ],
  ...,
  [-0.40625  , -0.921875 , -0.96875  ],
  [-0.4375   , -0.2578125, -0.2734375],
  [-0.5546875, -0.2890625, -0.28125  ]],

 [[-0.515625 , -0.609375 , -0.9921875],
  [-0.3828125, -0.4453125, -0.9140625],
  [-0.078125 , -0.265625 , -0.703125 ],
  ...,
  [-0.9921875, -0.9609375, -0.953125 ],
  [-0.546875 , -0.296875 , -0.3203125],
  [-0.5234375, -0.2734375, -0.28125  ]]],


[[[-0.375    ,  0.1171875,  0.9921875],
  [ 0.125    ,  0.453125 ,  0.9296875],
  [ 0.28125  ,  0.53125  ,  0.9609375],
  ...,
  [ 0.0078125,  0.421875 ,  0.9609375],
  [ 0.0703125,  0.453125 ,  0.9921875],
  [-0.078125 ,  0.4296875,  0.9921875]],

 [[-0.1640625,  0.2578125,  0.9375   ],
  [ 0.125    ,  0.46875  ,  0.8984375],
  [ 0.015625 ,  0.40625  ,  0.921875 ],
  ...,
  [ 0.1953125,  0.5546875,  0.9921875],
  [ 0.0078125,  0.4453125,  0.9765625],
  [-0.265625 ,  0.234375 ,  0.9296875]],
```

```
 [[-0.1796875,  0.2890625,  0.9140625],
  [-0.0390625,  0.359375 ,  0.9609375],
  [-0.1171875,  0.3515625,  0.96875  ],
  ...,
  [ 0.046875 ,  0.453125 ,  0.9921875],
  [-0.125    ,  0.3515625,  0.9921875],
  [-0.4140625,  0.0390625,  0.6484375]],

 ...,
 [[ 0.203125 ,  0.5234375,  0.921875 ],
  [ 0.0625   ,  0.4765625,  0.921875 ],
  [-0.0390625,  0.484375 ,  0.8984375],
  ...,
  [-0.375    , -0.5      , -0.8359375],
  [-0.40625  , -0.3125   , -0.78125  ],
  [-0.046875 ,  0.125    , -0.3984375]],

 [[-0.25     , -0.046875 , -0.4375   ],
  [-0.1953125,  0.109375 ,  0.015625 ],
  [-0.015625 ,  0.4921875,  0.921875 ],
  ...,
  [-0.484375 , -0.265625 , -0.8984375],
  [-0.5625   , -0.375    , -0.9296875],
  [-0.4453125, -0.3203125, -0.8515625]],

 [[-0.5078125, -0.3359375, -0.6953125],
  [-0.5078125, -0.3671875, -0.6953125],
  [-0.4453125, -0.3125   , -0.7265625],
  ...,
  [-0.5546875, -0.4140625, -1.       ],
  [-0.0078125,  0.125    , -0.53125  ],
  [-0.53125  , -0.4453125, -0.875    ]]],


[[[ 0.3125   ,  0.3828125, -0.3125   ],
  [ 0.453125 ,  0.5546875,  0.21875  ],
  [-0.0625   ,  0.0078125, -0.125    ],
  ...,
  [ 0.890625 ,  0.9140625,  0.9453125],
  [ 0.9453125,  0.9609375,  0.9765625],
  [ 0.953125 ,  0.9609375,  0.9921875]],

 [[ 0.4453125,  0.5      , -0.0078125],
  [-0.109375 , -0.046875 , -0.5390625],
  [ 0.234375 ,  0.359375 , -0.6796875],
  ...,
  [ 0.859375 ,  0.8828125,  0.9375   ],
  [ 0.9453125,  0.953125 ,  0.984375 ],
```

```
  [ 0.9375   ,  0.953125 ,  0.984375 ]],

 [[ 0.0625   ,  0.15625  ,  0.1953125],
  [ 0.0234375,  0.09375  , -0.296875 ],
  [-0.609375 , -0.625    , -0.796875 ],
  ...,
  [ 0.859375 ,  0.875    ,  0.9296875],
  [ 0.890625 ,  0.921875 ,  0.953125 ],
  [ 0.96875  ,  0.96875  ,  0.9921875]],

 ...,
 [[ 0.53125  ,  0.3984375, -0.375    ],
  [ 0.5078125, -0.9375   , -0.875    ],
  [ 0.3515625, -0.9609375, -0.828125 ],
  ...,
  [ 0.3515625, -0.96875  , -0.8125   ],
  [ 0.71875  ,  0.6484375,  0.6796875],
  [ 0.265625 ,  0.2421875,  0.2265625]],

 [[ 0.2890625,  0.4296875,  0.       ],
  [ 0.84375  ,  0.84375  ,  0.8828125],
  [ 0.453125 , -0.9765625, -0.8828125],
  ...,
  [ 0.34375  , -0.96875  , -0.8203125],
  [ 0.7734375,  0.7578125,  0.796875 ],
  [ 0.2109375,  0.203125 ,  0.2109375]],

 [[ 0.3125   ,  0.4140625, -0.390625 ],
  [ 0.109375 ,  0.2109375, -0.59375  ],
  [ 0.2109375,  0.2578125, -0.4296875],
  ...,
  [ 0.5546875,  0.5390625,  0.5546875],
  [-0.609375 , -0.609375 , -0.5859375],
  [ 0.9140625,  0.9375   ,  0.96875  ]]],


[[[ 0.1875   ,  0.59375  ,  0.984375 ],
  [ 0.2109375,  0.6171875,  0.9921875],
  [ 0.265625 ,  0.6328125,  0.9765625],
  ...,
  [-0.4765625, -0.3359375, -0.484375 ],
  [-0.171875 , -0.0234375, -0.2421875],
  [ 0.4296875,  0.5625   ,  0.4140625]],

 [[ 0.2109375,  0.6171875,  0.9921875],
  [ 0.234375 ,  0.6484375,  0.9765625],
  [ 0.2578125,  0.671875 ,  0.9921875],
  ...,
```

```
         [-0.1796875, -0.03125  ,  0.0234375],
         [ 0.625    ,  0.78125  ,  0.765625 ],
         [ 0.3359375,  0.4765625,  0.578125 ]],

        [[ 0.234375 ,  0.6484375,  0.9765625],
         [ 0.234375 ,  0.6484375,  0.9765625],
         [ 0.2890625,  0.65625  ,  0.9921875],
         ...,
         [ 0.375    ,  0.5625   ,  0.6015625],
         [ 0.8515625,  0.9921875,  0.9921875],
         [-0.6796875, -0.625    , -0.703125 ]],

        ...,
        [[ 0.828125 ,  0.9296875,  0.9921875],
         [ 0.8984375,  0.9296875,  0.9921875],
         [ 0.8984375,  0.9375   ,  0.96875  ],
         ...,
         [ 0.9921875,  0.9921875,  0.9921875],
         [ 0.9921875,  0.9921875,  0.9921875],
         [ 0.9921875,  0.9921875,  0.9921875]],

        [[ 0.8671875,  0.8984375,  0.9921875],
         [ 0.8984375,  0.9375   ,  0.96875  ],
         [ 0.8984375,  0.9375   ,  0.96875  ],
         ...,
         [ 0.9921875,  0.9921875,  0.9921875],
         [ 0.9921875,  0.9921875,  0.9921875],
         [ 0.9921875,  0.9921875,  0.9921875]],

        [[ 0.984375 ,  0.984375 ,  0.984375 ],
         [ 0.9765625,  0.9765625,  0.9765625],
         [ 0.9765625,  0.9765625,  0.9765625],
         ...,
         [ 0.9921875,  0.9921875,  0.9921875],
         [ 0.9921875,  0.9921875,  0.9921875],
         [ 0.9921875,  0.9921875,  0.9921875]]]])
```

In [54]:
```
print(X_new_norm.min())
print(X_new_norm.max())
```

```
-1.0
0.9921875
```

5.2.4 Create a list of corresponding traffic sign names

In [55]:
```
y_new = [11,25,4,22,14]
y_new = np.array(y_new, dtype=np.uint8)
y_new
```

Out[55]: array([11, 25, 4, 22, 14], dtype=uint8)

1.5.3 5.3 Classify new images with trained model and calculate total accuracy and cross entropy

5.3.1 Total accuracy

```
In [56]: with tf.Session() as sess:
             saver.restore(sess, tf.train.latest_checkpoint('.'))
             new_images_accuracy = evaluate(X_new_norm, y_new)
             print("New Images Accuracy = {:.3f}".format(new_images_accuracy))
```

```
INFO:tensorflow:Restoring parameters from ./lenet
New Images Accuracy = 0.800
```

5.3.2 Cross entropy loss

```
In [57]: with tf.Session() as sess:
             saver.restore(sess, tf.train.latest_checkpoint('.'))
             cross_entropy_new = sess.run(cross_entropy, feed_dict={x: X_new_norm, y: y_new, kee
             cross_entropy_average = sess.run(loss_operation, feed_dict={x: X_new_norm, y: y_new
             print(cross_entropy_new)
             print(cross_entropy_average)
```

```
INFO:tensorflow:Restoring parameters from ./lenet
[  0.00000000e+00   1.97274055e+01   0.00000000e+00   0.00000000e+00
   8.04588199e-03]
3.94709
```

5.3.3 Conclusion - 4 out of 5 images have been correctly classified with a total accuracy of 80% - Image 1 and 4 have been classified with absolute confidence since cross entropy, the difference between logits and one hot encoding is 0 - Image 3 and 4 have minor differences / lower confidence - Image 2 has the highest loss and represents close to the total loss since the averafe 3.24/5 = 0.648

Hence, as seen in section below, image 2 is the image that has been worngly classified.

1.6 6. Output Top 5 Softmax Probabilities for each image loaded from the Web

For each of the new images, print out the model's softmax probabilities to show the **certainty** of the model's predictions (limit the output to the top 5 probabilities for each image). `tf.nn.top_k` could prove helpful here.

`tf.nn.top_k` will return the values and indices (class ids) of the top k predictions. So if k=3, for each sign, it'll return the 3 largest probabilities (out of a possible 43) and the correspoding class ids.

Note: A Softmax Regression returns a list of values between 0 and 1 that add up to one.

1.6.1 6.1 Softmax logits

- Below are shown the probabilities, which range from 0..1 aligned to the softmax function.
- The decimals show a high confidence for the Top 1 softmax probability of 99%+

```
In [58]: softmax_logits = tf.nn.softmax(logits)

         with tf.Session() as sess:
             saver.restore(sess, "./lenet")
             softmax_logits_new = sess.run(softmax_logits, feed_dict={x: X_new_norm, y: y_new, k
             print(softmax_logits_new)
```

```
INFO:tensorflow:Restoring parameters from ./lenet
[[  0.00000000e+00   1.03929504e-32   0.00000000e+00   0.00000000e+00
    0.00000000e+00   0.00000000e+00   1.01508177e-37   0.00000000e+00
    0.00000000e+00   0.00000000e+00   0.00000000e+00   1.00000000e+00
    1.06079151e-26   0.00000000e+00   0.00000000e+00   0.00000000e+00
    3.62193388e-34   7.74801568e-28   5.53799520e-21   4.90293505e-34
    2.83991230e-31   4.49066428e-19   0.00000000e+00   7.65068799e-34
    1.14507692e-29   3.64816511e-29   3.20913065e-23   2.73875525e-15
    1.28975496e-29   0.00000000e+00   3.82318222e-12   0.00000000e+00
    2.10071115e-36   0.00000000e+00   0.00000000e+00   0.00000000e+00
    0.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00
    0.00000000e+00   8.88518435e-34   3.48705746e-29]
 [  9.06011702e-13   8.34834572e-15   3.63043903e-29   8.11129315e-25
    7.97391154e-25   7.98790846e-27   2.38987637e-18   2.05635780e-23
    1.28120579e-26   1.28412892e-23   4.20809234e-21   1.58053279e-01
    5.41562834e-13   6.38430492e-26   7.81339256e-17   2.69961013e-23
    3.69031037e-15   2.34471976e-15   4.60524546e-07   2.20003865e-13
    7.07182438e-11   4.11900930e-10   2.72485494e-23   5.80511955e-15
    7.96684344e-06   2.70705436e-09   1.79470270e-08   8.41914833e-01
    2.39577275e-07   4.56469973e-16   2.31592148e-05   1.21399186e-21
    1.51613205e-13   1.40361100e-28   2.79821238e-23   1.52305804e-22
    2.92482765e-22   2.84255961e-23   1.67389104e-26   5.83216857e-33
    2.29800925e-23   2.18876702e-13   2.90044860e-16]
 [  6.12853140e-20   1.10348451e-22   1.09360303e-37   0.00000000e+00
    1.00000000e+00   2.18975746e-22   0.00000000e+00   1.76992432e-25
    3.19021441e-14   0.00000000e+00   0.00000000e+00   0.00000000e+00
    0.00000000e+00   0.00000000e+00   0.00000000e+00   1.00580554e-34
    9.73262611e-27   1.25909732e-25   2.93246068e-26   0.00000000e+00
    0.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00
    0.00000000e+00   0.00000000e+00   3.46067225e-37   0.00000000e+00
    0.00000000e+00   9.65013923e-36   0.00000000e+00   0.00000000e+00
    0.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00
    0.00000000e+00   3.75940875e-28   0.00000000e+00   5.58854164e-30
    2.89563351e-26   0.00000000e+00   0.00000000e+00]
 [  3.85961953e-34   0.00000000e+00   0.00000000e+00   0.00000000e+00
    0.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00
    6.23744011e-38   0.00000000e+00   0.00000000e+00   0.00000000e+00
```

```
   0.00000000e+00   2.53691978e-29   0.00000000e+00   9.64236688e-30
   0.00000000e+00   5.03496323e-36   1.23325500e-35   0.00000000e+00
   1.76585156e-32   0.00000000e+00   1.00000000e+00   0.00000000e+00
   2.77818596e-28   1.43586379e-23   6.35763781e-19   0.00000000e+00
   1.47950834e-27   1.53058117e-08   2.10263539e-36   2.06211028e-34
   0.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00
   0.00000000e+00   0.00000000e+00   0.00000000e+00   8.00689200e-31
   0.00000000e+00   0.00000000e+00   0.00000000e+00]
 [ 5.71084250e-14   5.91132520e-15   1.19226799e-14   8.00494604e-09
   2.99283992e-12   8.35723201e-13   2.08518742e-30   1.31154074e-20
   1.41986547e-08   7.61083214e-08   2.39487929e-10   3.60816881e-19
   4.69641570e-08   6.49597379e-04   9.91986394e-01   7.36280158e-03
   8.62546430e-24   3.09250381e-09   1.80563705e-18   3.62553215e-23
   1.21292340e-13   7.53412020e-24   8.02917498e-07   1.56161088e-19
   7.53384239e-14   1.06493052e-11   5.28797237e-08   1.40139827e-22
   7.96127487e-13   1.88231979e-07   1.49034966e-19   1.47407553e-21
   4.35560268e-12   1.97522022e-20   1.25886616e-17   7.79097856e-15
   1.08451518e-17   9.92659682e-17   4.25349284e-10   1.46708405e-12
   3.52112647e-22   1.23041056e-21   8.36860232e-25]]
```

In [59]:
```
softmax_logits = tf.nn.softmax(logits)

with tf.Session() as sess:
    saver.restore(sess, "./lenet")
    softmax_logits_new = sess.run(softmax_logits, feed_dict={x: X_new_norm, y: y_new, k

    for i in range(0,5):
        for logit in range (0,43):
            print(i,logit, "{:.10f}".format(softmax_logits_new[i][logit]))
```

```
INFO:tensorflow:Restoring parameters from ./lenet
0 0 0.0000000000
0 1 0.0000000000
0 2 0.0000000000
0 3 0.0000000000
0 4 0.0000000000
0 5 0.0000000000
0 6 0.0000000000
0 7 0.0000000000
0 8 0.0000000000
0 9 0.0000000000
0 10 0.0000000000
0 11 1.0000000000
0 12 0.0000000000
0 13 0.0000000000
0 14 0.0000000000
0 15 0.0000000000
```

```
0 16 0.0000000000
0 17 0.0000000000
0 18 0.0000000000
0 19 0.0000000000
0 20 0.0000000000
0 21 0.0000000000
0 22 0.0000000000
0 23 0.0000000000
0 24 0.0000000000
0 25 0.0000000000
0 26 0.0000000000
0 27 0.0000000000
0 28 0.0000000000
0 29 0.0000000000
0 30 0.0000000000
0 31 0.0000000000
0 32 0.0000000000
0 33 0.0000000000
0 34 0.0000000000
0 35 0.0000000000
0 36 0.0000000000
0 37 0.0000000000
0 38 0.0000000000
0 39 0.0000000000
0 40 0.0000000000
0 41 0.0000000000
0 42 0.0000000000
1 0 0.0000000000
1 1 0.0000000000
1 2 0.0000000000
1 3 0.0000000000
1 4 0.0000000000
1 5 0.0000000000
1 6 0.0000000000
1 7 0.0000000000
1 8 0.0000000000
1 9 0.0000000000
1 10 0.0000000000
1 11 0.1580532789
1 12 0.0000000000
1 13 0.0000000000
1 14 0.0000000000
1 15 0.0000000000
1 16 0.0000000000
1 17 0.0000000000
1 18 0.0000004605
1 19 0.0000000000
1 20 0.0000000001
```

```
1 21 0.0000000004
1 22 0.0000000000
1 23 0.0000000000
1 24 0.0000079668
1 25 0.0000000027
1 26 0.0000000179
1 27 0.8419148326
1 28 0.0000002396
1 29 0.0000000000
1 30 0.0000231592
1 31 0.0000000000
1 32 0.0000000000
1 33 0.0000000000
1 34 0.0000000000
1 35 0.0000000000
1 36 0.0000000000
1 37 0.0000000000
1 38 0.0000000000
1 39 0.0000000000
1 40 0.0000000000
1 41 0.0000000000
1 42 0.0000000000
2 0 0.0000000000
2 1 0.0000000000
2 2 0.0000000000
2 3 0.0000000000
2 4 1.0000000000
2 5 0.0000000000
2 6 0.0000000000
2 7 0.0000000000
2 8 0.0000000000
2 9 0.0000000000
2 10 0.0000000000
2 11 0.0000000000
2 12 0.0000000000
2 13 0.0000000000
2 14 0.0000000000
2 15 0.0000000000
2 16 0.0000000000
2 17 0.0000000000
2 18 0.0000000000
2 19 0.0000000000
2 20 0.0000000000
2 21 0.0000000000
2 22 0.0000000000
2 23 0.0000000000
2 24 0.0000000000
2 25 0.0000000000
```

```
2 26 0.0000000000
2 27 0.0000000000
2 28 0.0000000000
2 29 0.0000000000
2 30 0.0000000000
2 31 0.0000000000
2 32 0.0000000000
2 33 0.0000000000
2 34 0.0000000000
2 35 0.0000000000
2 36 0.0000000000
2 37 0.0000000000
2 38 0.0000000000
2 39 0.0000000000
2 40 0.0000000000
2 41 0.0000000000
2 42 0.0000000000
3 0 0.0000000000
3 1 0.0000000000
3 2 0.0000000000
3 3 0.0000000000
3 4 0.0000000000
3 5 0.0000000000
3 6 0.0000000000
3 7 0.0000000000
3 8 0.0000000000
3 9 0.0000000000
3 10 0.0000000000
3 11 0.0000000000
3 12 0.0000000000
3 13 0.0000000000
3 14 0.0000000000
3 15 0.0000000000
3 16 0.0000000000
3 17 0.0000000000
3 18 0.0000000000
3 19 0.0000000000
3 20 0.0000000000
3 21 0.0000000000
3 22 1.0000000000
3 23 0.0000000000
3 24 0.0000000000
3 25 0.0000000000
3 26 0.0000000000
3 27 0.0000000000
3 28 0.0000000000
3 29 0.0000000153
3 30 0.0000000000
```

3 31 0.0000000000
3 32 0.0000000000
3 33 0.0000000000
3 34 0.0000000000
3 35 0.0000000000
3 36 0.0000000000
3 37 0.0000000000
3 38 0.0000000000
3 39 0.0000000000
3 40 0.0000000000
3 41 0.0000000000
3 42 0.0000000000
4 0 0.0000000000
4 1 0.0000000000
4 2 0.0000000000
4 3 0.0000000080
4 4 0.0000000000
4 5 0.0000000000
4 6 0.0000000000
4 7 0.0000000000
4 8 0.0000000142
4 9 0.0000000761
4 10 0.0000000002
4 11 0.0000000000
4 12 0.0000000470
4 13 0.0006495974
4 14 0.9919863939
4 15 0.0073628016
4 16 0.0000000000
4 17 0.0000000031
4 18 0.0000000000
4 19 0.0000000000
4 20 0.0000000000
4 21 0.0000000000
4 22 0.0000008029
4 23 0.0000000000
4 24 0.0000000000
4 25 0.0000000000
4 26 0.0000000529
4 27 0.0000000000
4 28 0.0000000000
4 29 0.0000001882
4 30 0.0000000000
4 31 0.0000000000
4 32 0.0000000000
4 33 0.0000000000
4 34 0.0000000000
4 35 0.0000000000

```
4 36 0.0000000000
4 37 0.0000000000
4 38 0.0000000004
4 39 0.0000000000
4 40 0.0000000000
4 41 0.0000000000
4 42 0.0000000000
```

1.6.2 6.2 Top 3 Softmax propabilites

In [60]:
```
softmax_logits = tf.nn.softmax(logits)
top_3 = tf.nn.top_k(softmax_logits, k=3)

with tf.Session() as sess:
    saver.restore(sess, "./lenet")
    sess.run(softmax_logits, feed_dict={x: X_new_norm, y: y_new, keep_prob: 1.0})
    softmax_top_3 = sess.run(top_3, feed_dict={x: X_new_norm, y: y_new, keep_prob: 1.0}
    print('Top 3 softmax probabilites')
    print(softmax_top_3)
```

```
INFO:tensorflow:Restoring parameters from ./lenet
Top 3 softmax probabilites
TopKV2(values=array([[  1.00000000e+00,   3.82318222e-12,   2.73875525e-15],
       [  8.41914833e-01,   1.58053279e-01,   2.31592148e-05],
       [  1.00000000e+00,   3.19021441e-14,   6.12853140e-20],
       [  1.00000000e+00,   1.53058117e-08,   6.35763781e-19],
       [  9.91986394e-01,   7.36280158e-03,   6.49597379e-04]], dtype=float32), indices=array([[
       [27, 11, 30],
       [ 4,  8,  0],
       [22, 29, 26],
       [14, 15, 13]], dtype=int32))
```

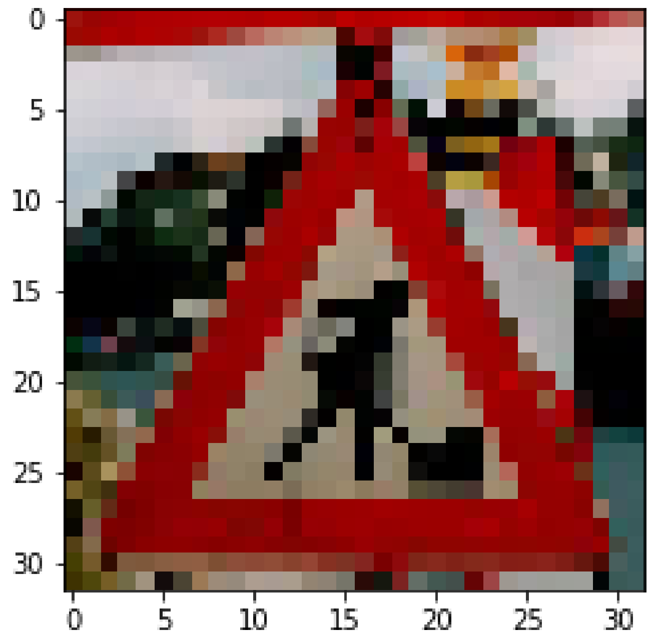
In [61]:
```
y_new
```

Out[61]:
```
array([11, 25,  4, 22, 14], dtype=uint8)
```

- 11 - Right-of-way at the next intersection
- 25 - Road work
- 4 - Speed limit (70km/h)
- 22 - Bumpy road
- 14 - Stop

In [62]:
```
plt.imshow(new_images[1])
```

Out[62]:
```
<matplotlib.image.AxesImage at 0x7f0547fcfef0>
```



The top 3 softmax probabilities show that the roadwork image has not been correctly classified in neither of the top 3 classes. It has shown a high confidence for Right-of-way at the next intersection. This is expected due to the unclear pixel due to image cropping during data preprocessing.

[27,11,30] - 27: Pedestrians - 11: Right-of-way at the next intersection - 30: Beware of ice/snow

---

1.7 7. Visualize the Neural Network's State with Test Images

While neural networks can be a great learning device they are often referred to as a black box. We can understand what the weights of a neural network look like better by plotting their feature maps. After successfully training your neural network you can see what it's feature maps look like by plotting the output of the network's weight layers in response to a test stimuli image. From these plotted feature maps, it's possible to see what characteristics of an image the network finds interesting. For a sign, maybe the inner network feature maps react with high activation to the sign's boundary outline or to the contrast in the sign's painted symbol.

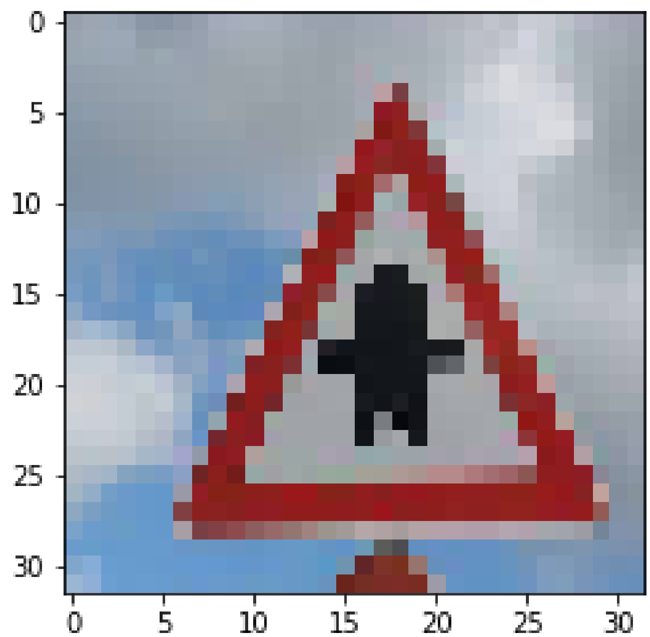
For an example of what feature map outputs look like, check out NVIDIA's results in their paper End-to-End Deep Learning for Self-Driving Cars in the section Visualization of internal CNN State. NVIDIA was able to show that their network's inner weights had high activations to road boundary lines by comparing feature maps from an image with a clear path to one without. Try experimenting with a similar test to show that your trained network's weights are looking for interesting features, whether it's looking at differences in feature maps from images with or without a sign, or even what feature maps look like in a trained network vs a completely untrained one on the same sign image.

1.7.1 7.1 Image Input

- image_input: the test image being fed into the network to produce the feature maps
- Make sure to preprocess your image_input in a way your network expects with size, normalization, ect if needed

```
In [63]: Trafficsign1
         plt.imshow(Trafficsign1)
```

```
Out[63]: <matplotlib.image.AxesImage at 0x7f0547fabef0>
```




```
In [64]: image_featuremap = X_new_norm[0]
         print(image_featuremap.shape)
         image_featuremap.dtype
```

```
(32, 32, 3)
```

```
Out[64]: dtype('float64')
```

```
In [65]: image_featuremap = image_featuremap.astype('float32')
         print(image_featuremap.dtype)
```

```
float32
```

```
In [66]: image_featuremap_lenet = np.expand_dims(image_featuremap,0)
         image_featuremap_lenet.shape
```

```
Out[66]: (1, 32, 32, 3)
```

1.7.2 7.2 Define function to access weights from LeNet

tf_activation - tf_activation: should be a tf variable name used during your training procedure that represents the calculated state of a specific weight layer - activation_min/max: can be used to view the activation contrast in more detail, by default matplot sets min and max to the actual min and max values of the output - If you get an error tf_activation is not defined it may be having trouble accessing the variable from inside a function

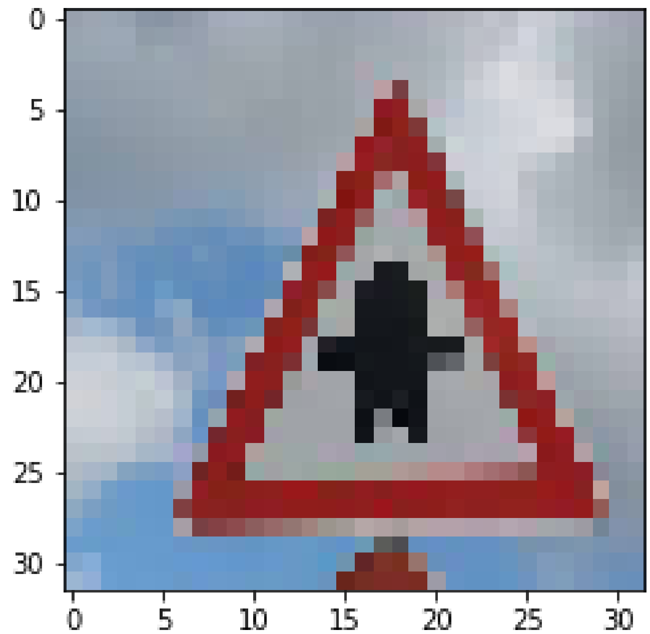
```
In [69]: def outputFeatureMap(image_input, tf_activation, activation_min=-1, activation_max=-1 ,

             activation = tf_activation.eval(session=sess,feed_dict={x : image_input})
             featuremaps = activation.shape[3]
             plt.figure(plt_num, figsize=(15,15))
             for featuremap in range(featuremaps):
                 plt.subplot(6,8, featuremap+1) # sets the number of feature maps to show on eac
                 plt.title('FeatureMap ' + str(featuremap)) # displays the feature map number
                 if activation_min != -1 & activation_max != -1:
                     plt.imshow(activation[0,:,:, featuremap], interpolation="nearest", vmin =ac
                 elif activation_max != -1:
                     plt.imshow(activation[0,:,:, featuremap], interpolation="nearest", vmax=act
                 elif activation_min !=-1:
                     plt.imshow(activation[0,:,:, featuremap], interpolation="nearest", vmin=act
                 else:
                     plt.imshow(activation[0,:,:, featuremap], interpolation="nearest", cmap='gr
```

1.7.3 7.3 Visualisation of convolutional layers

- First convolutional layer has k_output = 6 feature maps
- Second convolutional layer has k_output = 16 feature maps

```
In [70]: # plt_num: used to plot out multiple different weight feature map sets on the same bloc
         logits, conv1, conv2 = LeNet(image_featuremap_lenet)

         with tf.Session() as sess:
             sess.run(tf.global_variables_initializer())
             saver.restore(sess, tf.train.latest_checkpoint('.'))
             plt.imshow(Trafficsign1)
             plt.show()
             print('First and second convolutional layer')
             outputFeatureMap(image_featuremap_lenet, conv1, plt_num=1)
             outputFeatureMap(image_featuremap_lenet, conv2, plt_num=2)
```

```
INFO:tensorflow:Restoring parameters from ./lenet
```



First and second convolutional layer