Traffic Sign Recognition

Writeup Template

You can use this file as a template for your writeup if you want to submit it as a markdown file, but feel free to use some other method and submit a pdf if you prefer.

Build a Traffic Sign Recognition Project

The goals / steps of this project are the following:

- Load the data set (see below for links to the project data set)
- Explore, summarize and visualize the data set
- Design, train and test a model architecture
- Use the model to make predictions on new images
- Analyze the softmax probabilities of the new images
- Summarize the results with a written report

Rubric Points

Here I will consider the rubric points (https://review.udacity.com/#!/rubrics/481/view) individually and describe how I addressed each point in my implementation.

Writeup / README

1. Provide a Writeup / README that includes all the rubric points and how you addressed each one. You can submit your writeup as markdown or pdf. You can use this template as a guide for writing the report. The submission includes the project code.

You're reading it! and here is a link to my project code (https://github.com/udacity/CarND-Traffic-Sign-Classifier-Project/blob/master/Traffic_Sign_Classifier.ipynb)

Data Set Summary & Exploration

1. Provide a basic summary of the data set. In the code, the analysis should be done using python, numpy and/or pandas methods rather than hardcoding results manually.

- The size of training set is ?
  - 34799
- The size of the validation set is ?
  - 4410
- The size of test set is ?
  - 12630
- The shape of a traffic sign image is ?
  - (32, 32, 3)
- The number of unique classes/labels in the data set is ?
  - 43

2. Include an exploratory visualization of the dataset.

Here is an exploratory visualization of the data set.

0:Speed limit (20km/h)

1:Speed limit (30km/h)

2:Speed limit (50km/h)

3:Speed limit (60km/h)

4:Speed limit (70km/h)

5:Speed limit (80km/h)

6:End of speed limit (80km/h)

7:Speed limit (100km/h)

8:Speed limit (120km/h)

9:No passing

10:No passing for vehicles over 3.5 metric tons

11:Right-of-way at the next intersection

12:Priority road

13:Yield

14:Stop

15:No vehicles

16:Vehicles over 3.5 metric tons prohibited

17:No entry

18:General caution

19:Dangerous curve to the left

20:Dangerous curve to the right

21:Double curve

22:Bumpy road

23:Slippery road

24:Road narrows on the right

25:Road work

26:Traffic signals

27:Pedestrians

28:Children crossing

29:Bicycles crossing

30:Beware of ice/snow

31:Wild animals crossing

32:End of all speed and passing limits

33:Turn right ahead

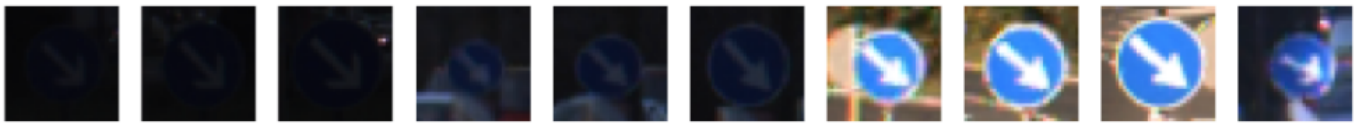
34:Turn left ahead

35:Ahead only

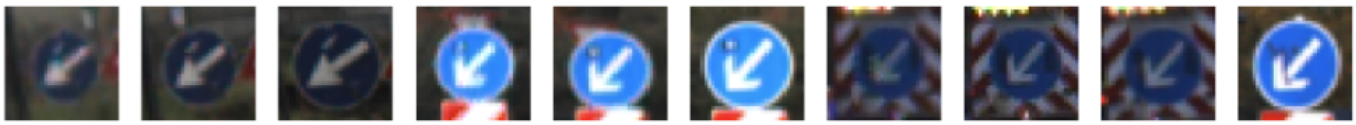
36:Go straight or right

37:Go straight or left

38:Keep right



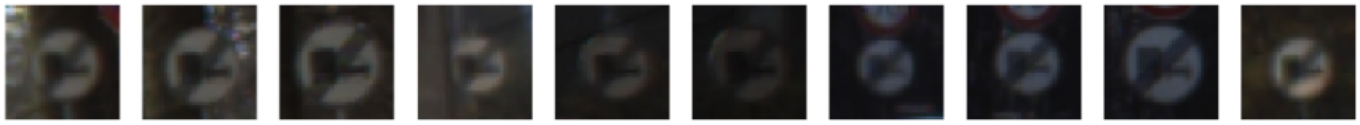
39:Keep left



40:Roundabout mandatory

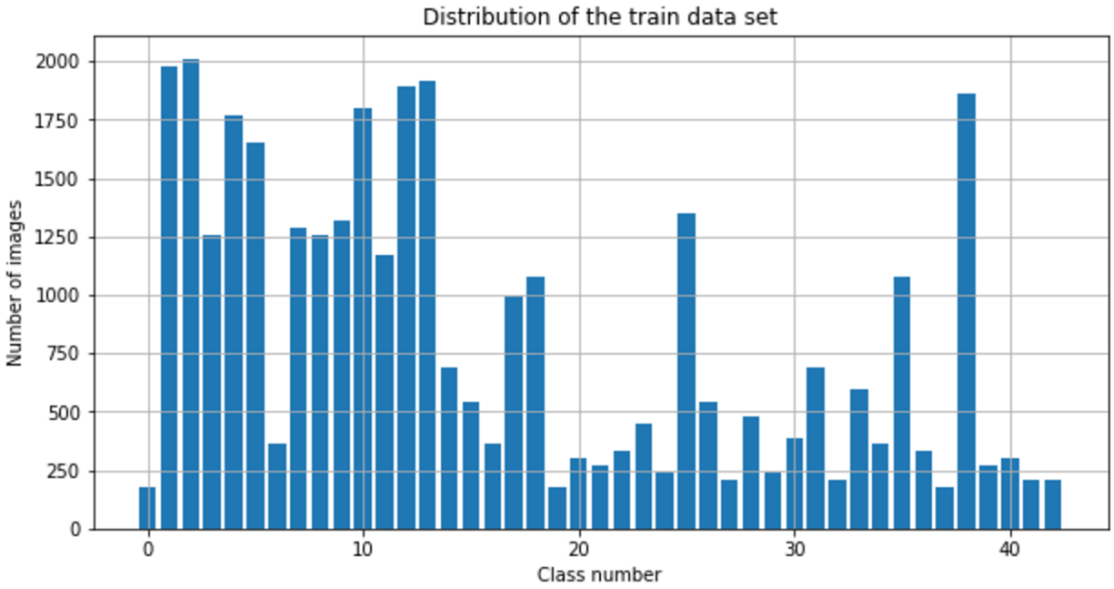
41:End of no passing

42:End of no passing by vehicles over 3.5 metric tons



As you can see the pictures above, there are big differences of contrast and brightness within the data set of the same label. So we need normalization to reduce the differences.

The following bar chart shows how the data is distributed.



- Min number of images per class = 180
- Max number of images per class = 2010
- Max number of images per class = 809.279069767

It turned out that there is a big discrepancy of number of images per label. We can think of add fake images to the training data set that the number is too small. We didn't implement that in this project.

Design and Test a Model Architecture

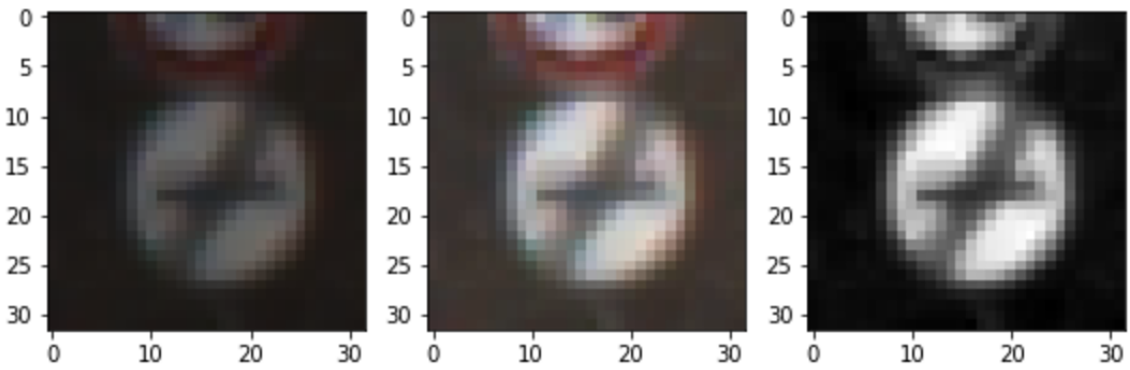
1. Describe how you preprocessed the image data.

What techniques were chosen and why did you choose these techniques? Consider including images showing the output of each preprocessing technique. Pre-processing refers to techniques such as converting to grayscale, normalization, etc. (OPTIONAL: As described in the "Stand Out Suggestions" part of the rubric, if you generated additional data for training, describe why you decided to generate additional data, how you generated the data, and provide example images of the additional data. Then describe the characteristics of the augmented training set like number of images in the set, number of images for each class, etc.)

I used two preprocessing techniques:

- normailization
  - Convert the pixel values to the range of (-1.0, 1.0) from (0 to 255).
  - X_train_norm = (X_train - 128.0) / 128.0
- grayscale
  - X_train_gray = np.sum(X_train_norm/3, axis=3, keepdims=True)

The following images show the results after normalization and then grayscale.



2. Describe what your final model architecture looks like including model type, layers, layer sizes, connectivity, etc.) Consider including a diagram and/or table describing the final model.

My final model consisted of the following layers:

Layer	Description
Input	32x32x1 Grayscale image
Convolution 5x5x6	1x1 stride, valid padding, outputs 28x28x6
RELU	
Max pooling	kernel=2x2, valid padding, outputs 14x14x6
Convolution 5x5x16	1x1 stride, valid padding, outputs 10x10x16
RELU	
Max pooling	kernel=2x2, valid padding, outputs 5x5x16
Fully connected	Output = 120 (400x120)
RELU	
DROPOUT	Keep_prob = 0.5
Fully connected	Output = 84 (120x84)
RELU	
DROPOUT	Keep_prob = 0.5
Fully connected	Outpout = 43 (84x43)
Softmax	

3. Describe how you trained your model. The discussion can include the type of optimizer, the batch size, number of epochs and any hyperparameters such as learning rate.

To train the model, I used the following hyper parameters.

- Optimizer: Adam Optimizer
- Cross Entropy : Softmax_cross_entropy_with_logits (from Tensorflow)
- Learning rate: 0.001
- Batch size: 128
- Epochs : 30

4. Describe the approach taken for finding a solution and getting the validation set accuracy to be at least 0.93.

Include in the discussion the results on the training, validation and test sets and where in the code these were calculated. Your approach may have been an iterative process, in which case, outline the steps you took to get to the final solution and why you chose those steps. Perhaps your solution involved an already well known implementation or architecture. In this case, discuss why you think the architecture is suitable for the current problem.

My final model results were:

- training set accuracy of ?
  - 0.996
- validation set accuracy of ?
  - 0.958
- test set accuracy of ?
  - 0.941

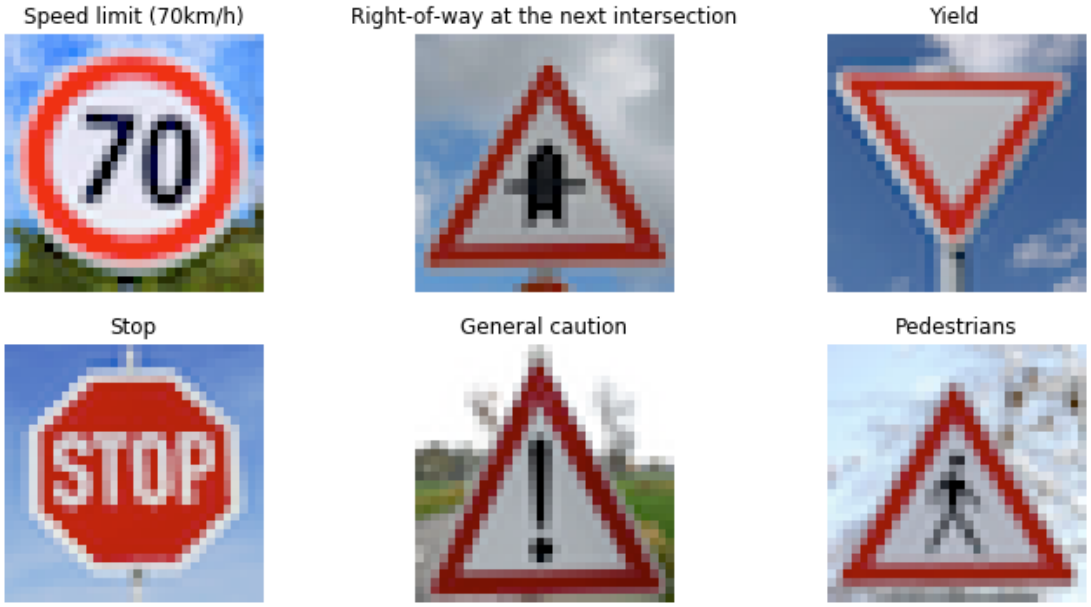
If an iterative approach was chosen:

- What was the first architecture that was tried and why was it chosen?
  - At first, we just reused the LeNet model without changing any architecture.
- What were some problems with the initial architecture?
  - The validation accuracy didn't reach to 0.93. (It was around below 0.9)
  - The model seems suffering from overfitting.
- How was the architecture adjusted and why was it adjusted? Typical adjustments could include choosing a different model architecture, adding or taking away layers (pooling, dropout, convolution, etc), using an activation function or changing the activation function. One common justification for adjusting an architecture would be due to overfitting or underfitting. A high accuracy on the training set but low accuracy on the validation set indicates over fitting; a low accuracy on both sets indicates under fitting.
  - Because the model is overfitting, I've applied DROPOUT for fully connected networks
- Which parameters were tuned? How were they adjusted and why?
  - Keep_prob for dropout : this was needed for fix the overfitting.
  - epochs : At first I started with 10, and the accuarcy was not reaching to 0.93. So I increased it to 30, and then I could get 0.958 of vaidation accuarcy.
- What are some of the important design choices and why were they chosen? For example, why might a convolution layer work well with this problem? How might a dropout layer help with creating a successful model?
  - Convolution Network, such as LeNet, has been shown a great performance in image classification.

Test a Model on New Images

1. Choose five German traffic signs found on the web and provide them in the report. For each image, discuss what quality or qualities might be difficult to classify.

Here are six German traffic signs that I found on the web:



2. Discuss the model's predictions on these new traffic signs and compare the results to predicting on the test set. At a minimum, discuss what the predictions were, the accuracy on these new predictions, and compare the accuracy to the accuracy on the test set (OPTIONAL: Discuss the results in more detail as described in the "Stand Out Suggestions" part of the rubric).

Here are the results of the prediction:

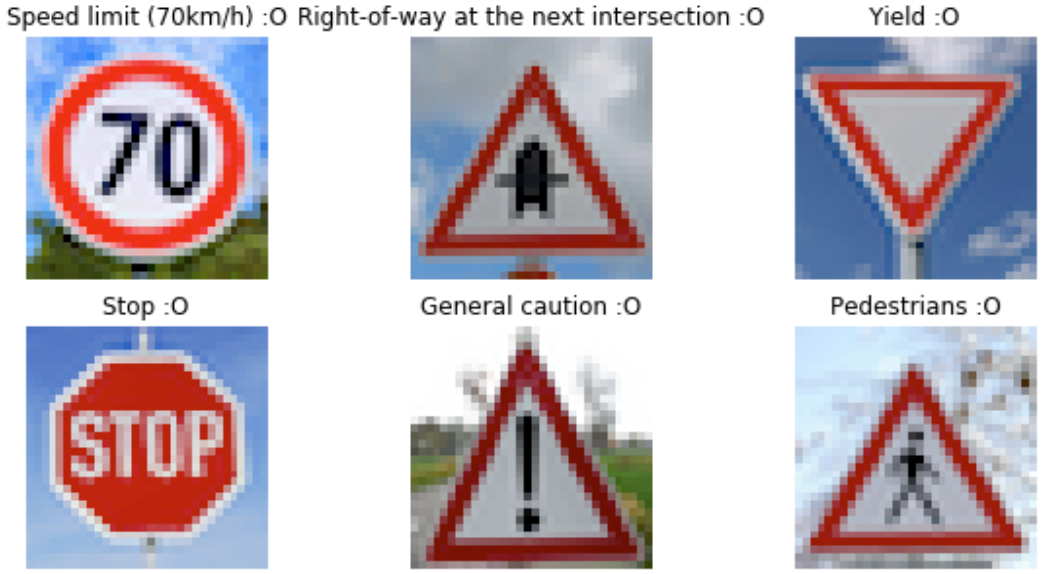


Image	Prediction

Speed limit(70km/h)	Speed limit(70km/h)
Right-of-way at the next intersection	Right-of-way at the next intersection
Yield	Yield
Stop	Stop
General Caution	General Caution
Pedestrians	Pedestrians

The model was able to correctly guess 6 of the 6 traffic signs, which gives an accuracy of 100%.

3. Describe how certain the model is when predicting on each of the six new images by looking at the softmax probabilities for each prediction. Provide the top 5 softmax probabilities for each image along with the sign type of each probability. (OPTIONAL: as described in the "Stand Out Suggestions" part of the rubric, visualizations can also be provided such as bar charts)

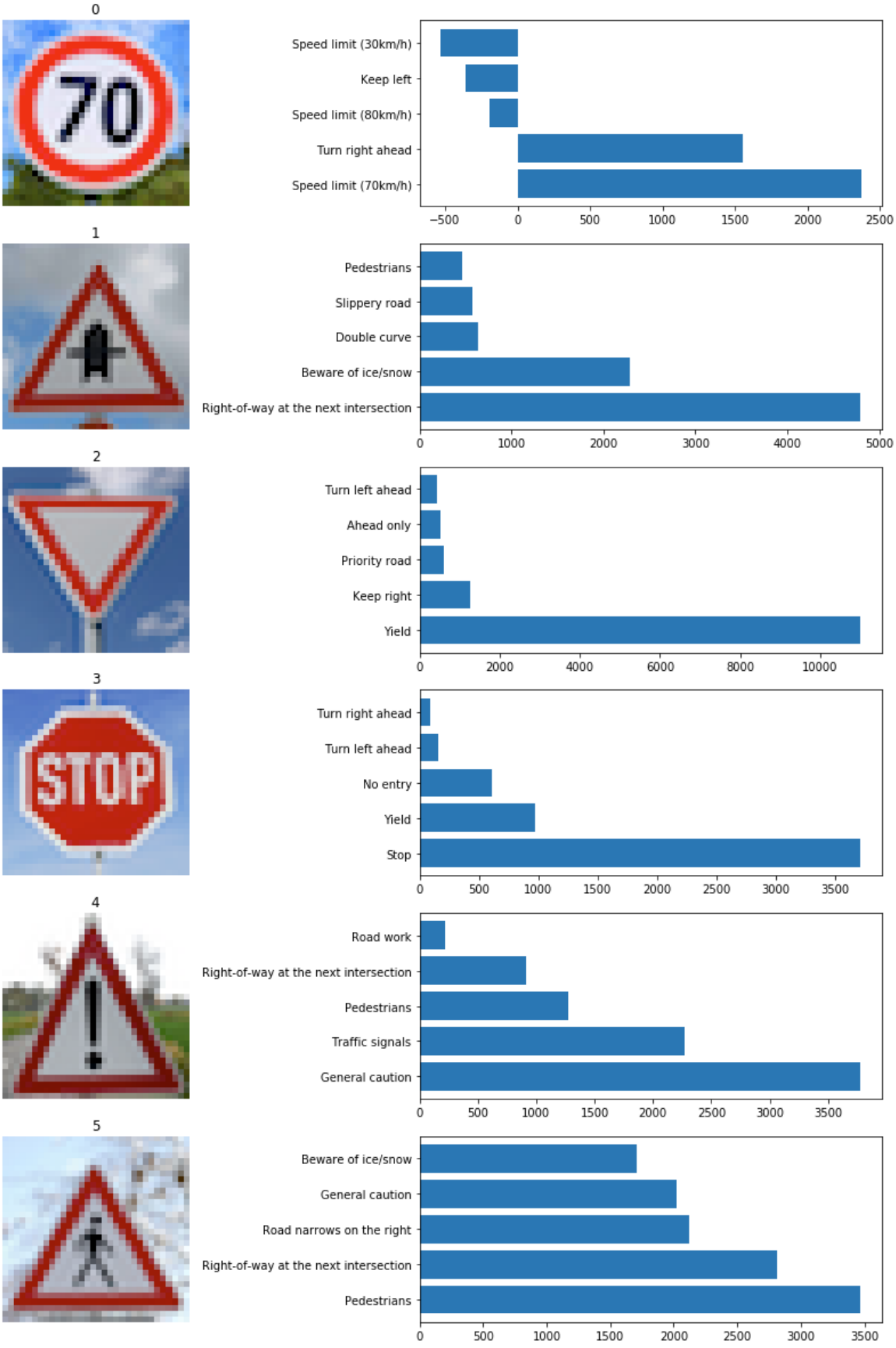
The code for making predictions on my final model is located in the Ipython notebook.

I tried to get softmax probabilities for top 5 candidates, but only the highest one got 1, and all others got 0 as probability. I think the reason of 0 as probability is because the logits values are too small or negative compared to the highest one.

```
TopKV2(values=array([[ 1.,  0.,  0.,  0.,  0.],
       [ 1.,  0.,  0.,  0.,  0.],
       [ 1.,  0.,  0.,  0.,  0.],
       [ 1.,  0.,  0.,  0.,  0.],
       [ 1.,  0.,  0.,  0.,  0.]], dtype=float32), indices=array([[ 4,  0,
 1,  2,  3],
       [11,  0,  1,  2,  3],
       [13,  0,  1,  2,  3],
       [14,  0,  1,  2,  3],
       [18,  0,  1,  2,  3],
       [27,  0,  1,  2,  3]], dtype=int32))
```

To see the original logits score differences among the top 5 candidates, I also printed the original logits values as below. With the logits values, I could get the right rankings of the candidates.

```
TopKV2(values=array([[  2371.92480469,   1551.98217773,   -192.00080872,   -
360.52252197,
         -531.40838623],
       [  4793.05810547,   2290.59521484,    636.65185547,    576.65856934,
          468.70645142],
       [ 11001.54199219,   1272.41125488,    614.79345703,    520.40283203,
          425.92300415],
       [  3716.22021484,    976.8616333 ,    605.56561279,    161.79014587,
           88.4176712 ],
       [  3775.68579102,   2274.43164062,   1277.84851074,    911.05566406,
          218.16653442],
       [  3469.00512695,   2815.8046875 ,   2122.03271484,   2019.87878418,
         1711.10058594]], dtype=float32), indices=array([[ 4, 33,  5, 39,
 1],
       [11, 30, 21, 23, 27],
       [13, 38, 12, 35, 34],
       [14, 13, 17, 34, 33],
       [18, 26, 27, 11, 25],
       [27, 11, 24, 18, 30]], dtype=int32))
```



(Optional) Visualizing the Neural Network (See Step 4 of the Ipython notebook for more details)

1. Discuss the visual output of your trained network's feature maps. What characteristics did the neural network use to make classifications?

In [ ]: