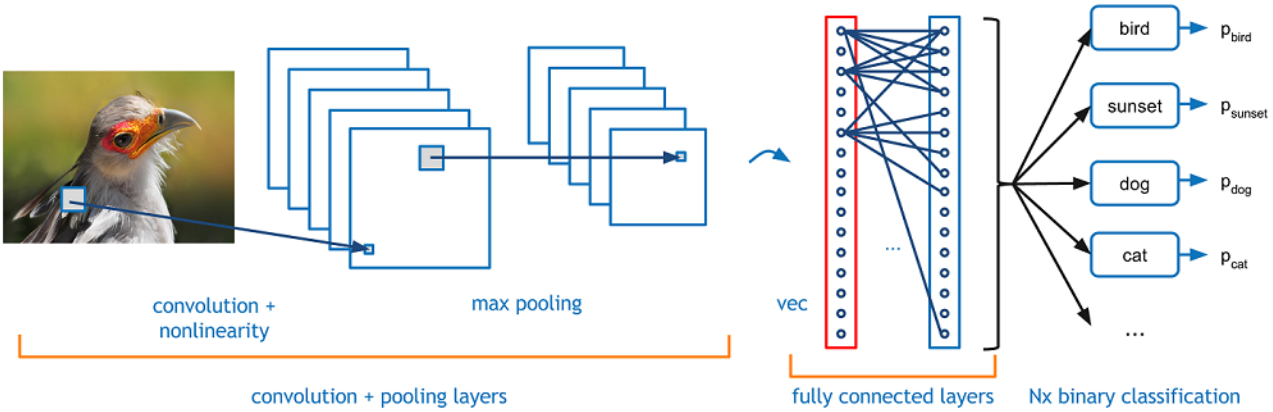
A simplified visualization

Convolution Neural Network

Abdulwahab H. Sahyoun – Oct. 23rd, 2017



Python Libraries used

- Keras (TensorFlow backend)
- Numpy
- Matplotlib
- OpenCV

All running on Anaconda Python 3.5 using Spyder

Note that a lot of Keras function visualizations are simplified in this document. Image used in this experiment:

1- Reading the image

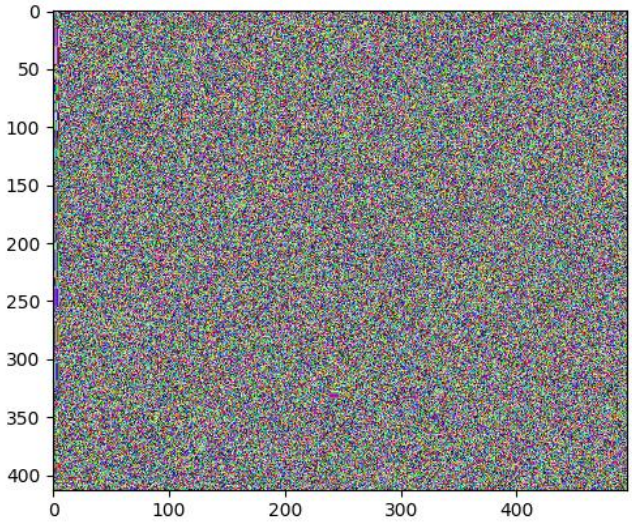


Figure 1- Plotted pixels of cat image after being passed through a single convolution layer (3 filters)

1- Enhancing the view with NumPy

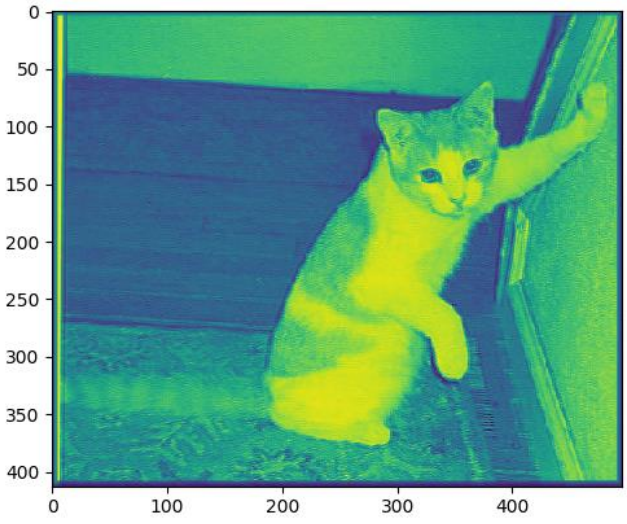


Figure 2- image passed through a convolution layer containing 3 filters with 3 rows , 3 cols each (3x3 matrix) and no actiovation functions

2- Single Filter

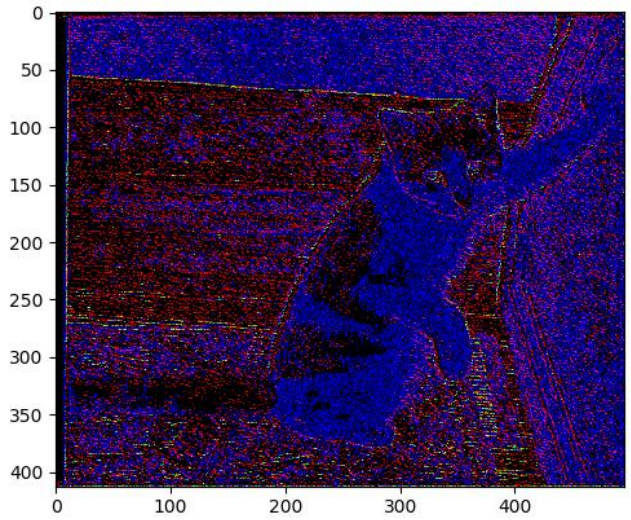


Figure 3- convolution layer with 1 filter (3x3) and no activation function

3- Adding rectifier

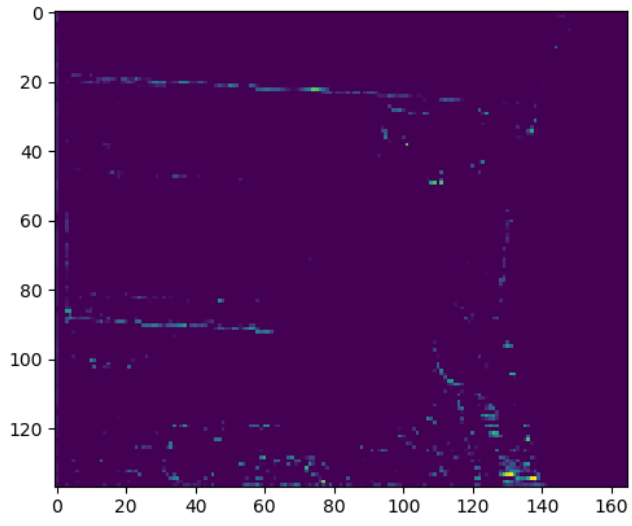


Figure 4 - Conv layer + activation function (rectifier) or 'relu'

4- Pooling the feature map (pooling layer)

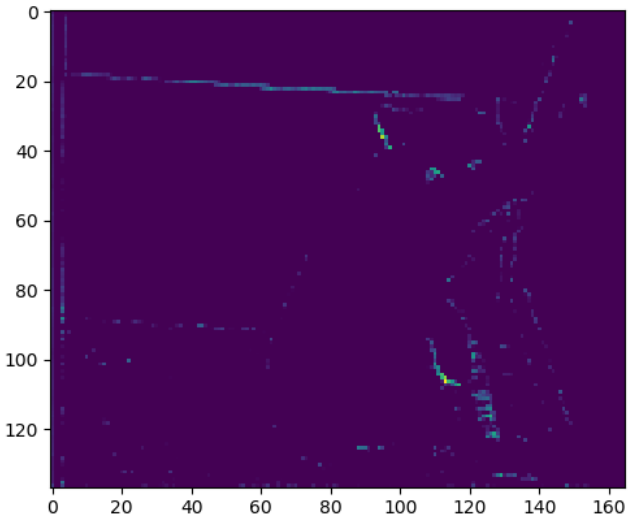


Figure 5 - image resulting from pooled map with a kernel size of (2,2) this divides the input size by 2 (76% accuracy)

5- Adding a second conv. and pooling layer

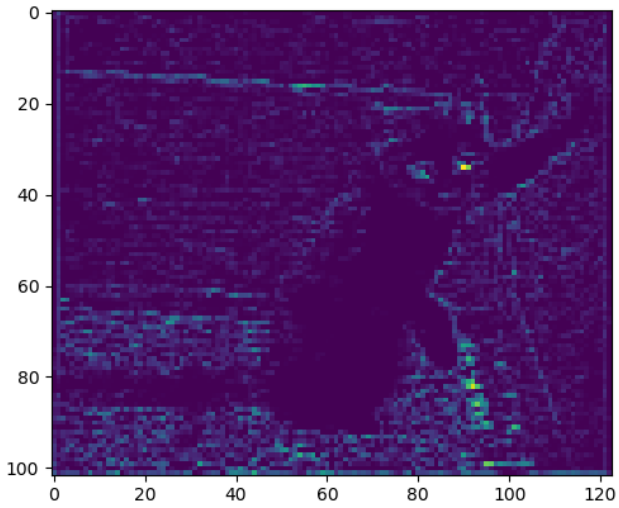
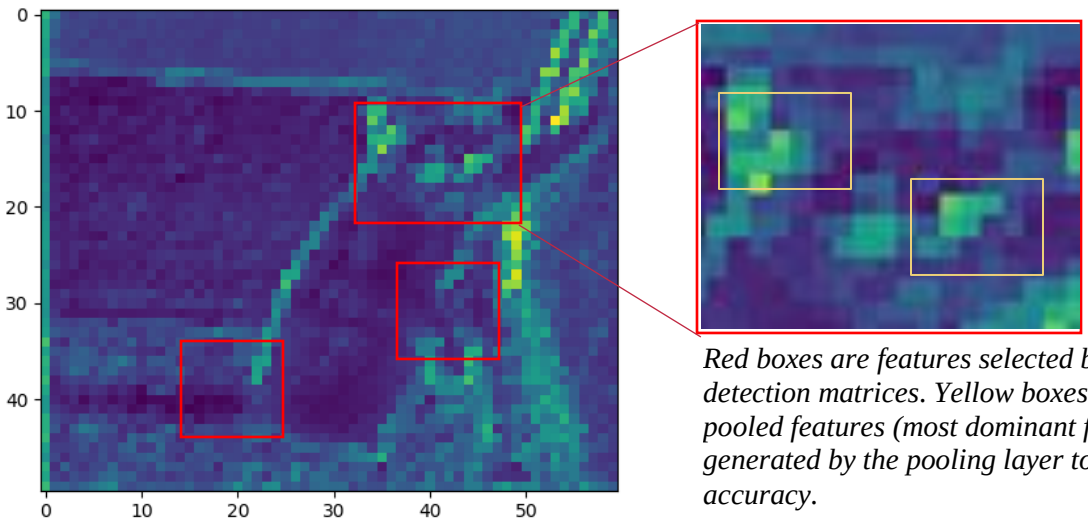


Figure 6- 2 conv layers (each with 1 filter) and 2 pooling layers with same kernel size of (2,2) this results in higher accuracy (85% accuracy)

6- Adding a third conv. and pooling layer



Red boxes are features selected by the detection matrices. Yellow boxes are pooled features (most dominant features) generated by the pooling layer to enhance accuracy.

Figure 7 - increased feature detail and higher accuracy (88% accuracy)- this is the final resulting image that contains feature maps to be flattened and passed through the hidden layer for full connection. The matrix will then be used as an input for the CNN.

7- The CNN

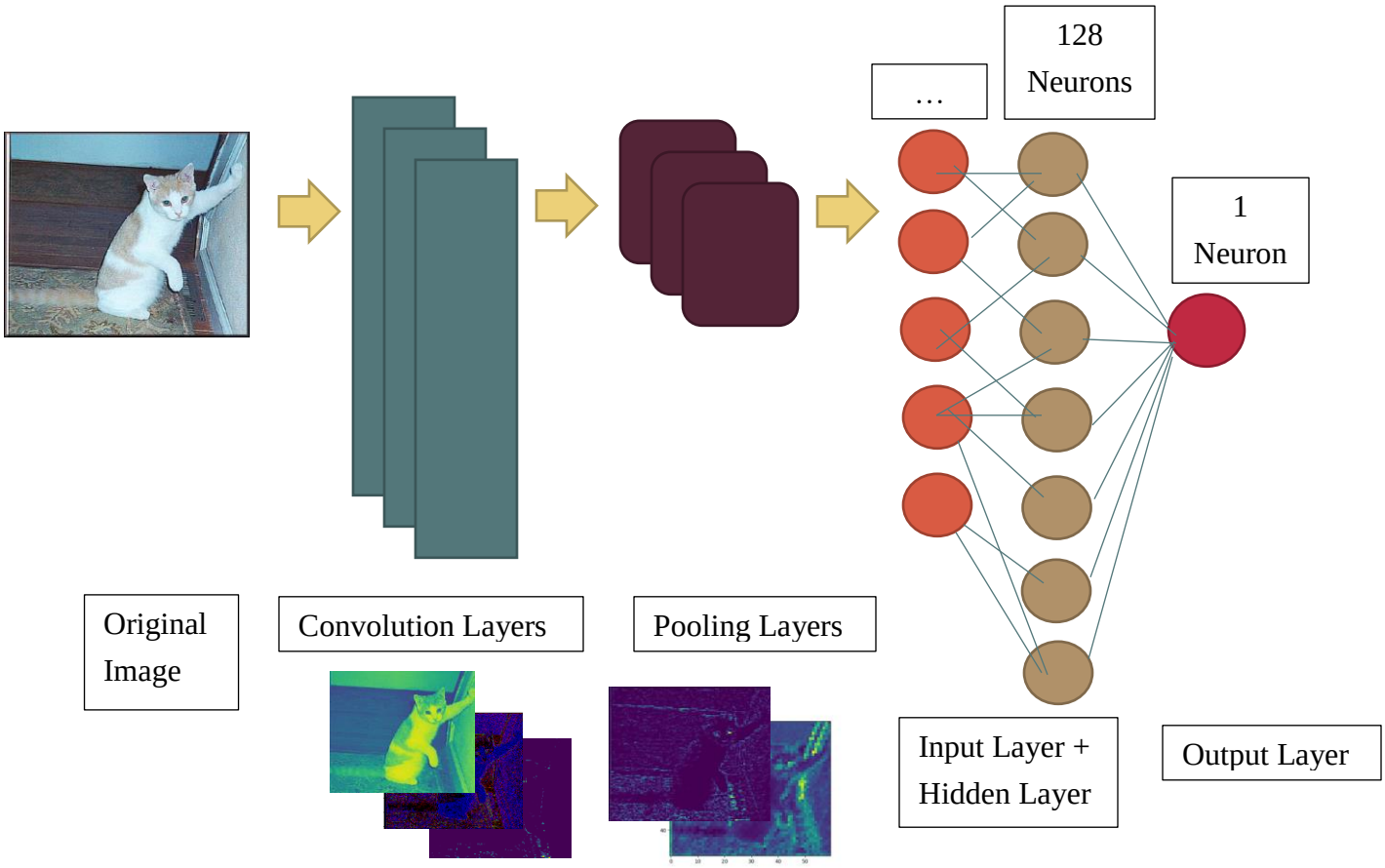


Figure 8 - The CNN