*CS 6001: MACHINE LEARINING IN COMPUTER VISION*

TRAFFIC SIGN CLASSIFIER

*Kaustubh Deshpande*

*Sohan Samanta*

ABSTRACT

Self-driving cars are a rapidly evolving technology, which until a few years back was still at the design table. But today, for the car manufacturers, it has become a race against time to launch their fully autonomous vehicle on the market before their competitors. Thus, with companies like Tesla, Intel, Google, Uber, BMW to name a few, competing for the number one position in self driving cars, the demand for real world perception algorithms are higher than ever.

Some critical decisions that an autonomous vehicle system needs to be able to decipher include object detection, lane detection, traffic sign classification, etc. And thus enters the role of machine learning in machine vision. Machine learning has seen strides in its performance over the last few years, mainly in their branch dealing with deep learning and convolutional neural networks.

A few of the reasons for such improvements in performance is owed to the availability of larger training data sets, powerful GPU implementations and better regularization techniques.

Recognition of traffic signals consists of two parts:

1. Image detection and
2. Classification.

In this project, we endeavor to create a working model using convolutional neural networks that can classify detected images of traffic signals.

INTRODUCTION:

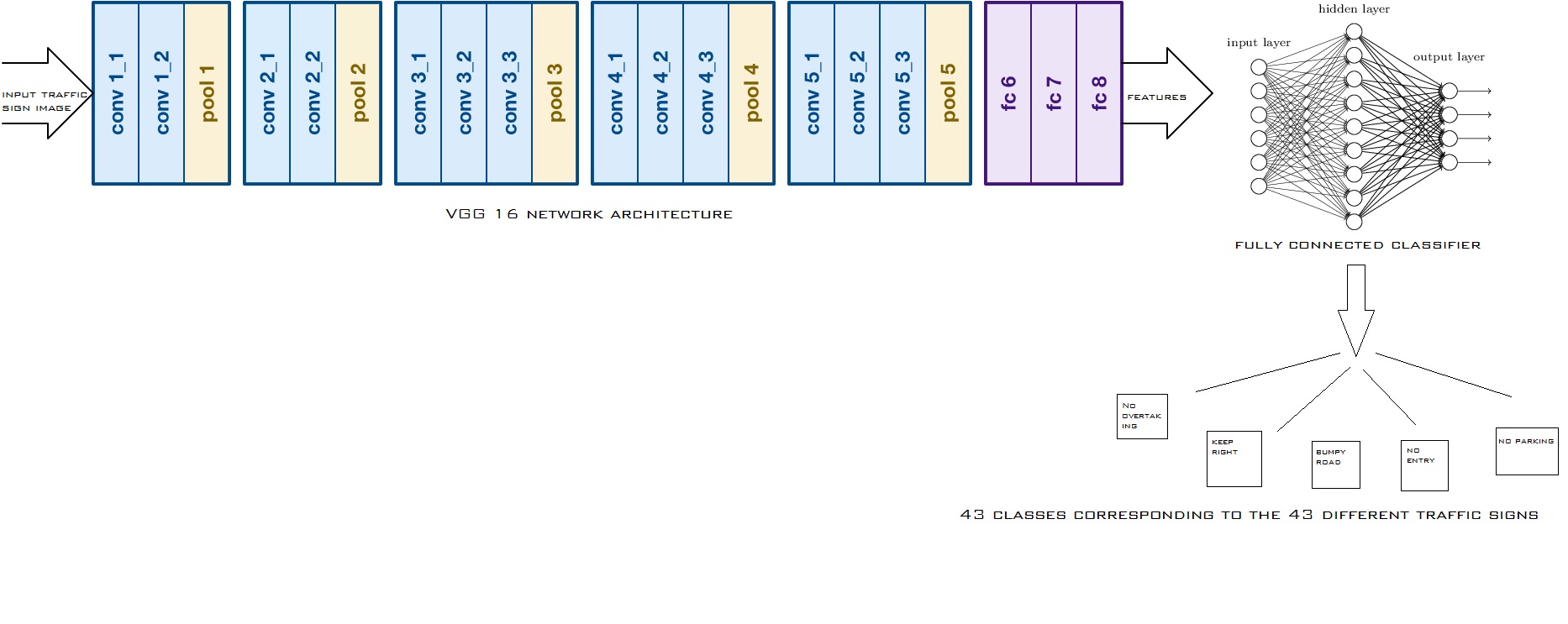
The autonomous mobile robot system deals with perception, classification and then acting on the classified information to take real world decisions. Below image shows an autonomous mobile robot system representation by Roland Seigwart. The information extraction part inside the perception layer is the one we will be dealing with in this project.



We will be classifying traffic signals, from images captured by the autonomous vehicles, from its on board cameras or any other image capturing devices available. The data set used is the German Traffic sign data set available online.

In our program, we will be using the VGG, which is a deep convolutional network, predominantly used for object recognition and comes pre-trained by the Visual Geometry Group of Oxford. This cuts down our work for the training of the convolutional network. We have used the VGG network to compute features from our input data and then used these features as input to our own fully connected layers to give the required outputs.

The structure of our network would look something like the figure shown in the next page.



Now, the output layer will have 43 classes corresponding to the 43 classes in which the German traffic signal has been classified into.

Overview: -

Let us now look at the dataset and the deep networks that we have used for this project.

*Dataset: -*

We have extensively applied our deep networks on the proposed GTSRB (German Traffic Sign Recognition Benchmark.) GTSRB is the standard state-of-the-art revelation benchmark for traffic sign recognition/ classification. The German Traffic Sign dataset consists of 43 different traffic signs with each image having 32x32 colour size. This dataset has 39,209 images as training data and 12,630 images as testing data. Each image is a 32x32x3 array of pixel intensities represented as [0,255] integer values in RGB colour space. Class of each image is encoded as an integer in a 0 to 42 range. Sample images from the GTSRB dataset are as shown below in the figure.



*VGG Net*

VGG Net was first developed by Karen Simonyan and Andrew Zisserman at University of Oxford for the ImageNet Challenge 2014. Two of their best performing models were publicly made available for research. We have decided to use one of such models. It is as shown in the figure below:



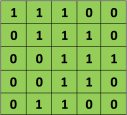
The deep network shown above consists of 16 weighted layers and 3 fully connected layers with max pooling and ReLU layers in between. The final fully connected layer is followed by a SoftMax layer. We shall discuss about the ReLU and Pooling layers in detail in the next section. We will be using this VGG net as a feature extractor. This VGG net will be used as a pre-trained network instead of a randomly initialized weighted network since pre-trained networks provide better results. We then process the obtained features through a simple classifier to separate them into the required different classes.

*Fully Connected Classifier*

Before considering what, we have implemented for this project let us first look at a few important terminologies involved with convolutional neural networks.

1. Convolution operator

The primary purpose of this convolutional operator is to extract features from the input image. Convolutional preserves the spatial relationship between pixels by learning image features using small squares of input data. For simplicity of understanding, consider a 5x5 image whose pixel values are only 0 and 1 and consider another 3x3 matrix as shown below:

 Screen Shot 2016-07-24 at 11.25.24 PM

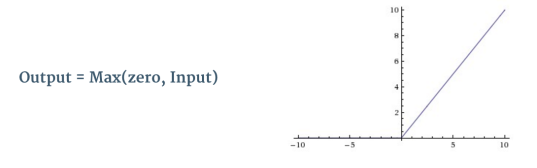
Then the result of the convolutional of the two matrices is as shown below:

|  |  |  |
| --- | --- | --- |
| 4 | 3 | 4 |
| 2 | 4 | 3 |
| 2 | 3 | 4 |

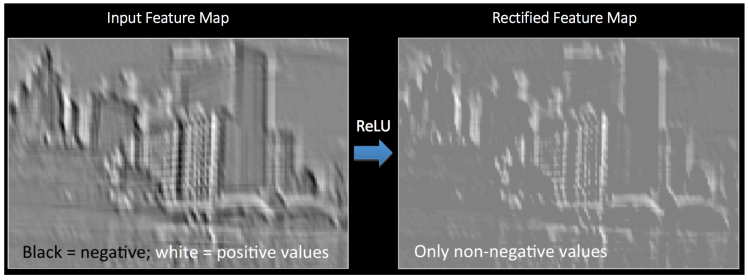
Let us understand how we obtained this result. We slide the orange matrix (3x3) over our original image (5x5) by 1 pixel (also called ‘Stride’) and for every position, we compute element wise multiplication and add the multiplication outputs to get the final integer which forms a single element of the output matrix. We call this 3x3 matrix a ‘filter’ or ‘feature detector’ and the matrix formed by sliding over the image and computing the dot product is called the ‘Convolved Feature’ or the ‘Feature Map’.

1. ReLU Non-Linearity

ReLU stands for Rectified Linear Unit and is a non-linear operation. Its output is given by



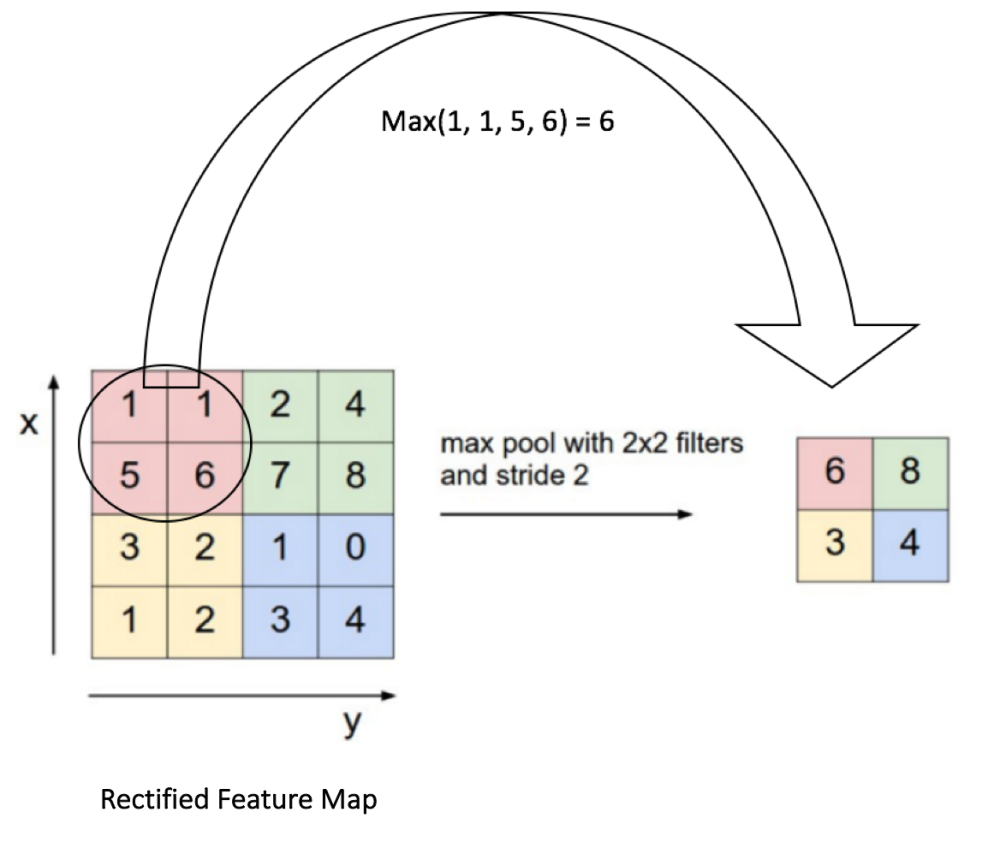
It is an element wise operation and replaces all the negative pixel values in the feature map by zero. For better understanding of ReLU operation, look at the figure below:



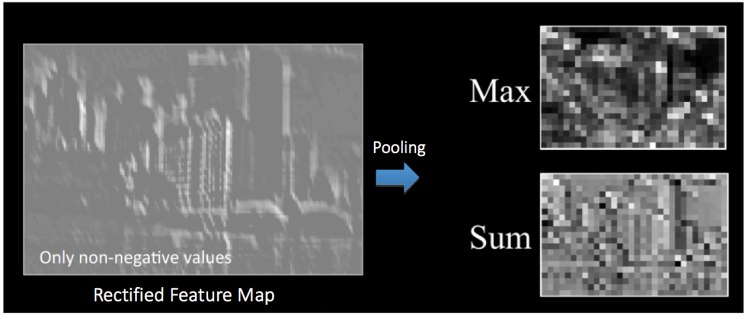
ReLU is preferred over other activation functions such as tanh or sigmoid since it deals with the problem of vanishing gradients better than them.

1. Pooling Operation

Pooling also called subsampling or down sampling reduces the dimensionality of each feature map. There are different types of pooling Max, Average, Sum, etc. We have used Max Pooling in this project. In case of Max Pooling, we define a spatial neighbourhood and take the largest element from the rectified feature map within that window. The figure below shows how max pooling works



The figure below shows the effect of the Pooling operation on the Rectified Feature Map we received after the ReLU operation.



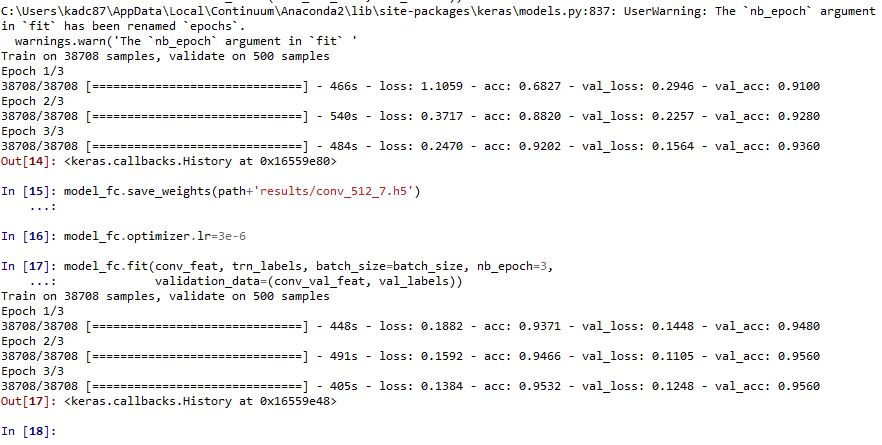
One of the benefits of performing this pooling operation is that it reduces the number of parameters and computations in the network thereby helping us to control overfitting.

1. Fully Connected Classifier

After fine tuning our pre-trained network, we use it with a simple classifier to separate the obtained the features into different classes. We then split the pre-trained network into two parts and use one of them to convert the images into features by convoluting them into a 512x14x14 shape. These filtered features can now be separated into classes using our simple classifier. This smaller classifier network consists of two dense layers containing 512 hidden neurons followed by the last dense layer which uses the SoftMax function to provide a (43,) output which will be a probability of the sign being in a certain class.

Experimentation

The original VGG 16 network was downloaded to use as a feature extractor. It was fine-tuned till it reached 70% accuracy. Then the network was fine-tuned for the current dataset which makes the network to obtain the features from the given dataset. Then the network was split into two parts a) Convolutional Layers b) Fully Connected Layers. Then the model from the given convolutional layer was designed. This model was used as a feature extractor and the idea was to use these features to classify the dataset using a simple classifier rather than using a complete deep neural network since it is very time consuming. The training is carried out by using the ‘Adam’ optimizer. The batch size is set to 64 and the training was regularized by dropout regularization for the fully connected layers with the dropout ratio being 0.125, 0.5 and .25 respectively. We have separated the dataset into Training and validation datasets. We do this to have some sort of confirmation of our fully connected classifier for any unseen data. We have 38,709 images in the training dataset and 500 images in the validation dataset. The initialisation of the network weights for our classifier is important, since bad initialisation can stall learning due to the instability of gradients in the deep networks. To overcome this problem, we trained our classifier networks with random initialisation and ran it for 3 epochs with a learning rate of 10^-3 and saved this model. Then to improve our validation accuracy we changed the learning rate to 3e-6 and refitted our model on the validation data. The images below show the successful training procedure of our classifier.



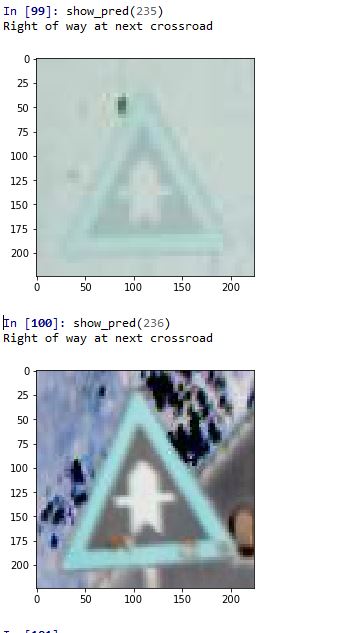
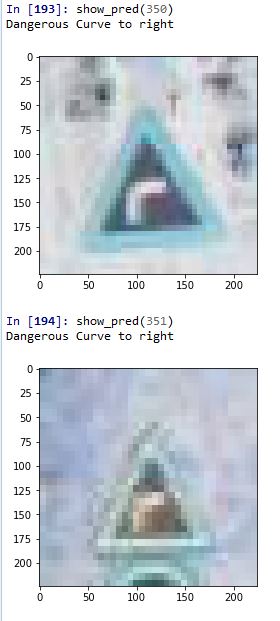
Results

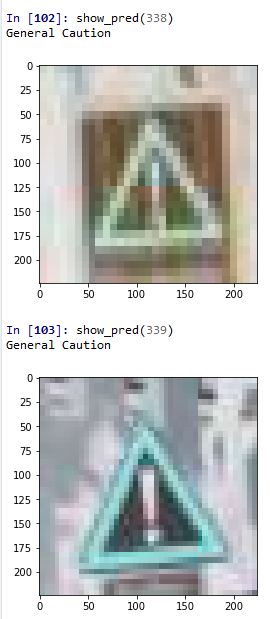
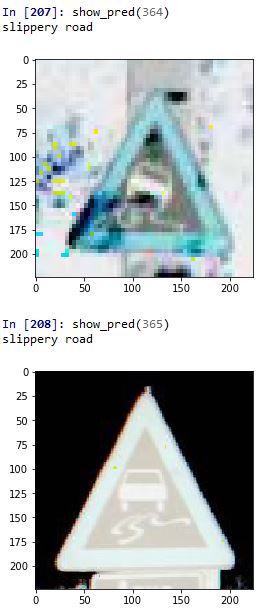
Let us now discuss the results we obtained after implementing our model on the GTSRB. We have used only the validation accuracy as the parameter for success in this project. Due to heavy computations, this project was very time consuming and coupled with other problems like Memory errors we could not get satisfying results for the actual testing dataset. So, the results for only the validation dataset are included.

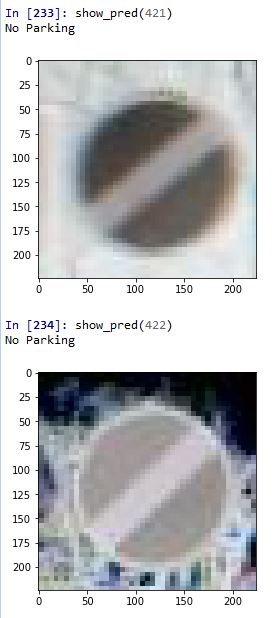
Final Training Accuracy: 0.9530

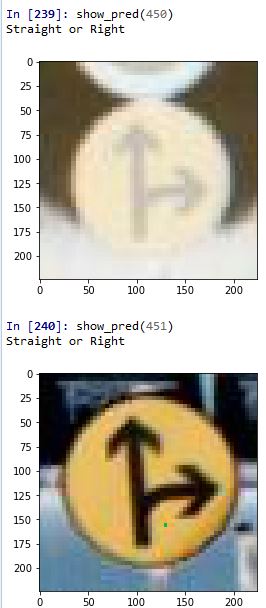
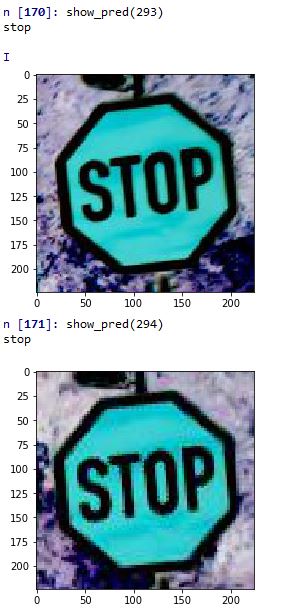
Final Validation Accuracy: 0.9560

Here are few of the images from the validation dataset which were correctly classified and incorrectly classified respectively.

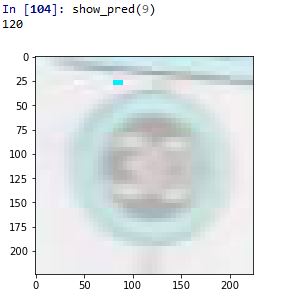
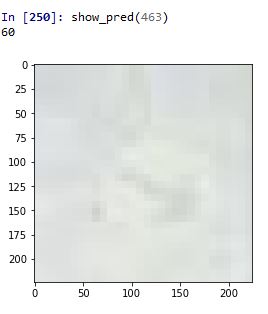








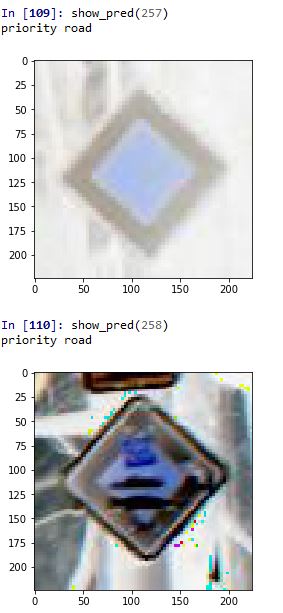
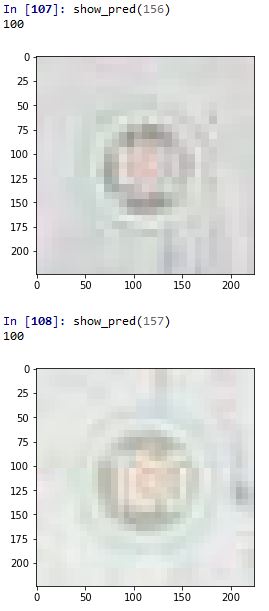
Incorrectly Classified



We can see from the images above that the actual image is the ‘Keep Right’ sign but it is incorrectly classified as ‘Speed Limit 60’ sign and in the other image the ‘Speed Limit 30’ sign is incorrectly classified as ‘Speed Limit 120’ sign.

Conclusion

The classifier could classify the training dataset successfully with 95.30 % accuracy and the validation dataset with an accuracy of 95.60%. This shows that our model was not overfit. With the appropriate approach like training a proper deep net we found that our methodology of implementing a pre-trained network provided us very good results and was less time consuming. The fine-tuned pre-trained network could extract features from the dataset images successfully. Even for bad quality images from the given dataset it could correctly classify the signs as shown below.

For future scope, we can improve the results by approaching a few data augmentation techniques such as cropping, horizontal and vertical flipping. Such methods have proven to improve the results by a significant margin. Also, we have included the results for only the training and validation datasets, we have not tested our model on the testing dataset due to time constraints but we are confident about our validation dataset results and sure that this model will provide very good results for the testing dataset too.

References

1. VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION Karen Simonyan ∗ & Andrew Zisserman + Visual Geometry Group, Department of Engineering Science, University of Oxford {karen, az}@robots.ox.ac.uk

2. <http://mlss.tuebingen.mpg.de/2015/slides/fergus/Fergus_1.pdf>

3. [https://en.wikipedia.org/wiki/Kernel\_(image\_processing)](%20https:/en.wikipedia.org/wiki/Kernel_(image_processing))

4. <http://andrew.gibiansky.com/blog/machine-learning/convolutional-neural-networks/>

5. <http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution>

6. [https://www.quora.com/How-is-a-convolutional-neural-network-able-to-learn-invariant-features](%20https:/www.quora.com/How-is-a-convolutional-neural-network-able-to-learn-invariant-features)