CONVOLUTIONAL NEURAL NETWORKS

Assit.Lec Rasha Amer Kadhim Al- Khalissi
MS.c.. Computer science
College of Agriculture
Animal Production Department
Diyala University
Email : rashaakm@agriculture.uodiyala.edu.iq

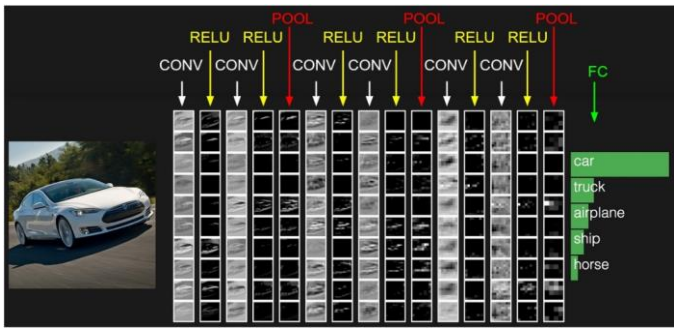
***Abstract*—** Convolutional Neural Networks (ConvNets or CNNs) are a part of Neural Networks that have been exceptionally compelling in areas of image recognition and classification. ConvNets have been effective in recognizing faces, objects and traffic signs robots and self driving cars .[1]

I have attempted to clarify the primary ideas driving Convolutional Neural Networks in straightforward terms. I clarify the most well-known ConvNet Architectures and I clarify profound Learning Libraries that used to fabricate CNN

Keywords— Convolutional Neural Networks; ConvNets ; CNN; classification; image recognition ; Layer; ReLU; Pooling; Backpropagation; LeNet ; AlexNet ; ZF Net ;GoogLeNet; VGGNet; ResNets; DenseNet; Caffe; Deeplearning4j; Theano; TensorFlow; Torch; Keras; MXNET.

1-INTRODUCTION

CNN have been some of the most powerful innovations in the field of computer vision. The first year that neural nets grew when Alex Krizhevsky used them in year's ImageNet competition dropping the classification error record from 26% to 15% and he win the competition that was in 2012 ,since then, a lot of companies have been using deep learning at their services. Such us Facebook uses neural nets for their automatic tagging algorithms, Google for their photo search, Amazon for their product recommendations, Pinterest, and Instagram.[2] CNN have figured out how to sort pictures into classes far better than people in some cases (figure 1). It is easy to understand especially when we break them down into their essential parts[3]. I'll walk you through these essential parts and portray them.



(figure 1)

2-HOW CNN WORKE

CNNs are active at processing data in the form of arrays, which makes it ideal for computer vision tasks (Lecun et al., 2015). CNNs are based on Multilayer Perceptrons (MLP), since these consist of fully connected layers, they do not scale well with image sizes. Interestingly a CNN tries to exploit the spatially neighborhood relationship in pictures, by stacking the component maps and just interfacing every neuron to a little locale of the information volume, this is also called the receptive field of the convolutional layer. For each feature map, the weight and bias will be shared, this is possible by assuming that a feature which is useful to compute at one position, is also useful to compute at another spatial position.[4][5].

The conv layer receives the raw image data. It uses its filters to recognize the features that it look important . Each filter produces a 2 dimensional array of dot products . this arrays called an activation map. The activation maps for all the filters in one layer are put together into a 3 dimensional activation volume. This volume will be the output of the conv layer. After the conv layer and ReLU the data passes through a max pool layer This cycle (Conv layer, ReLU layer, then Maxpool layer) happens several times, depending on the size of your CNN. The results from the last conv fed into a fully connected layers. Finally, the output of this last fully connected layer goes through a softmax layer which converts the fully connected layer's output into a set of class probabilities, one foreach class.[4][5].. (figure 2)

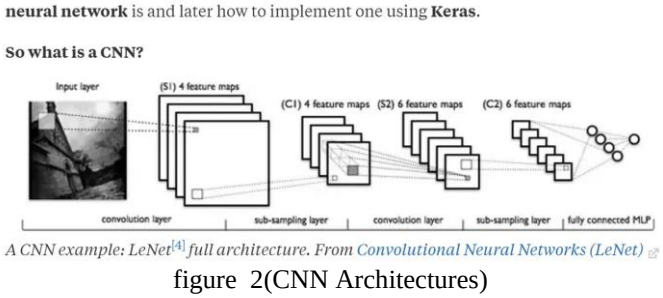


figure 2(CNN Architectures)

In the first layers a CNN detects simple features, such as edges, then corners. In the later layers, the network starts to learn more complex features, which it is seem t to the human eye. Activation function CNNs are developed of neurons, these have learnable weights and biases and can be communicated by the linear function:

$$y = w .x + b$$

w is the weight, x the input and b is the bias.

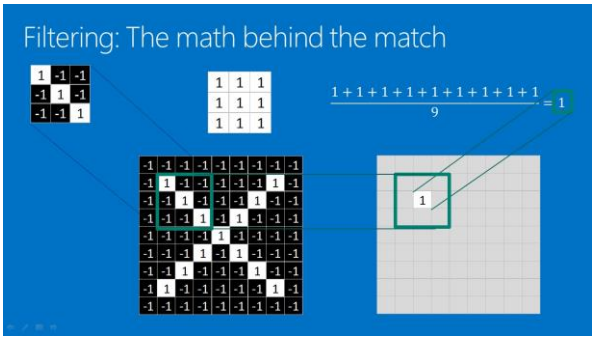
3-THE CNN STEPS

"the main purpose of Convolution in case of a ConvNet is to extract features from the input image. "[16].

A-Convolutional Layer :- this is the most important part. It takes an image as an input and produces a smaller image where each pixel are a results from a mathematical convolution(process of a filter) with neighboring pixels also called kernel .In one convolution layer, the network will apply multiply filters [6] .

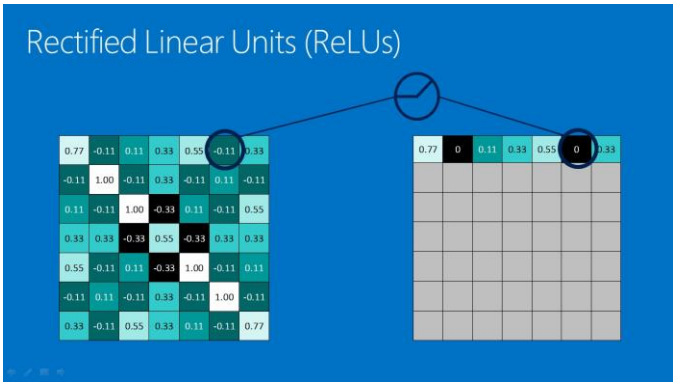
A CNN comprises of multiple convolutional layers. These layers scan over the image, using a small, M * M pixel feature detector called a filter, and mark the regions of the image that align most closely with that filter. It does this by sliding a M*M window over the image and computing the dot product between that window's pixel values and the filter's pixel values. A CNN normally consist of several convolutional layers, an activation function, pooling layers and lastly the classification layer, which is normally afully connected Neural Network [5].

"To calculate the match of a feature to a patch of the image, simply multiply each pixel in the feature by the value of the corresponding pixel in the image. Then add up the answers and divide by the total number of pixels in the feature(figure 3). If both pixels are white (a value of 1) then 1 * 1 = 1. If both are black, then (-1) * (-1) = 1. Either way, every matching pixel results in a 1. Similarly, any mismatch is a -1. If all the pixels in a feature match, then adding them up and dividing by the total number of pixels gives a 1. Similarly, if none of the pixels in a feature match the image patch, then the answer is a -1 .The next step is to repeat the convolution process in its entirety for each of the other features. The result is a set of filtered images, one for each of our filters."as [3]



(figure 3) Convolutional Layer

B -Introducing Non Linearity (ReLU) :- An additional operation called ReLU is used after every Convolution operation by replaces all negative pixel values in the feature map by zero(figure 4).. The purpose of ReLU is to introduce non linearity in ConvNet[1],



(figure 4)ReLUs

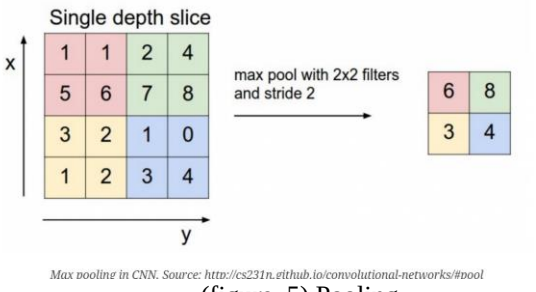
The activation function is an optional part of the nodes, it familiarizes a non-linearity to the output of the node. The proposed system uses Recified Linear Unit (ReLU) as the activation function, which can be communicated as:

$$f(x) = \max(0;x)$$

The reason for using the ReLU, is it's computational efficiency, resulting in less training time. It doesn't have an issue with vanishing gradients and has shown to greatly accelerate convergence (Glorot et al., 2011).

C - Pooling layer : A key part of Convolutional Neural Networks are pooling layers. Typically applied after the convolutional layers. Pooling layers subsample their input. The mean of pooling is to apply a max operation to the result of each filter [8]. This layer will be utilized to simplify an input by "pooling" neighboring pixels. Regularly the pooling function will be the maximum [6].

"Pooling is a way to take large images and lessen them down while keeping the most important information in them. It consists of stepping a small window across an image and taking the maximum value from the window at each step." As [3] (figure 5).



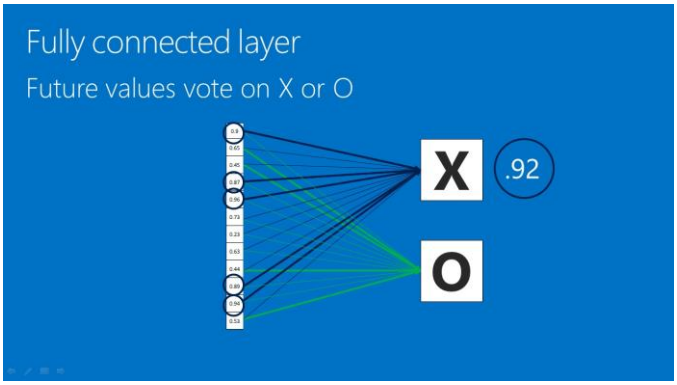
(figure 5) Pooling

Pooling (subsampling or downsampling) diminishes the dimensionality of each feature map but holds the most imperative information. Spatial Pooling can be of various types: Max, Average, Sum etc. In case of Max Pooling, we characterize a spatial neighborhood (for instance, a 2×2 window) and take the biggest component from the corrected feature map within that window. Rather than taking the biggest component we could likewise take the average (Average Pooling) or the sum of all components in that window.. In practice, Max Pooling has been shown to work bette[1].

A pooling layer is added between every convolutional layer. The function of it, is to reduce the spatial size and that will reduce the amount of parameters. This also helps to control overfitting [9].

D- 4-Fully connected layer:- The Fully Connected infers that each neuron in the past layer is connected to every neuron on the following layer .' The output from the convolutional and pooling layers represent high-level features of the input image.' [ ].The reason for the Fully Connected layer is to utilize these features for characterizing the input image into various classes based on the training dataset [1].

Fully connected layers take the high-level filtered images and make an interpretation of them into votes Rather than regarding inputs as a two-dimensional array, they are dealt with as a single list and all treated identically. Each value gets its own vote . These votes are communicated as weights, or connection strengths, between each value and every classification (figure 6). [3].



(figure 6) Fully connected layer

E- Backpropagation :

The proposed system utilizes the Cross-entropy cost capacity to compute the error, which is then utilized by backpropagation in order to calculate the gradient for each weight. In conclusion gradient descent is used to utilized the weights's changes that needs to be applied during the network, before starting over a correct learning rate can be primary Choose , "a higher learning rate results in faster learning, but it might not end up at the ultimate minimal loss. Choosing a too low value can result in very slow convergence, while a too high value can result in oscillation" (Wilson and Martinez, 2003).

Use Backpropagation to ascertain the inclinations of the error regarding all weights :-

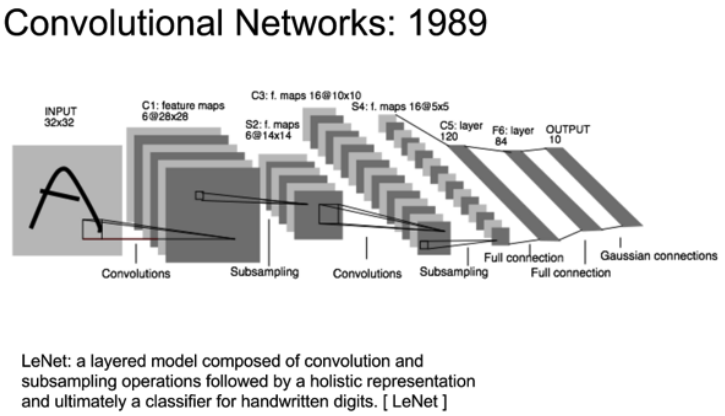
Error = right answer – actual answer

The fully connected layer implements classification while the loss layer tries to find the error The idea is that the network learns by its mistake and then updates the parameters, weights and bias, throughout the system.

4- CNN ARCHITECTURES

1-LeNet (1990s to 2012):. (figure 7)

- The first successful applications of Convolutional Networks .
- developed by : Yann LeCun in 1990's.
- used to read zip codes, digits, etc.[10].



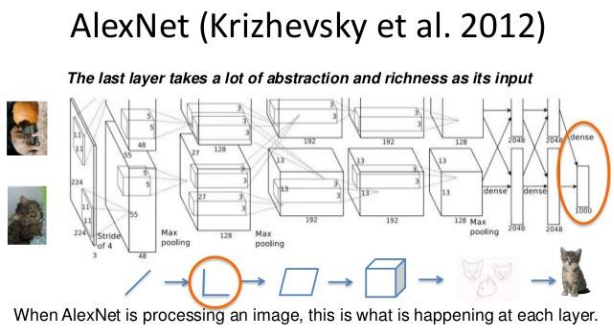
LeNet: a layered model composed of convolution and subsampling operations followed by a holistic representation and ultimately a classifier for handwritten digits. [LeNet]

(figure 7) LeNet

2- AlexNet (2012) :-

- The first CNN that work in Computer Vision .
- developed by Alex Krizhevsky, Ilya Sutskever and Geoff Hinton.
- The AlexNet was submitted to the ImageNet ILSVRC challenge in 2012 and significantly outperformed the second runner-up (top 5 error of 16% compared to runner-up with 26% error).
- It is "very similar architecture to LeNet but was deeper, bigger, and featured Convolutional Layers stacked on top of each other (previously it was common to only have a single CONV layer always immediately followed by a POOL layer)." [10].

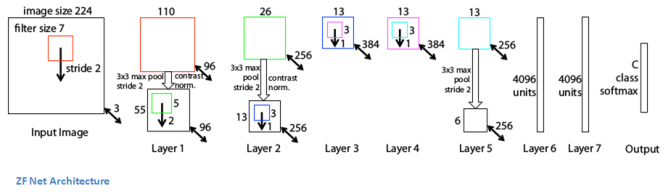
AlexNet scaled the insights of LeNet into a much larger neural network that could be used to learn much more complex objects and object hierarchies. (figure 8)



(figure 8) AlexNet

3- ZF Net (2013) –

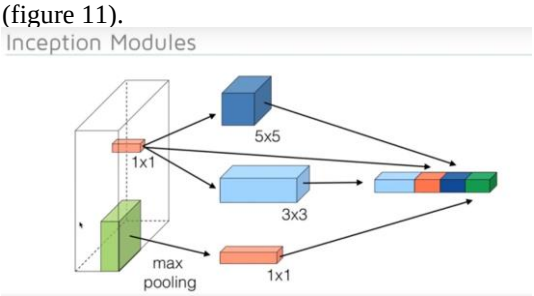
- developed by The ILSVRC 2013 winner was a Convolutional Network from Matthew Zeiler and Rob Fergus.
- It known as the ZFNet (short for Zeiler & Fergus Net).
- "It was an improvement on AlexNet by tweaking the architecture hyperparameters, in particular by expanding the size of the middle convolutional layers and making the stride and filter size on the first layer smaller." [10]. (figure 9).



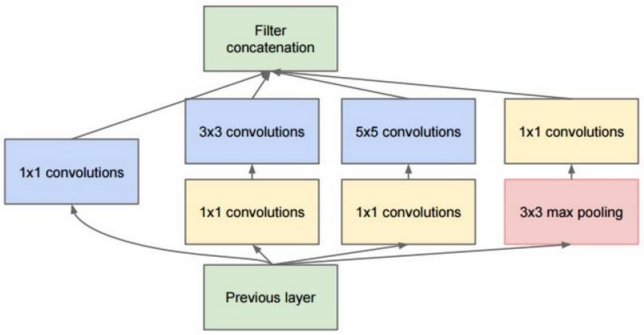
(figure 9) ZF Net

4- GoogLeNet (2014):-

- developed by Google The ILSVRC 2014 winner was a Convolutional Network from Szegedy et al.
- " Its main contribution was the development of an Inception Module that dramatically reduced the number of parameters in the network (4M, compared to AlexNet with 60M). Additionally, this paper uses Average Pooling instead of Fully Connected layers at the top of the ConvNet(figure10), eliminating a large amount of parameters that do not seem to matter much. There are also several followup versions to the GoogLeNet, most recently Inception-v4." [10]. (figure 11).



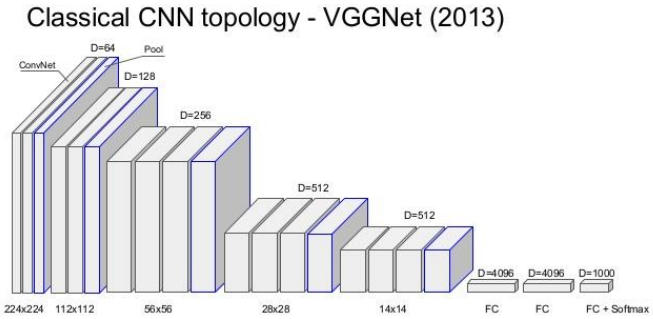
(figure 10) Inception Module GoogLeNet



(figure 11) GoogLeNet

5- VGGNet (2014) –

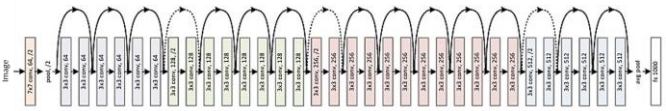
- developed by The runner-up in ILSVRC 2014 was the network from Karen Simonyan and Andrew Zisserman
- It known as the VGGNet.
- "Its main contribution was in showing that the depth of the network is a critical component for good performance. Their final best network contains 16 CONV/FC layers and, appealingly, features an extremely homogeneous architecture that only performs 3x3 convolutions and 2x2 pooling from the beginning to the end. Their pretrained model is available for plug and play use in Caffe (figure 12). A downside of the VGGNet is that it is more expensive to evaluate and uses a lot more memory and parameters (140M). Most of these parameters are in the first fully connected layer, and it was since found that these FC layers can be removed with no performance downgrade, significantly reducing the number of necessary parameters." [10].



(figure 12) VGGNet

6- ResNets (2015) – The revolution then came in December 2015, at about the same time as Inception v3. ResNet have a simple ideas: feed the output of two successive convolutional layer AND also bypass the input to the next layers.

- developed by Kaiming He et al. was the winner of ILSVRC 2015.
- "It features special skip connections and a heavy use of batch normalization. The architecture is also missing fully connected layers at the end of the network. The reader is also referred to Kaiming's presentation (video, slides), and some recent experiments that reproduce these networks in Torch. ResNets are currently by far state of the art Convolutional Neural Network models and are the default choice for using ConvNets in practice (as of May 10, 2016). In particular, also see more recent developments that tweak the original architecture from Kaiming He et al. Identity Mappings in Deep Residual Networks (published March 2016)." [10]. (figure 13)



(figure 13) ResNets

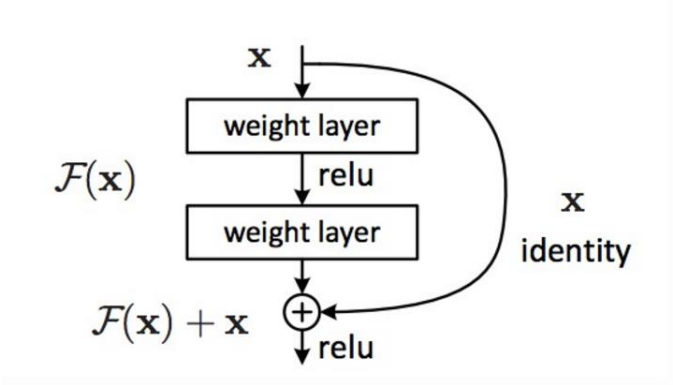
In a traditional network the activation at a layer is know as follows:

$$Y=f(x)$$

Where f(x) is our convolution, matrix multiplication. At each layer the ResNet implements:

$$Y=f(x)+x$$

It allows the slope to pass backwards directly. By amass these layers, the gradient could skip over all the intermediate layers and reach the bottom without being minimize (figure 14).

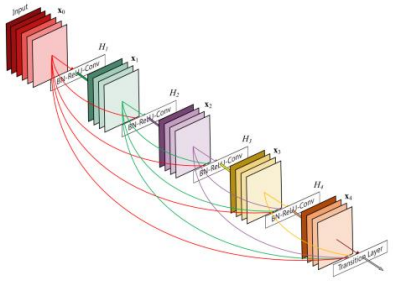


(figure 14) ResNets

7- DenseNet (August 2016) –" Recently published by Gao Huang (and others), the Densely Connected Convolutional Network has each layer directly connected to every other layer in a feed-forward fashion. The DenseNet has been shown to obtain significant improvements over previous state-of-the-art architectures on five highly competitive object recognition benchmark tasks. "[10]. (figure 15)

the DenseNet relies on stacking of layers. Mathematically this looks like:

y = f(x,x-1,x-2…x-n)



(figure 15) DenseNet

5-DEEP LEARNING LIBRARIES

A- Caffe: is a deep learning framework.

- developed by the Berkeley Vision and Learning Center (BVLC) and by community contributors.
- Google's DeepDream is based on Caffe Framework. it is a BSD-licensed C++ library with Python Interface.
- open-source [11][12].

B- Deeplearning4j : is the first commercial-grade.

- distributed deep-learning library written for Java and Scala.
- It is designed to be used in business environments, rather than as a research tool.
- open-source [11][12].

C - Theano: is a low-level library that specializes in efficient computation.

- Creator: Université de Montréal
- Open source
- Interface: Python [11][12].

D -TensorFlow :is another low-level library that is less mature than Theano. it's supported by Google.

- Creator: Google Brain Team
- Open source
- Interface: Python, C/C++ [11][12].

E- Torch : is a scientific computing framework with wide support for machine learning algorithms. It is easy to use and efficient, fast scripting language, LuaJIT, and an underlying C/CUDA implementation.

- Interface : Lua programming language.
- Open source . [11][12].

F- Keras : a high-level neural networks library, written in Python and capable of running on top of either TensorFlow or Theano. It's minimalistic, modular, and awesome for rapid experimentation. It is the best place to start for beginners.[17]

This is my favorite Python library for deep learning .

G - MXNET : s another high-level library similar to Keras. It offers bindings for multiple languages and support for distributed computing.

- Creator: Distributed (Deep) Machine Learning Community
- Open source: Yes
- Interface: C++, Python, Julia, Matlab, JavaScript, R, Scala. [12][13].

6 -Conclusion

Convolutional Neural Networks (ConvNets or CNNs) are a part of Neural Networks that have been exceptionally compelling in areas of image recognition and classification. CNNs give the best execution in image recognition problems and even beat people in specific cases . The viability of convolutional nets in image is one of the principle reasons why the world has woken up to profound learning. They are powering major advances in machine vision, hey are controlling real advances in machine vision, which has clear applications for treatments for the visually impaired , robotics ,and self-driving cars. I have attempted to clarify the primary ideas driving Convolutional Neural Networks in straightforward terms. I explain the most common ConvNet Architectures and I explain deep Learning Libraries that used to build CNN .

7- Future work

I will used CNN to build my parking lots classification model to classify the empty spot and count them. I will used Resnet and Keras and **TensorFlow** to build my model .

REFERENCES

[1] https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

[2] https://adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/

[3] https://brohrer.github.io/how_convolutional_neural_networks_work.html

[4] http://publications.idiap.ch/downloads/papers/2017/Pinheiro_THESIS_2017.pdf

[5] http://vbn.aau.dk/files/244496579/Parking.pdf

[6] https://dsotb.quora.com/Deep-learning-with-Keras-convolutional-neural-networks-demystified#QYgbJ

[7] http://vbn.aau.dk/files/244496579/Parking.pdf

[8] http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/

[9] http://cs231n.github.io/convolutional-networks/#pool

[10] http://cs231n.github.io/convolutional-networks/

[11] http://www.teglor.com/b/deep-learning-libraries-language-cm569/

[12] https://elitedatascience.com/python-deep-learning-libraries

[13] http://blog.revolutionanalytics.com/2016/08/deep-learning-part-1.html

[14] https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2/

[15] https://adeshpande3.github.io/adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html

[16] http://www.kdnuggets.com/2016/11/intuitive-explanation-convolutional-neural-networks.html

[17] https://keras.io/