**DEEP LEARNING NOTEBOOK**

**Shreenidhi Sudhakar**

**Table of Contents**

Convolution Neural Network

* Introduction
* Overview of CNN Layers
* Extraction of Feature - MNIST
* Hyper Parameter Grid Search

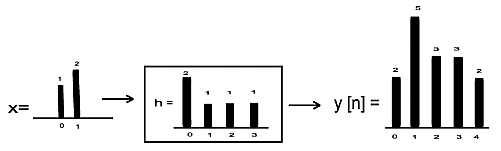
**Convolution Neural Network**

**Introduction**

Convolution layers are used to extract the features from input training samples. Each convolution layer has a set of filters that helps in feature extraction. In general, as the depth of CNN model increases, complexity of features learnt by convolution layers increases. For example, first convolution layer captures simple features while the last convolution layer captures complex features of training samples.

1 line 🡪 2 lines 🡪 Square 🡪 Combination of Squares 🡪 Combinations of squares and other shapes

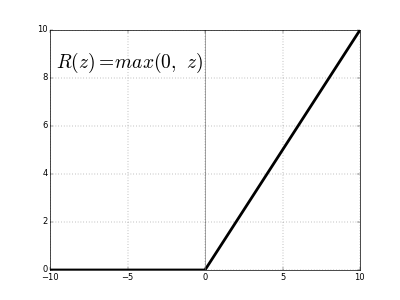
Features are extracted by taking the convolution of portion of data sample under consideration. The amount of data portion that the filter traverses each time is proportional to the stride length and padding value. Data samples may/ may not be subjected to zero padding before convolution.



Convolution of a Signal

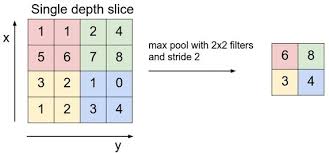
The convolution output is then passed through an activation unit called ReLU (Rectified Linear Unit). This unit converts the data into its non-linear form. The output of ReLU is clipped to zero only if convolution output is negative.

Sigmoid units are not preferred as activation unit because of vanishing gradient problem. If the depth of CNN is large, then by the time the gradient found at the input layer traverses to the output layer, it’s value would have diminished largely. This results in the overall output of the network varying marginally. This, in turn, results in slow/no convergence. To avoid such a situation, ReLU is preferred.



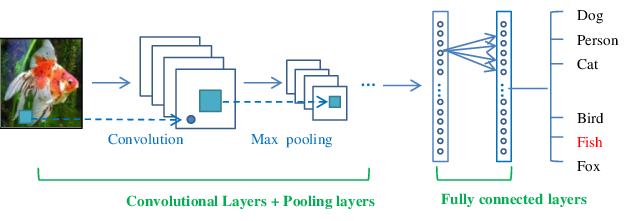
Output of ReLU

The output of ReLU is then passed through a pooling layer. Pooling layer remove any redundant features that’s captured during convolution. Thus, this layer reduces the size of data sample. The principle behind pooling is that it assumes that adjacent values of image pixels are nearly identical. The average/minimum/maximum of four adjacent pixel values are used to carry out pooling. In general, size of input image is reduced by half with help of a 2\*2 filter. The input data may/ may not be subjected to zero padding before pooling.



Max Pooling

This process of passing data through convolution and pooling layer successively is repeated as per the design of CNN model. For learning purpose, this process is repeated 2-4 times. The output from successive convolution and pooling layer is then passed through a multi-layer neural network. Here, each neuron unit acts as feature map that carries information about a unit.

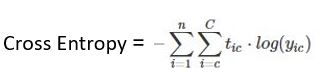


Design of Convolution Neural Network

Dropout layer is used to reduce over-fitting by making the CNN model robust to noise. These layers are generally introduced between 2 fully connected neural network layers. They temporarily cut a portion of data flowing between two fully connected layers. This is equivalent to making the model learn to classify accurately in presence of noise. Thus, chances of model classifying inaccurately because of overfitting is reduced.

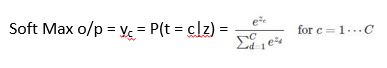
The output of CNN model is calculated using SoftMax function. SoftMax is preferred as it gives the probability of outputs for different classes rather than just >= 0.5 in the case of sigmoid output. The usage of SoftMax function to find output results based on the highest probability of class results in an increase in accuracy the of output.

Cross entropy is used to measure the performance of the system. They are calculated with help of a SoftMax function. The advantage here is that the SoftMax output is the trace of the elements corresponding to the class that we know that the output belongs too. This, in general, saves the computation time.



Here,

* Tic = Target Output
* yic = Soft Max Output
* C= = Number of Classes
* N = Number of Data Samples



Here,

* yc = probability of current output belonging to class c
* Numerator = exponential of weighted sum o/p of class c
* Denominator = sum of exponential of weighted sum o/p of classes 1 to C

More info regarding CNN can be found at:

* [CS231n Convolutional Neural Networks for Visual Recognition](https://cs231n.github.io/convolutional-networks/)
* [A Beginner's Guide To Understanding Convolutional Neural Networks](https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/)

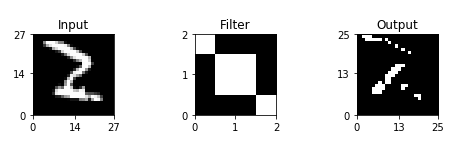
**Overview of CNN Layers**

|  |  |  |  |
| --- | --- | --- | --- |
| Layer | Purpose | Implementation | Reason for Working |
| Convolution | Extract features from i/p data sample | y(t) = x(t)\*h(t-T) | Small size filters trace the entire data sample at a faster rate. Hence, they learn from the mapping with pictures efficiently. |
| ReLU | Convert data to non-linear form | y = y, when i/p > 0  0, otherwise | Removes problem of vanishing gradient descent |
| Pooling | Remove redundant features  Make model invariant to translation, rotation and scaling | y = max of 4 adjacent  feature values | Features next to each other have similar values |
| Fully Connected Neural Network | Classification of i/p data | Back Propagation Algorithm | Each neuron acts as a feature map |
| Dropout | Make model robust to noise | Temporarily cut the flow of small portion of data b/w 2 fully connected layers | Reduces overfitting by making the model less complex |
| Regularization | Make model robust to noise | Penalizes cost function and weight updates for every wrong prediction | Reduces overfitting by making the model less complex |
| Multiple Convolution Layer | Extract high level/more complex features | y(t) = x(t)\*h(t-T) | Lesser number of filters enable faster extraction of features from data set |

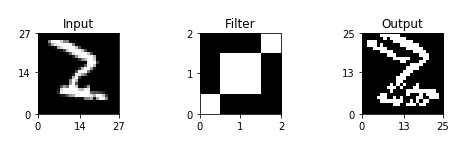
**Extraction of Feature – MNIST**

This section gives the reader an intuitive idea of how convolving an image with a kernel will result in extraction of features from the input image. We consider an image of digit 2 that is being convolved with the same 3\*3 filter flipped by 90 degrees in each example.

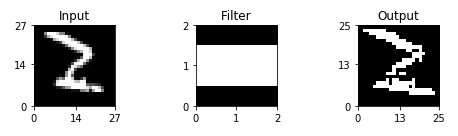
In our experiment, we have assumed threshold of filter as +2. Black pixel is represented as 0 and white pixel is represented as 1. Results of convolution are as seen below.



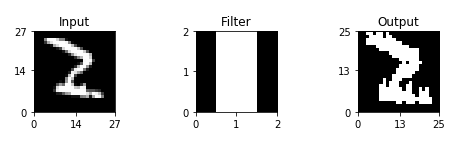
The output of convolution says that filter is good at detecting inner edges. This is shown by distinct edges in image of picture along the diagonal. The 3\*3 filter values used are [[2,-1,-1], [-1,2,-1], [-1,-1,2]]. Thus, when convolution output is greater than threshold of 2 defined, we consider output of that convolution as 1 else 0.



The output of convolution says that the filter is bad at diagonal edges. This is shown by distinct edges in image of picture along the diagonal. The 3\*3 filter values used are [[-1,-1,2], [-1,2,-1], [2,-1,-1]]. Thus, when convolution output is greater than threshold of 2 defined, we consider output of that convolution as 1 else 0.



The output of convolution says that the filter is good at detecting horizontal edges. This is shown by distinct edges in image of picture along the horizontal plane. The 3\*3 filter values used are [[-1,2,-1], [-1,2,-1], [-1,2,-1]]. Thus, when convolution output is greater than threshold of 2 defined, we consider output of that convolution as 1 else 0.



The output of convolution says that the filter is good at detecting vertical edges. This is shown by distinct edges in image of picture along the vertical plane. The 3\*3 filter values used are [[-1,2,-1], [-1,2,-1], [-1,2,-1]]. Thus, when convolution output is greater than threshold of 2 defined, we consider output of that convolution as 1 else 0.

**Hyper-Parameter Grid Search**

Grid Search is used to find the best combination of hyper-parameters that help convnet to achieve highest accuracy. This approach is preferred as it’s difficult to configure a convnet accurately as it depends on a lot of hyper-parameters. Few examples of convnet hyper-parameters are, size of kernel, dropout, number of convolution layers, batch size, type of activation function, stride of filters, and so on.

Different combination of convnet hyper-parameters yield different accuracy of inputs. In our experiment, we have assumed dropout and kernel size to be our hyper-parameters that we are going to vary. It’s found that kernel size of 5 and dropout of 0.1 yields the best convnet model. Architecture design of the this best convnet model is as below:

|  |  |  |  |
| --- | --- | --- | --- |
| Layer | Input | Output | Filter |
| Conv\_1 | (?, 28, 28, 1) | (?, 24, 24, 8) | 5\*5 |
| Pool\_1 | (?, 24, 24, 8) | (?, 12, 12, 8) | 2\*2 |
| Conv\_2 | (?, 12, 12, 8) | (?, 8, 8, 32) | 5\*5 |
| Pool\_2 | (?, 8, 8, 32) | (?, 4, 4, 32) | 2\*2 |
| Dropout\_1 | (?, 4, 4, 32) | (?, 4, 4, 32) | N/A |
| Flatten | (?, 4, 4, 32) | (?, 512) | N/A |
| FC\_1 | (?, 512) | (?, 64) | N/A |
| Dropout\_2 | (?, 64) | (?, 64) | N/A |
| Output\_Layer | (?, 64) | (?, 2) | N/A |

Table 1 Architecture Design of Best Model Found Using Hyper-Parameter Grid Search

Note:

* Pool = Pooling
* Conv = Convolution
* FC = Fully Connected

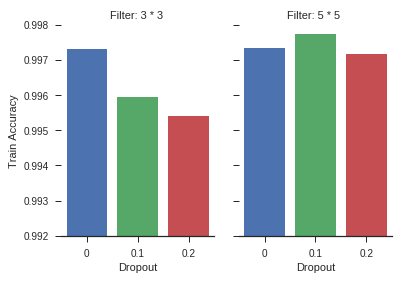
We infer from our experiment that hyperparameters play a pivotal role in choosing a model that enhance its accuracy. In our case, increase in dropout rate and reduction in filter size has been found to reduce accuracy of model. These results can’t be argued to hold good for other convnet models designed for different dataset.

Source Code

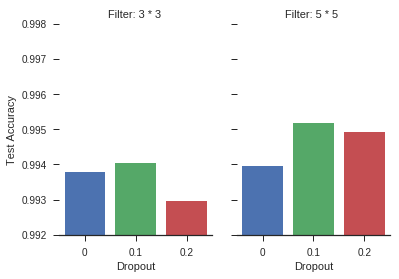
* <https://github.com/shree6791/Deep-Learning/blob/master/CNN/MNIST/keras/src/intern_task_1.ipynb>

Accuracy

* Train



* Test



*\* Training accuracy is always greater than or equal to test accuracy.*

Source Code

* <https://github.com/shree6791/Deep-Learning/blob/master/CNN/MNIST/keras/src/intern_task_1.ipynb>

More info on Hyper-Parameter grid search can be found at:

* [How to Grid Search Hyperparameters for Deep Learning Models in Python With Keras](http://machinelearningmastery.com/grid-search-hyperparameters-deep-learning-models-python-keras/)