ijates
ISSN 2348 - 7550

OVERVIEW OF CONVOLUTIONAL NEURAL NETWORKS

Purvil Bambharolia

Department of Computer Science and Engineering, Institute of Technology, Nirma University (India)

ABSTRACT

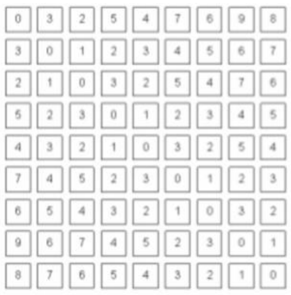
Neural networks are one of the most powerful technologies that are used for a variety of classification and prediction problems. This paper summarizes convolutional neural network which is the new buzzword in the world of machine learning and deep learning. They are similar to simple neural networks. Convolutional neural networks involve a huge number of neurons. Each neuron has weights and biases associated with them which can be learned over time to fit the data properly. Convolutional neural networks, referred to as CNNs, are used in a variety of deep learning problems. They can be used for classification as well as prediction problems which involve images as input. Some of the examples of such problems are – facial key point detection, emotion detection, facial recognition, speech recognition, etc. In this paper, we will also focus on how much better are CNNs than simple neural networks by illustrating our claims on MNIST data set.

Keywords—convolutional neural networks, CNN, data mining, machine learning

I. INTRODUCTION

A convolutional neural network is a feed-forward multi layer neural network. They were introduced back in 1990's and the pattern represented by their neuron connections is actually inspired from the visual cortex of various animals according to [1]. Researchers have found a way to emulate the visual cortex of animals which is one of the most powerful computer vision processing systems.Convolutional neural networks are specifically designed to work with problems involving images as inputs. CNNs can be used to solve machine learning or data mining problems wherein inputs can be represented by an image or a set of images. CNNs can be visualized as a modified version of multi-perceptron neural network model [2]. CNNs have taken over the simple neural networks because all the techniques that we applied to simple neural networks can still be applied to much more powerful CNNs. According to [3], big tech giant companies such as Facebook and Google are using deep convolutional neural networks with a large number of convolutional layers for various purposes such as face recognition and search by image. Before discussing the working of convolutional neural networks, let us now identify what the problems with computer vision tasks such as image classification are. We as humans having a visual cortex situated in our brain, are capable of identifying little details such as dark and light regions in an image. We can seamlessly identify various patterns occurring in our surroundings as well as that in the scene we are looking at. We identify these patterns using our previous knowledge of identified patterns. Devoid of a visual cortex, a computer just sees everything we feed it in terms of numbers. It visualizes pictures or images as a matrix of numbers, each number representing a value corresponding to a pixel.

Fig.1 What we humans see



0	3	2	5	4	7	6	9	8
3	0	1	2	3	4	5	6	7
2	1	0	3	2	5	4	7	6
5	2	3	0	1	2	3	4	5
4	3	2	1	0	3	2	5	4
7	4	5	2	3	0	1	2	3
6	5	4	3	2	1	0	3	2
9	6	7	4	5	2	3	0	1
8	7	6	5	4	3	2	1	0

Fig.2 What computers see

II. INPUTS AND OUTPUTS

According to [3], inputs given to convolutional neural networks are an array of pixel values with each value ranging from 0 to 255. For example, if the input is an image of dimensions 50x50 then the array would contain 50x50x3 = 7500 values. Each pixel in an image is represented by 3 values. These 3 values represent the intensities of RGB i.e., red, green and blue. In the case of image classification, the job of CNN would be to take this image i.e., an array of pixel values, as an input and output the probabilities of it belonging to a certain class. In the case of prediction problems, the model would take an array of pixel values as input and predict the corresponding output values.

Taking an example of classifying oranges and watermelons. We could classify them based on their shape, taste, colour, etc. Thus, if we could find unique identifiable features of oranges and watermelons, we could detect whether a newly acquired fruit is an orange or a watermelon. In a similar way, the computer is able to perform such complex tasks of identifying features and classifying the inputs. A CNN model first identifies low-level features such as dark and light regions. In the next step, it tries to identify more complex features such as edges and curves. Successively, it can build more abstract high-level features such as legs, face, paws, hands, etc. through a series of convolutional layers.

III. STRUCTURE

A simple neural network is defined by only 2 dimensions. A CNN is defined by 3 dimensions namely height, width, and depth. The depth refers to the third dimension of the activation volume [2]. Considering an image with 3 color channels i.e., red, green and blue, the depth of the network would be equal to 3.

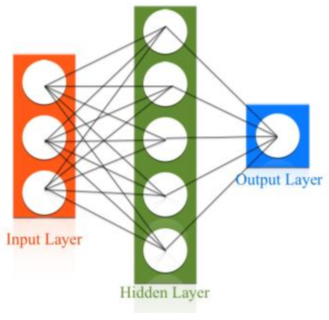


Fig. 3 Simple Neural Network

Each layer of the CNN model transforms a 3-dimensional input to 3-dimensional output. CNN model is a sequence of layers. Let us now discuss the types of layers used to build a CNN and how the training process works.

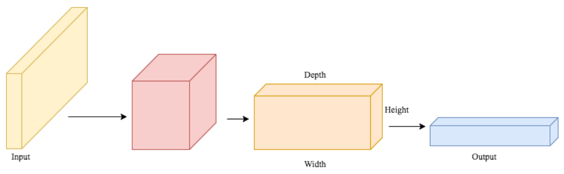


Fig. 4 3 dimensional input transformation to 3 dimensional output

3.1 Convolutional layer

The most important part or layer of a convolutional neural network is a convolutional layer. There may be one or more convolutional layer. Generally, the first layer in CNN model is a convolutional layer. Deep convolutional neural networks involve a high number of convolutional layers. These layers are responsible for identifying low level as well as high-level complex features in a given input.

A very good way to understand a convolutional layer is to imagine a torch flashing light on the input neurons [4]. Consider that a 6x6 image is passed as an input in the convolutional layer. Now this 6x6 matrix of pixel values in convoluted in the following manner. First we decide a few parameters such as filter size. Filter size is often referred to as kernel size as well. For illustration, we select a filter size of 3 by 3. This filter is also an array of numbers called weights having the same depth as the input. For illustration purposes, consider a torch capable of flashing light at a matrix who size is same as our selected filter, flashing light at the top left corner of the input neurons. We multiply the matrix on which the torch is flashing light and the filter element wise.

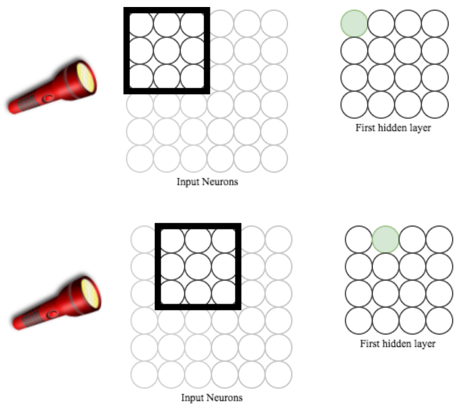


Fig 5 Process of convolution

$$O_{1,1} = \sum_{i=1}^{i=3} \sum_{j=1}^{j=3} F_{i,j} * I_{i,j}$$

$$O_{1,2} = \sum_{i=1}^{i=3} \sum_{j=2}^{j=4} F_{i,j} * I_{i,j}$$

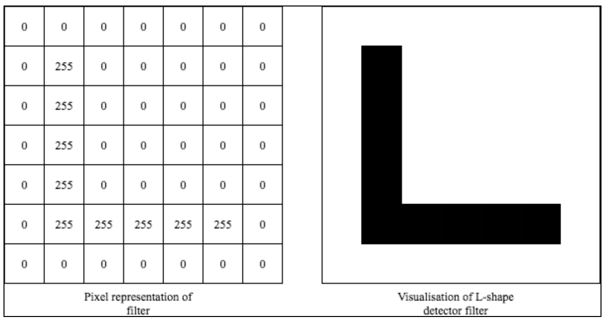
O = Feature map (Output matrix)

F = Filter

I = Input neurons

The region over which the torch is shining the light is termed as a receptive field. The filter slides over the input neurons and as it slides, it multiplies the elements of the filter with the elements in the receptive field element-wise. The process of sliding the filter is called convolving. Every unique receptive field on the input neurons gets us a number. These numbers form what we call feature map. Feature maps are sometimes referred to as activation maps. A large number of feature maps are created by using different filters.

Each filter represents a feature such as edge, curve, etc. Filters can also be referred to as feature identifiers. Complex features can be detected using high level filters. For illustration purposes, let us look at a simple filter for detecting a L-shape ignoring the depth.



0	0	0	0	0	0	0
0	255	0	0	0	0	0
0	255	0	0	0	0	0
0	255	0	0	0	0	0
0	255	0	0	0	0	0
0	255	255	255	255	255	0
0	0	0	0	0	0	0

Pixel representation of filter
Visualisation of L-shape detector filter

Fig. 6 Visualisation of filter

In the above figure, we can visualize what an L-shaped detecting filter would look like. A filter is a pixel structure which will have high values at certain places where it expects a part of a shape to exist [3]. When we apply this filter to a part of the input image, we will eventually multiply the two matrices namely filter and part of the image element-wise to get a number. If this number is high, we say that there is some L-shape in that part of the input image which caused the filter to activate [3]. To summarize the process of convolution, first we select a filter corresponding to the feature or pattern we are trying to identify. We then perform convolution over the input image and prepare a feature map for further processing.

There are other two parameters associated with a convolutional layer. Stride is the amount of units by which the filter slides over an input image [2]. In the above illustrations, value of stride was equal to one. Stride controls how the filter convolves around the input image [3].

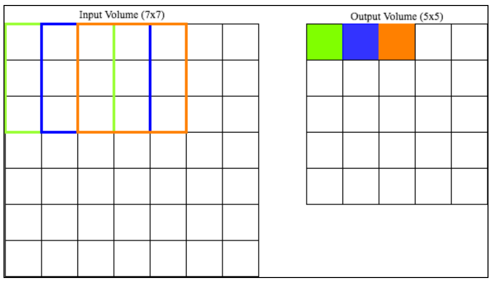


Fig.7 Stride = 1

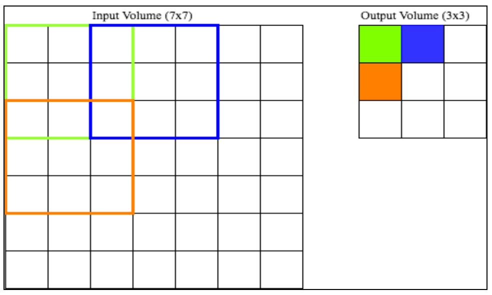


Fig. 8 Stride = 2

The other parameter is padding. Zeroes are padded on the border of the input. It is also referred to as zero padding. Zero padding provides control of the output volume spatial size. Zero padding, depth, and stride forms a group of parameters called hyperparameters [2].

3.2 ReLU layer

ReLU stands for Rectified Linear Units. The purpose of this layer is to introduce non-linearity in the system. Until convolutional layer, the system has been making calculations linearly. In the past, sigmoid and tanh functions were used to introduce non-linearity [3]. According to [2], ReLU layer applies max(0,x) to all the activations formed by the convolutional layer. In naïve terms, the purpose of ReLU is to replace negative activations by 0.

3.3 Pooling layer

According to [4], pooling layers are generally used just after the convolutional layers. Their purpose is to simplify information outputted by the convolutional layer [4]. In sum, it prepares a condensed feature map from the feature map generated in the previous step. One of the most common procedures for pooling is max-pooling. Let us consider the following example.

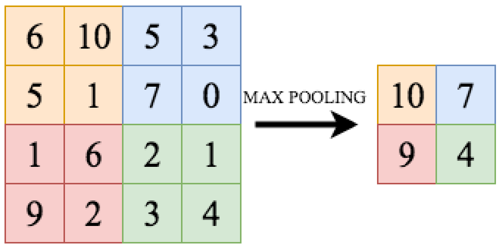


Fig.9 Max pooling

Considering the above example, we have a 4x4 feature map being converted to a much more condensed feature map of dimensions 2x2. We slice up the feature map into equal sub regions. We calculate the maximum value in each of these sub regions and output it to the new feature map. While extracting features from image, we are not concerned with exact or absolute positions of these features. Moreover, we are concerned with relative positions of these features. Pooling reduces the amount of parameters and weights by a significant 75%. This reduces computation cost. It also controls the problem of over fitting [1] [2].

Other techniques such as average-pooling and L2-norm pooling can also be used [3].

3.4 Fully-connected layer

This layer is the last layer of the CNN model. Its job is to input 3-dimensional volume and output an N dimensional vector where N is the number of classes. In case of MNIST dataset, N would be equal to 10. Fully connected layer at the end of the model would provide us a N dimensional vector. Each value in this vector represents the probability of the input image belonging to a certain class. For example,
[0.5 0.1 0.23 0.33 0.9] is a 5-dimensional vector outputted by the fully-connected layer, 0.5 is the probability of the input given to the model belonging to class 1, 0.1 is the probability of it belonging to class 2 and so on. Since, probability of input belonging to class 5 is the highest, we conclude that our CNN model classifies the input to class 5. This is possible by having a softmax function as a loss function on the last fully-connected layer [2]. Such approach of predicting the probabilities of the input belonging to each class is called softmax approach [3]. The way a fully-connected layer works in CNN model is that it takes the activation or feature maps of high level features generated in the previous step and determines to which particular class it most correlates to [3]. The high level features might be paws, legs, face etc.

IV. TRAINING THE MODEL

The most important part of a neural network is the training process. The model needs to adjust its weights as well as filter values. One of the most popular technique for doing that is backpropogation.
Before the CNN model starts the training, its weights, biases and filter values are completely randomized. The filters at the lower layer do not have any idea whether to look for an edge or a curve and filters at the higher level do not know which high level features to look for. Let us assume that we have a training set consisting of images with its corresponding correct labels. Backpropogation can be broken down into 4 parts, forward pass, loss function, backward pass, and weight adjustment [3][4]. During the forward pass, an image from the training set is passed into the model as an input. The model will output something random as the weights, biases and filter values of our model are completely random and hence it fails to identify low-level features as well as high-level ones. Loss function now calculates the error between the output of the model and the correct label which we can obtain from the training set. Most common loss function is mean square error.

$$MeanSquareError = \frac{1}{n}\sum_{i=1}^{n}(target - output)^2$$

Fig. 10 General formula for mean square error

The loss for first couple of iterations would be high. Now, we need to adjust the weights of our model using this loss and correct label. We develop a cost function and try to minimise that cost function. We use optimisation algorithms like gradient descent to reduce this cost function. We do not discuss those topics in this paper.

V. COMPARISION

In this section, we compare the performance of CNN and simple neural network. We test the performance by classifying hand written digits using MNIST data set. Hand written digit recognition using MNIST data set is considered to be the 'hello world' program in the field of machine learning. The dataset contains several images of 0 to 9 digits. Each image is 28 pixels wide and 28 pixels long.

I designed a simple neural network and a CNN by following a guide on Tensor flow's official website [5]. Libraries used for implementing these neural nets are tensorflow, keras, and numpy. Simple neural network has a single hidden layer with 100 neurons. To measure the loss error, I used root mean square method. I trained the neural network for 1000 epoch and it gave me an accuracy of 96.77 % on the testing set.

In the convolutional neural network, there are two convolutional layers, one relu layer and one fully connected layer. To measure the loss error, I used root mean square error. I trained the neural network for 1000 epoch. The maximum accuracy achieved on the testing set was 98.56 %. This is a very big difference in terms of accuracy as compared to simple neural network.

The problem with simple neural network is that it looks at the input images at a whole and not it much detail. For example, if the training set has all the images in which digits are at perfectly at the centre of the image. If we now, supply an image with digit a bit shifted from the centre, the simple neural network will fail to identify the digit. In case of CNN, the image is broken down into sub parts and regions and then the neural network is trained. This causes the CNN to look at the image more closely and in much detail. This makes high level features like legs, hands, etc. to be identified and extracted by the neural network

VI. CONCLUSION

From the structure and working of a convolutional neural network model, we can conclude that it is a much powerful form of neural network model. It is useful in solving problems where inputs to the neural network can be represented in form of an image. They are useful in wide range of applications such as facial key point detection, face recognition, object detection, image classification problems, etc.

REFERENCES

[1] "Convolutional Neural Networks (Lenet) — Deeplearning 0.1 Documentation". *Deeplearning.net*. N.p., 2017. Web. 8 Apr. 2017.

[2] "CS231n Convolutional Neural Networks for VisualRecognition", *Cs231n.github.io*, 2017. [Online]. Available: http://cs231n.github.io/convolutional-networks/. [Accessed: 08- Apr- 2017].

[3] A. Deshpande, "A Beginner's Guide To Understanding Convolutional Neural Networks", *Adeshpande3.github.io*, 2017. [Online].Available:https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/. [Accessed: 08- Apr- 2017].

[4] M. Nielsen, "Neural Networks and Deep Learning", *Neuralnetworksanddeeplearning.com*, 2017. [Online]. Available: http://neuralnetworksanddeeplearning.com/chap6.html. [Accessed: 08- Apr- 2017].

[5] "MNIST For ML Beginners | TensorFlow", *TensorFlow*, 2017. [Online]. Available: https://www.tensorflow.org/get_started/mnist/beginners. [Accessed: 12- Apr- 2017].

[6] "Deep MNIST for Experts | TensorFlow", *TensorFlow*, 2017. [Online]. Available: https://www.tensorflow.org/get_started/mnist/pros. [Accessed: 12- Apr- 2017].