International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 04 Issue: 07 | July -2017 www.irjet.net p-ISSN: 2395-0072

Handwritten Digit Recognition Using Convolutional Neural Networks

T SIVA AJAY[1]

School of Computer Science and Engineering
VIT University
Vellore, TamilNadu,India

Abstract - *Handwritten digit recognition is gaining a huge demand in the branch of computer vision. We are going to implement a better and accurate approach to perceive and foresee manually written digits from 0 to 9. A class of multilayer sustain forward system called Convolutional network is taken into consideration. A Convolutional network has a benefit over other Artificial Neural networks in extracting and utilizing the features data, enhancing the knowledge of 2D shapes with higher degree of accuracy and unvarying to translation,scaling and other distortions. The LeNet engineering was initially presented by LeCun et al in their paper.The creators excecution of LeNet was primarily focused on digit and character recognition.LeNet engineering is clear and simple making it easy for implementation of CNN's. We are going to take the MNIST dataset for training and recognition. The primary aim of this dataset is to classify the handwritten digits 0-9 . We have a total of 70,000 images for training and testing. Each digit is represented as a 28 by 28 grey scale pixel intensities for better results. The digits are passed into input layers of LeNet and then into the hidden layers which contain two sets of convolutional,activation and pooling layers. Then finally it is mapped onto the fully connected layer and given a softmax classifier to classify the digits.We are going to implement this network using keras deep learning inbuilt python library.*

***KeyWords*:ConvolutionalNeuralNetworks (CNN's),LeNet, Artificial Neural networks**

1.INTRODUCTION

In our half of the globe of our mind otherwise called V1 ,contains millions of neutrons with billions of connections between them. The thought is to take digits into consideration and build up a framework which can gain from these. At the end the neural system utilizes the cases to naturally construct rules for perceiving handwritten digits. There are two types of neurons accessible in our brain are perceptron, sigmoid neuron. To calculate the yield we will present weights computing the significance of the separate contributions to the yield. The neuron's give an output of 0 or 1 if the weighted sum is below or above some threshold value. Various decision making models are formed by different weights and threshold values. In the network the first layer of perceptron's that makes very simple decisions ,by multiplying the weights with the inputs. In this way a perceptron in the second layer can make even more complex decision than a perceptron in the first layer. The layers away from the first layer make progressively more complex decisions compared to the first layer. For learning purpose we should continuously change the weights so that the network finds out the aggregate and compares it with a threshold value of bias. If a small change in the weights modifies the output in the direction we want to proceed then we can use small weights or we can take large weights for training ,this method is like hit and trial which we use in solving higher degree polynomials. The architecture of neural networks is divided into three categories,the input layer neurons, the output layer neurons , the layers in between input and output layer called as hidden layers. Sometimes the networks have multiple layers they are coined as Multilayer perceptrons or MLP's. The input layer of our network consists of input neurons encoding the values taken from input pixels of our handwritten digit.Our training data which is fetched from MNIST data set consists of many 28 by 28 pixel images and so input layer contains 784 input neurons. The second layer of our network will be the hidden layer and it takes the aggregated output of first layer and applies activation function to detect the pattern of input images. We will experiment with different values for the number of neurons in the hidden layer. Next coming to output layer of our network contains 10 neurons, each neuron if fired gives any output between 0 to 9 ,so this is the basic working of neural networks for an image. Similar to those of artificial neural networks but with a small modification comes Deep learning networks also termed as Convolutional networks with a higher degree of accuracy for classification. In Deep networks features are extracted not from the entire domain of input but several features are extracted from parts of the domain. Our deep networks use LeNet architecture which enables the networks fast to train and obtaining better results of classification. Keras is a high-level deep networks library written in python and capable of running by using backend as either Tensor Flow or Theano. It was developed for focusing mainly on deriving fast training. Supports both convolutional and recurrent networks and combination of both also. The main principle of this library is modularity , which is understood as a sequence of fully configurable modules that can be combined together with little constraints on them. In particular neural layers, cost functions, optimizers, activation functions are all modules that you can combine to form a new network. The main aim of keras is modularity ,a way of architecting the

layers. Hence we use keras deep learning library for implementing our architecture.

2. LITERATURE REVIEW

2.1 Fully Connected Multi-Layer Nueral Network:

A Multi-Layer Neural Network with one or more number of hidden units is capable of classifying the digits in MNIST data set with a less than 2 % error rate on test set.This network extracts features based on the entire spatial domain of images hence the number of parameters required is very high. The problem with these networks is they tend to be over parameterized, in order of 100,000's which is unwanted when working with complex classification problems with complex data sets.

2.2 K-nearest neighbour classifier:

A KNN classifier with a distance measure like Euclidean distance between the data sets input images is also capable of classification of digits but at higher error rate than a fully connected ML neural network. The key features of this classifier is that it requires no training time and no input from the programmer in terms of knowledge for designing the system. The big over head of this classifier is memory requirement and the classification or recognition time. We take into consideration that this nearest-neighbor system works on raw pixels instead of feature vectors.

3. RESEARCH FRAMEWORK

The ability of Convolutional networks which are trained with gradient descent to learn tougher,multi dimensional,non convex mappings from large datasets. In the traditional model of digit recognition , a hand desgined feature extractor gathers important features from the input and eliminates other distortions. A trainable classifier then classifies the features into classes. In this type fully connected layers can be used as the classifiers. There are also problems in this type of implementation firstly, images are large, often with hundred pixels and by contrast in the next layer would contain several weights. Such a large number of weights are often good for improving the training accuracy but the memory requirement is very huge. But the main problem lies in unstructured nets for image recognition is that they have no built in invariance with respect to translations and disturbances of inputs. Before being sent into a neural net , the images must be normalized and centralized in the input field. But in our case there is no pre-processing step which can normalize the inputs. This will cause the variation in the position of different features in input image.In principle a fully connected network of sufficient size can learn to predict outputs that are variant with respect to such variations.

3.1 Convolutional Networks

Convolutional neural networks use three basic factors to implement classification and recognition problem. In the local receptive fields which are the fully connected layers ,the input were taken as vertical column of pixel intensities. In a convolutional net, we will take it as a 28 by 28 square matrix of neurons , which corresponds to the input image. Here we won't connect every input pixel in first layer to every other neuron in the hidden layer , instead we make connections in small and localized regions of input image. Let's say for example a 5 by 5 region , corresponds to 25 input pixels. So for a neuron we might have connections like.

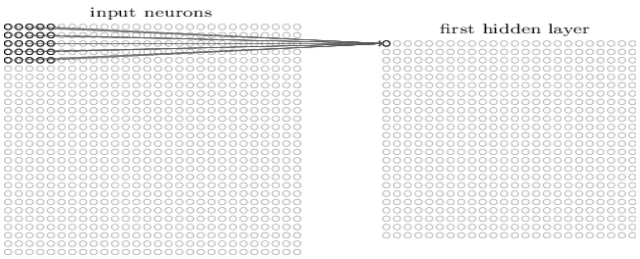

Fig 1. Connections between input layer and hidden layer

The region in the input image is called the local receptive field of hidden layer neurons. Each connection learns a weight.

In the next step of Shared weights and biases the main objective that it has a bias and weights connected to its local receptive field. A note worthy point is that we are going to use the same weights and bias for each of 24 by 24 hidden neurons.To make it in simple terms is that all the neurons in the first hidden layer detect exactly the same feature but at different places of the input image as the local receptive field moves through the input . To make it sensible , suppose the weights and bias are in such a way that hidden layer can predict a vertical edge in a particular local receptive field, this prediction can be useful at other parts of image. To put in practical terms convolutional neural networks are well habituated to invariance of images. In our implementation we are going to use MNIST datasets has less invariance compared to other images. So sometimes we call this mapping from input layer to hidden layer as the feature map. We define weights as shared weights and bias for knowing the feature as the shared bias , both often termed as kernel or filter.

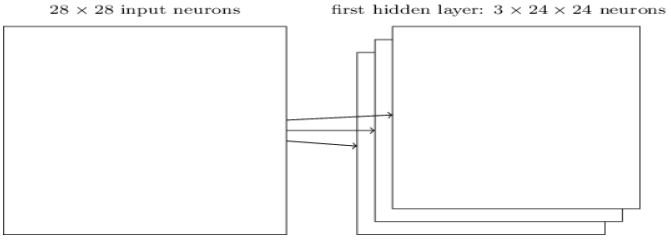

Fig 2. Showing different feature maps from input layer

In the above figure showing 3 feature mappings defined by a 5 by 5 shared weights and single bias per feature map. Now our network has the ability to detect 3 different kinds of features ,with each feature can be predicted in every part of image.

The next processing step used is termed as Pooling layers which are also a part of hidden layer and present after the Convolutional layer and gives more finer details of the feature mapped . Pooling layers simplifies the information from the output of convolutional layer and makes a very thin and condensed feature map which has ability to predict more thinner and finer details for feature extraction. These features can be again predicted at every place of image. To explain in practical terms each unit of pooling layer may predict a 2 by 2 neurons from the convolutional layer. The procedure used for pooling is coined as Max-pooling. We can combine these layers together using models of keras library ,but it has a additional of a layer of 10 classes of neurons representing the 10 possible values for MNIST digits.

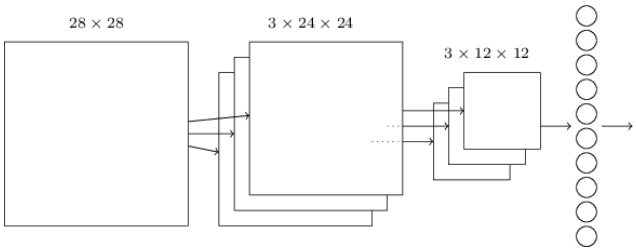

Fig 3. Network showing 28 by 28 input neurons , followed by convolutional layer and then pooling layer which is then connected to output layer.

In the above figure the network takes 28 by 28 square matrix of neurons which are used to encode the pixel intensities.Then it is followed by a convolutional layer of 5 by 5 local receptive field and containing 3 feature maps. The result is passed is passed onto pooling layers , which takes 2 by 2 regions as kernals. The output layer is a fully connected layer , in this layer every neuron from the pooled layer connected to every one of 10 neurons in output layer.

3.2 LeNet Architecture of CNN'S

In our implementation of Convolutional networks we will be using Lenet engineering which is primarily based on local receptive fields and kernels and sub-sampling or pooling to make sure that the invariance of several distortions possible. A typical convolutional neural network for classifying and recognizing digits as shown in figure 4 . The input layer gets the input images from MNIST dataset which can be downloaded by libraries available. They are normalized and centralized . With local receptive fields neurons can learn different vision features such as end-points,oriented edges,curves .These features are further extracted with more accurate by subsequent layers.Units in a feature map are all derived to perform same operation on different places of image. A complete convolutional network comprises of several feature mappings with different weights and biases so that multiple features can be extracted at once and can be applied on every part of image. A practical example is the first layer of our LeNet architecture which is shown in figure 4. Units which are present in the first hidden layer of LeNet are arranged in 6 feature mappings in our network. Each unit takes a 5 by 5 local receptive filed or kernel from input layer. Hence each neuron in hidden layer has 25 different trainable variable equations which when applying learning algorithm can be learned and these variables can be optimized using cost function and activation function. It additionally has a trainable variant bias which is unique for every feature learned. The receptive fields which are contiguous in manner are formed by corresponding contiguous units in the input layer. The other feature maps in the layer learn different weights and biases and can be useful for prediction of other features. In our LeNet architecture of CNN's at each input layer locates six different types of features are being derived by six units at same location in feature maps. A step by step implementation of feature map would scan the given input image with a single unit that has a local receptive field and stores the data of unit at corresponding locations in feature maps. This operation is identical to a convolution , followed by an additional bias and activation function so the name convolutional neural network.The kernel which we use in the network is set of weights used by units in feature mapping in mathematical approach to convolution . An astonishing property is that the amount by which the input image is shifted , the same amount is shifted towards the output of feature map. This property ensures that our network is invariant to shifts and disturbances of the input.

Once the feature is extracted , its exact position in image is not required , only the approximate location relative to other extracted features is important. For example in our analysis we came to know that the input image contains the endpoint of a horizontal line segment in upper left area , and the endpoint of a vertical line segment is located in the lower portion of image we can predict that image is a 7. But these precise positions of each of their features are required because the positions are likely to vary for different instances of the input digit. The only way to deal with this problem is by reducing the spatial resolution of feature map. This is implemented with sub sampling layers which performs a local averaging and a sub-sampling and therefore reducing the resolution of feature map. The second hidden layer of LeNet architecture is a sub-sampling layer . This layer compromises of distinct six feature maps one from every feature map in the previous layer. The receptive field for each unit compromises of 2 by 2 kernel in previous layer feature map. Each unit performs a typical operation like convolution which is averaging of its four inputs and multiplied by trainable weight's matrix ,adds a trainable bias and passes through the activation function like sigmoid

function , in our implementation we will be using ReLu activation function which is used in rectifiers. By practical approach a sub-sampling layer feature map has half number of rows and coloums as the feature maps in our convolutional layers had. The trainable coefficient and bias have control over the effect of activation function. If the coefficient is small , the unit operates in a quasi-linear mode and sub sampling layer merely blurs the image. Successive layers of convolution and sampling if performed on the imaged results in a bi-pyramid at each layer. A higher degree of invariance can be obtained with this successive reduction of spatial resolution of feature maps and also enables us to predict and detect more finer details and increase in the representation of data more richer. It makes easy for the network to identify features at very fast rate which makes CNN's more preffered to be used.

Since all weights are learned using learning algorithm , in our case we will be using back propogation , convolutional networks can be viewed as self extractor of its features. The weight sharing method has reduced memory to store and time to compute these weights.

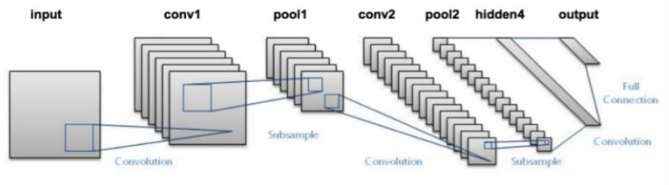

Fig 4. Architecture of LeNet Convolutional networks.

4. Methodology

We can implement Convolutional neural networks in PYTHON/MATLAB. In our implementation we will be using python because we have our keras deep learning library built in python . So by using keras models we can implement our network and create a driver program to call the network to take inputs from the data set. The driver program also has learning algorithm , training and testing datasets. MNIST datasets are the best and well known , and easily understood dataset in the computer vision branch and machine learning to use it as first dataset which we can use in our journey of deep learning. After implementing we can find that our network can classify the digits upto >98% accuracy with less training time. This implementation can be done in both CPU or GPU enabled system , but CPU takes more training time than GPU. We will be using 66% of data for training our network and rest of theb data to test our network. Each digit is taken as 28 by 28 greyscale image which are available from MNIST dataset and can be directly downloaded. These greyscale pixel intensities fall in the range of 0 to 255 . All digits are presented on black background colour with a light foreground colour being white, the digit itself and includes various shades of grey. The code is is organized this way, we will define a package name pyimagesearch and within that module we will create a cnn sub-module which will help us to store the Convolutional neural networks implementation. Then going into our cnn module or folder you should have your networks submodule , this is where the network implementations must be stored. So now create a python file inside that folder which implements our network and define a class inside our file which is our code implementing the LeNet architecture using Python + keras. Now we need a driver program to instantiate the LeNet architecture, train the model, load the datasets, and give the accuracy rate of our network and test the results. Finally the output folder we will store out LeNet model after it has been trained , so that it is not required to train the network after sub-sequent calls to classify digits.

4.1 Libraries required to install

Here we define our list of libraries we need to install for keras library to work for our networks. The most important library is the NUMPY is a library that provides support for large, multi-dimensional arrays where we can store our input pixel matrix of size 28 by 28 , using numpy we can express images as multi-dimensional arrays of pixel intensity values. We can also rely on the NumPy's built-in highly advanced mathematical functions and we can apply logistic regression on the image. The next library which is to be installed is the Python SCIPY library. It adds futher help for scientific and technical computing of our functions. The important subpackage of SciPy is the package that has a huge amount of distance funtions which are implemented using trees. Normally after extracting features the image is represented as a list of numbers , in order to compare these two images we need distance computation methods , such as Euclidean distance.. Next up is PILLOW library useful for manipulations on image such as resizing, rotation. Then we come to OPENCV library and the main goal of this library is real-time image processing. Next we can install SCIKIT-LEARN library which is by the way not a image vision library but a machine learning library.This library helps us with advanced computer vision whether it may be in clustering, quantization, classification models. The library next to be installed is h5py to store large numerical datasets, it also provides support for NumPy arrays it has efficient and long term storage of NumPy arrays.

4.2 Implementation of our LeNet architecture

After installing these libraries in python we can use our keras deep learning library to implement our network and create python files for network creation and instantiation. In our implementation we have trained the network in such a way that it learns many filters of size 5 by 5 and then pass it through a ReLU activation function followed by 2 by 2 max pooling in both dimensions. We then take the output of Max-pooling layers to apply it to fully connected layers. Our fully connected layers contain around 500 units which we will pass through another ReLU activation that enables us to

combine them into classes , which are useful for identifying our image , we have 10 classes one for each digit to classify. Finally we apply a softmax classifier that will give us the list of probabilities , one for each of the 10 classes created . The class label with largest probability will be taken as the final classification of our network. In our driver program datasets can be downloaded from mldata submodule and load it to our network , then train the network using Stocastic Gradient Descent with learning rate and number of iteration. Now we atre running keras on the top of TENSOR FLOW as backend to train the network. In our implementation we have given 20 epochs for better accuracy

5. RESULTS AND DISCUSSION

Our network has been trained with and tested with 70,000 datasets and we obtained an accuracy of > 98% which is good enough to test our classification implementation. We have given a learning rate of 0.01 to our algorithm and obtained good classification results , everytime after training we are taking random inputs from our testing dataset and calculating the efficiency each time it is excecuted. An interesting property of these networks is that the training error keeps decreasing over time but the test error goes through a minimum and starts increasing after a certain number of iterations , this is possibly because of the higher learning rate and by decreasing it we can get our results , if not reduced the learning rate the Stocastic gradient descent may get stuck in local minimum and finds it difficult to predict the optimized weights , which affects the prediction and accuracy of our network. The figure 5 shows how they change when the learning rate is high.

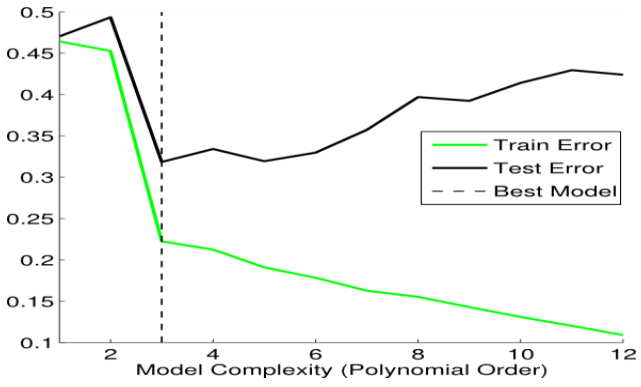

Fig 5. Showing how the test and train error changes and predicting the best model to be used.

In our discussion we can refer to other methods and their accuracy although all methods did well with all the classifiers, boosted LeNet 4 did best , achieving a score of 0.7% and rest of them acquired better accuracy than other methods . So it us best to rely on LeNet architecture rather than other methods for classification.

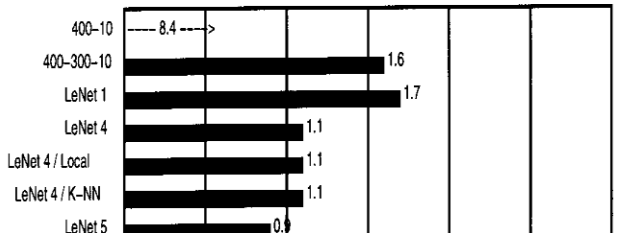

Fig 6. Showing the raw error rate of the classifiers on the 10,000 example test set.

6. CONCLUSION

Performance of a network depends on many factors like low memory requirements, low run time and better accuracy , although in this paper it is primarily focused on getting better accuracy rate for classification . Before Artificial neurons had better accuracy but now the branch of computer vision mainly depends on deep learning features like convolutional neural networks. Research is still going on in this field and researches have developed many forms of LeNet architecture like LeNet-1,LeNet-4, Boosted LeNet-4 and also combination of many methods like LeNet-4 with KNN's , but for a quite long time our LeNet architecture was considered as state of the art.Many other methods like Tangent Distance Classifier were developed using LeNet architecture. The main aim of this paper deals with one of the method in which it can be implemented , there are several methods in which they can be done and using different frameworks like matlab,octave. The branch of computer vision in artificial intelligence primary motive is to develop a network which is better to every performance measure and provide results for all kinds of datasets which can be trained and trained and recognized.

7. FUTURE WORK

Fixed size Convolutional Neural Networks has been applied to many applications like handwritten digit recognition , machine printed character recognition and on-line handwriting recognition, they can also be useful for signature verification .The more the training examples the more is the accuracy of the networks .Unsupersived machine learning was made easier using Convolutional Neural networks , some of the future works possible to implement by CNN's are compressing or obtaining same results from smaller networks by optimization tricks , more invariant feature learning such that the input images dosen't gets distorted. The major 3D vision networks is a scope for researches to develop using LeNet architecture and more biologically concordant methods , a hope for future is that Unsupervised CNN's .

8. REFERENCES

[1] Yann LeCun, Leon Bottou,Yoshua Bengio and Patrick Haffner , paper on "Gradient Based Learning Applied

to Document Recognition" , Proc of the IEEE , NOVEMBER 1998

[2] Y.LeCun,L.Jackel,L.bottom,A.brunot,C.Cortes,J.Denker ,H.Drucker,I.guyon,U.muller paper on "Comparision of Learning Algorithms for handwritten digit recognition"

[3] Y.LeCun,B.Baser,J.S.Denker,D.Henderson ,R.E.Howard,R.Hubbard,and L.D.Jackel , Handwritten digit recognition with a back- propogation network in D.Tourezky, Advances in Neural Information Processing Systems 2, Morgan Kaufman(1990)

[4] Corima Cartes and Vladimir Vapnik The Soft Margin Classifier , Machine Learning to appear(1995)

[5] Haider A.Alwzwazy, Hayder M. Albehadili, Younes S.Alwan ,Naz E Islam paper on "Handwritten Digit Recognition Using Convolutional Neural Networks" Vol 4 ,Issue 2,February 2016

[6] Xuan Yang , Jing Pu paper on "Mdig:Multi-digit Recognition using Convolutional Neural Network on Mobile

[7] Saeed Al-Mansoori paper on "Intelligent Digit Recognition using Artificial Neural Networks Vol 5, Issue 5, (Part-3) May 2015 , pp 46-51