COMPUTER VISION PROJECT REPORT

On

Handwritten digit classification using MNIST

BY

Suryaveer (150002037)

Ankit Gaur (150002007)

DISCIPLINE OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

May 2019

Handwritten digit classification using MNIST

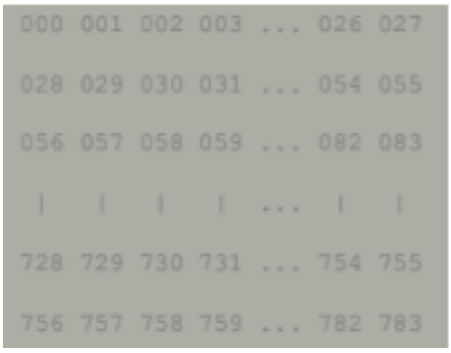
The goal of this project is to take an image of a handwritten single digit, and determine what that digit is.

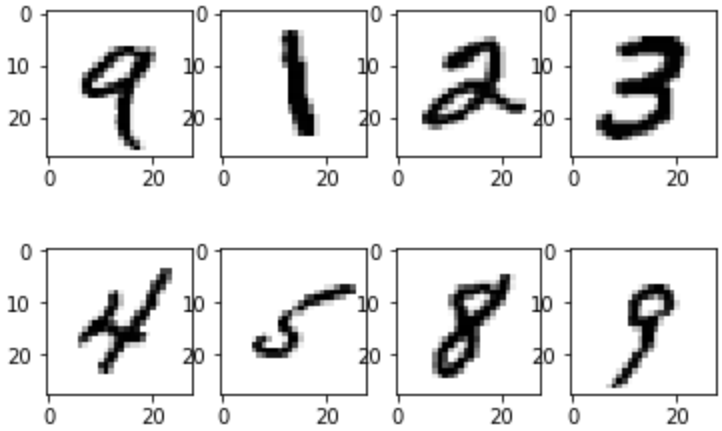
Introduction:

Recently Deep Convolutional Neural Networks (CNNs) becomes one of the most appealing approaches and has been a crucial factor in the variety of recent success and challenging machine learning applications such as object detection, and face recognition. Therefore, CNNs is considered our main model for our challenging tasks of image classification. Specifically, it is used for is one of high research and business transactions. Handwriting digit recognition application is used in different tasks of our real-life time purposes. Precisely, it is used in vehicle number plate detection, banks for reading checks, post offices for sorting letter, and many other related tasks.

Dataset:

The data files train.csv and test.csv contain gray-scale images of hand-drawn digits, from zero through nine.Each image is 28 pixels in height and 28 pixels in width, for a total of 784 pixels in total. Each pixel has a single pixel-value associated with it, indicating the lightness or darkness of that pixel, with higher numbers meaning darker. This pixel-value is an integer between 0 and 255, Inclusive. The training data set, (train.csv), has 784 columns. The first column, called "label", is the digit that was drawn by the user. The rest of the columns contain the pixel-values of the associated image. Each pixel column in the training set has a name like pixelx, where x is an integer between 0 and 783, inclusive. To locate this pixel on the image, suppose that we have decomposed x as x = i 28 + j, where i and j are integers between 0 and 27, inclusive. Then pixelx is located on row i and column j of a 28 x 28 matrix, (indexing by zero).
For example, pixel-31 indicates the pixel that is in the fourth column from the left, and the second row from the top, as in the ascii-diagram below.
Visually, if we omit the "pixel" prefix, the pixels make up the image like this:





Method:

Convolutional Neural Networks:

Work done by Hubel and Wiesel laid foundation to the invention of convolutional neural networks (CNNs). They have studied the visual cortex responses of cat and monkey. They observed that these neurons respond individually to small regions of visual fields. Provided the eyes are not moving, the region of visual space within which visual stimuli affect the firing of a single neuron is known as its receptive field. Neighboring cells have similar and overlapping receptive fields.

Broadly, architecture of CNN consist of an input layer, an output layer, and variable number of hidden layers. These hidden layers contain convolutional layers, activation function (non-linearity layer), pooling layers, fully connected layers and normalization layers.

Convolutional networks (CNN) are a special type of neural network that are especially well adapted to computer vision applications because of their ability to hierarchically abstract representations with local operations. There are two key design ideas driving the success of convolutional architectures in computer vision. First, CNN take

advantage of the 2D structure of images and the fact that pixels within a neighborhood are usually highly correlated. Therefore, CNN eschew the use of one-to-one connections between all pixel units (i.e. as is the case of most neural networks) in favor of using grouped local connections. Further, ConvNet architectures rely on feature sharing and each channel (or output feature map) is thereby generated from convolution with the same filter at all locations. This important characteristic of CNN leads to an architecture that relies on far fewer parameters compared to standard Neural Networks. Second, CNN also introduce a pooling step that provides a degree of translation invariance making the architecture less affected by small variations in position. Notably, pooling also allows the network to gradually see larger portions of the input thanks to an increased size of the network's receptive field. The increase in receptive field size (coupled with a decrease in the input's resolution) allows the network to represent more abstract characteristics of the input as the network's depth increase. For example, for the task of object recognition, it is advocated that CNN layers start by focusing on edges to parts of the object to finally cover the entire object at higher layers in the hierarchy.

CNN is now the go-to model on every image related problem. In terms of accuracy they blow competition out of the water. It is also successfully applied to recommender systems, natural language processing and more. The main advantage of CNN compared to its predecessors is that it automatically detects the important features without any human supervision. For example, given many pictures of cats and dogs it learns distinctive features for each class by itself.
CNN is also computationally efficient. It uses special convolution and pooling operations and performs parameter sharing as mentioned above. This enables CNN models to run on any device, making them universally attractive.

Architecture

Convolutional Layers

Convolution is a mathematical operation to merge two sets of information. The convolution is applied on the input data using a *convolution filter* to produce a *feature map*.

Activation Function

This layer contains blocks which apply different activation functions also called Transfer functions. Activation functions are of two types, namely: 1. Linear activation function and 2. Nonlinear activation function. It is used to determine the output of neural network like yes or no. It maps the resulting values in between 0 to 1 or -1 to 1 etc. (depending upon the function). Nonlinear activation functions can be, Sigmoid or Logistic, tanh, ReLU (Rectified Linear Units) etc[cite:Applications of CNN ijcsit journal].

> **Sigmoid function**: The range of *sigmoid* function(1) is 0 to 1 inclusive of both numbers (Fig. 3.2). This function comes in handy when we have to predict

probability as output. In such cases, since probability also has the same range, sigmoid can be a better pick.

$$\phi(z) = \frac{1}{(1+e^{-z})} \qquad (1)$$

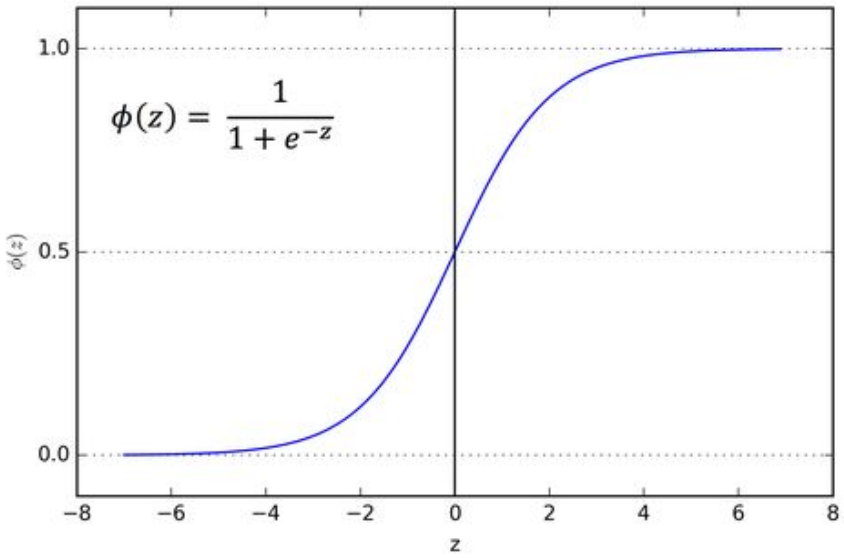


Fig. 3.2 : Sigmoid Function graph

Tanh or Hyperbolic tangent: The range of *tanh* function is -1 to 1 inclusive of both numbers. Tanh (Fig. 3.3) is also like sigmoid function but a little better than sigmoid. The advantage is, the negative inputs will be mapped strongly negative and zero inputs will be mapped near zero in tanh graph. This function is used

mainly for binary classification.

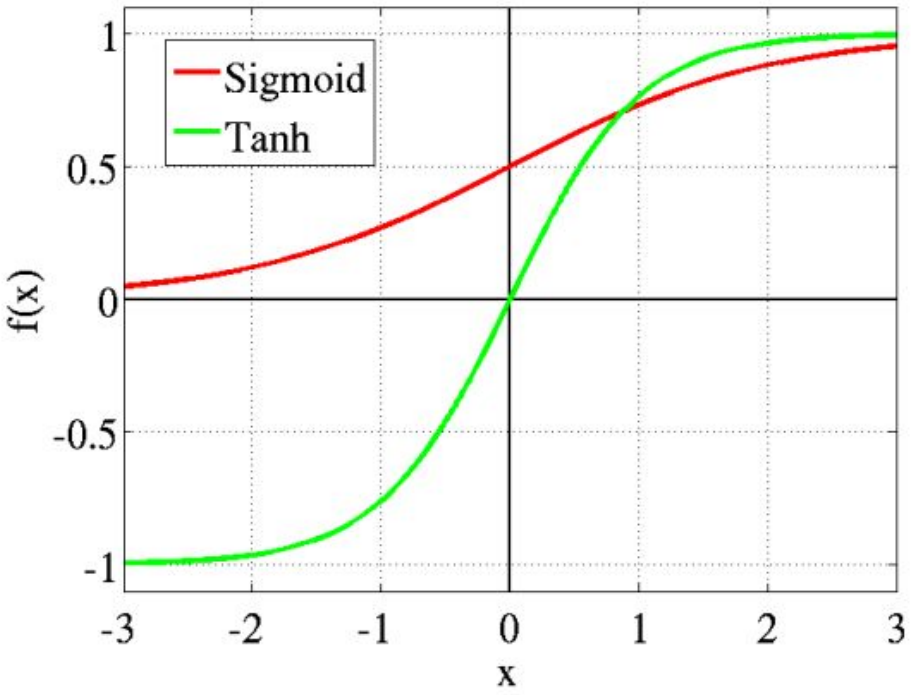


Fig. 3.3 : Sigmoid versus Tanh function graphs.

Rectified Linear Units(ReLU): It is the most widely used activation function. It is the used in almost all the convolutional neural networks and deep learning tasks. The range of this function is from 0 to ∞. The function graph is shown in Fig. 3.4.

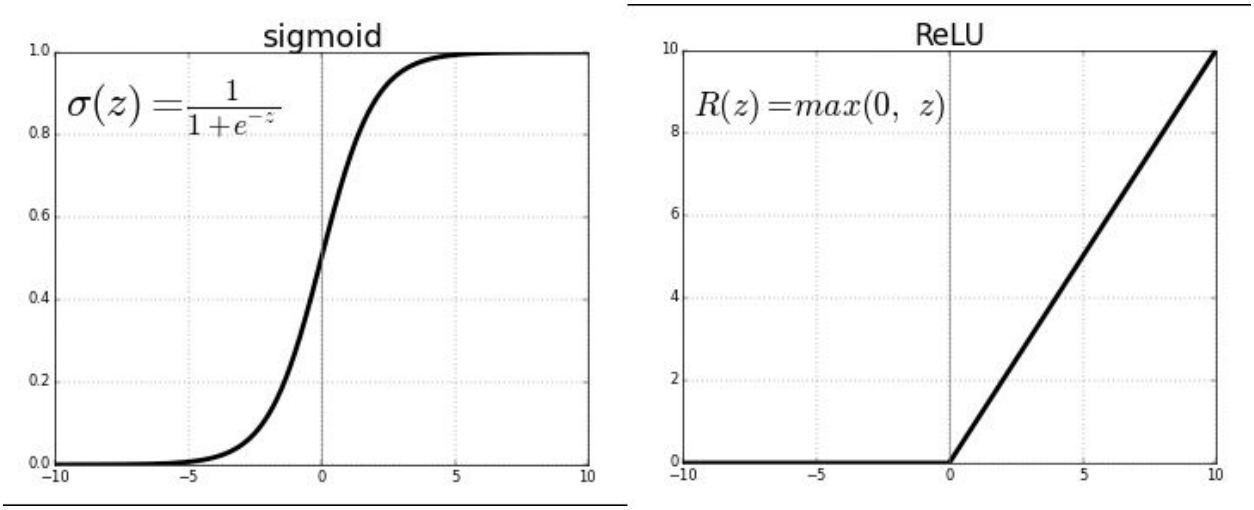


Fig. 3.4 : Sigmoid versus ReLU function graphs.

Pooling

After a convolution operation we usually perform *pooling* to reduce the dimensionality. This enables us to reduce the number of parameters, which both shortens the training time and combats overfitting. Pooling layers downsample each feature map independently, reducing the height and width, keeping the depth intact. The most common type of pooling is *max pooling* which just takes the max value in the pooling window. Contrary to the convolution operation, pooling has no parameters. It slides a window over its input, and simply takes the max value in the window.

Fully connected

Fully connected layers connect every neuron in one layer to every neuron in another layer. It is in principle the same as the traditional multi-layer perceptron neural network (MLP).

Receptive field

In neural networks, each neuron receives input from some number of locations in the previous layer. In a fully connected layer, each neuron receives input from *every* element of the previous layer. In a convolutional layer, neurons receive input from only a restricted subarea of the previous layer. Typically the subarea is of a square shape (e.g., size 5 by 5). The input area of a neuron is called its *receptive field*. So, in a fully connected layer, the receptive field is the entire previous layer. In a convolutional layer, the receptive area is smaller than the entire previous layer.

Weights

Each neuron in a neural network computes an output value by applying some function to the input values coming from the receptive field in the previous layer. The function that is applied to the input values is specified by a vector of weights and a bias (typically real numbers). Learning in a neural network progresses by making incremental adjustments to the biases and weights. The vector of weights and the bias are called a *filter* and represents some feature of the input (e.g., a particular shape). A distinguishing feature of CNNs is that many neurons share the same filter. This reduces the amount of memory a program uses or references while running called as memory footprint because a single bias and a single vector of weights is used across all receptive fields sharing that filter, rather than each receptive field having its own bias and vector of weights.

Applications of CNNs

A. Computer Vision:

Convolutional neural networks are employed to identify the hierarchy or conceptual structure of an image. Instead of feeding each image into the neural network as one grid of numbers, the image is broken down into

overlapping image tiles that are each fed into a small neural network. Convolutional neural networks are trainable multi-stage architectures, with the inputs and outputs of each stage consisting of sets of arrays called feature maps. If the input is a colour image, each feature map is a 2D array containing a colour channel of the input image, for a video or a volumetric image it would be a 3D array. Each feature extracted at all locations on the input is represented by a feature map at the output. Each stage is composed of a filter bank layer, a non-linearity layer and a feature pooling layer. A typical CNN is composed of one, two or three such 3-layer stages, followed by a classification module.

a. *Face recognition:* Face recognition constitutes a series of related problems-
    - Identifying all the faces in the picture
    - Focussing on each face despite bad lighting or different pose
    - Identifying unique features
    - Comparing identified features to existing database and determining the person's name

    Faces represent a complex, multi-dimensional, visual
    stimulus which was earlier presented using a hybrid neural network combining local image sampling, a self-organizing map neural network and a convolutional neural network. The results were presented using Karhunen-Loe`ve transform in place of the self-organizing map which performed almost as well (5.3% error versus 3.8%) and a multi-layer perceptron which performed poorly (40% error versus 3.8%).

b. *Scene Labelling:* Each pixel is labelled with the category of the object it belongs to in scene labelling. Clement Farabet et al proposed a method using a multiscale convolutional network that yielded record accuracies on the Sift Flow Dataset (33 classes) and the Barcelona Dataset (170 classes) and near-record accuracy on Stanford Background Dataset (8 classes). Their method produced 320 X 240 image labelling in under a second including feature extraction.
c. *Image Classification:* Compared with other methods CNNs achieve better classification accuracy on large scale datasets due to their capability of joint feature and classifier learning. Krizhevsky et al. develop the AlexNet and achieve the best performance in ILSVRC 2012. Following the success of the AlexNet, several works made significant improvements in classification accuracy by reducing filter size or expanding the network depth. A fast, fully parameterizable GPU implementation of CNN published benchmark results for object classification (NORB, CIFAR10) with error rates of 2.53%, 19.51%. GPU code for image classification is upto two magnitudes faster than its CPU counterpart. Multi-column deep neural networks(MCDNN) can outperform all previous methods of image classification and demonstrate that pre-training is not necessary(though

sometimes beneficial for small datasets) while decreasing the error rate by 30-40%. Non-saturating neurons and efficient GPU implementation of the convolution operation resulted in a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry in the ILSVRC-2012 competition for classification of 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. Hierarchical Deep Convolutional neural Networks (HD-CNN) are based on the intuition that some classes in image classification are more confusing than other classes. It builds on the conventional CNNs which are N-way classifiers and follows the coarse-to-fine classification strategy and design module. HD-CNN with CIFAR100-NIN building block is seen to show a testing accuracy of 65.33% which is higher than the accuracy for other standard deep models and HD-CNN models on CIFAR100 dataset.

d. *Document Analysis:* Many traditional handwriting recognizers use the sequential nature of the pen trajectory by representing the input in the time domain. However, these representations tend to be sensitive to stroke order, writing speed and other irrelevant parameters. This system AMAP preserves the pictorial nature of the handwriting images. This model uses a combination of CNNS, Markov models, EM algorithm and encoding of words into low resolution images to recognize handwriting. Using this system error rates dropped to 4.6% and 2.0% from 5% for word and character errors respectively. This model was implemented using two different practices: a database was expanded by adding a new generic collection of elastic distortions. Convolutional neural networks were used. The highest performance was achieved on the MNSIT data set using elastic distortion and convolutional neural networks. As compared to the 2 layer Multi-layer perceptron models error rate of 1.6% this model achieves an error rate of 0.4% , thus significantly improving the performance. This is because MCP suffered from the problem of inverse problem or transformation invariance. Though OCR and other systems have already reached high recognition rates for printed documents recognition of characters in natural scene images is a challenging task for the system because of complex background, low resolution, non-uniform light etc. Thus this model proposes complex colour scene text image recognition, based on specific convolution neural network architecture. Using this system the average recognition rate is found to be an improved 84.53% ranging from 93.47% for clearer images and 67.86% for seriously distorted images using the ICDAR 2003 dataset.

Procedure:

MNIST is a widely used dataset for the hand-written digit classification task. It consists of 70,000 labelled 28x28 pixel grayscale images of hand-written digits. The dataset is split into 60,000 training images and 10,000 test images. There are 10 classes (one for each of the 10 digits). The task at hand is to train a model using the 60,000 training images and subsequently test its classification accuracy on the 10,000 test images.

We converted raw format data to grayscale image to be used in 2d Convolutional Neural Network. After that we created a cnn model as explained below:

Python Application for Real Time Digit Recognition using Pygame and OpenCV libraries

We developed an application in python using pygame library which facilitates user to write numbers on the screen. Working of the application is given below:

1. User is given a pygame window in which he/she can use mouse cursor to write numbers on the screen.
2. On regular interval window image is taken and using opencv library and contour detection method areas of interest are selected.
3. Selected areas of interest then are converted into the format which is compatible with the cnn based model.
4. Output from the cnn based model is displayed on the pygame window.

PyGame Window

PyGame Window User Input and Output from Model

Results:

Validation accuracy: 0.992

Confusion matrix

0	0	1	2	3	4	5	6	7	8	9
0	975	0	1	0	0	0	2	0	1	1
1	0	1132	1	0	0	0	0	2	0	0
2	1	1	1022	0	1	0	0	5	2	0
3	0	0	0	1006	0	3	0	0	0	1
4	0	0	0	0	980	0	0	0	1	1
5	0	0	0	2	0	889	1	0	0	0
6	2	2	0	0	1	1	952	0	0	0
7	0	2	1	1	0	0	0	1021	1	2
8	3	0	2	1	0	2	1	2	962	1
9	1	2	0	0	10	5	0	7	3	981

Result Statistics for Different Digits

In a confusion matrix for a class x:

- True positive: diagonal position, cm(x, x).
- False positive: sum of column x (without main diagonal)
- False negative: sum of row x (without main diagonal)

Precision = TP/(TP+FP)
Recall = TP/(TP+FN)
Accuracy = TP/TP+FP+FN

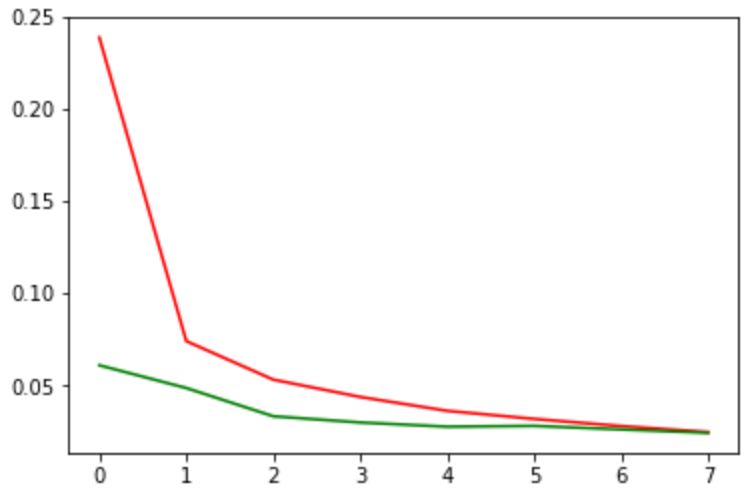
F1 Score = 2*Precision*Recall/(Precision + Recall)

Digit	0	1	2	3	4	5	6	7	8	9
TP	975	1132	1022	1006	980	889	952	1021	962	981
FP	7	7	5	4	12	11	4	16	8	6
FN	5	3	10	4	2	3	6	7	12	28
Precision	0.9928	0.9938	0.9951	0.9960	0.9879	0.9878	0.9958	0.9845	0.9917	0.9939
Recall	0.9948	0.9973	0.9903	0.9960	0.9800	0.9966	0.9937	0.9932	0.9876	0.9722
F1 Score	0.9937	0.9955	0.9926	0.9960	0.9839	0.9921	0.9947	0.9888	0.9896	0.9829

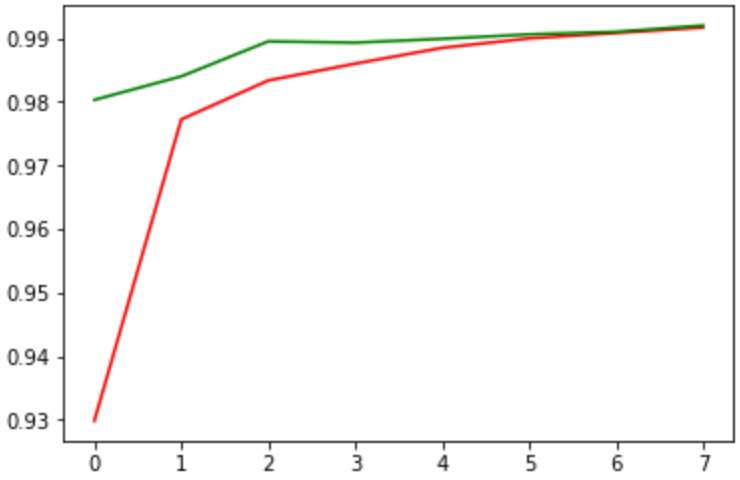
Testing Accuracy = 9920/10000 = 0.9920

Training Curves:

Loss with epoch:



Accuracy with epoch:



Testing window: