The Experiment Report of Machine Learning

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

Author: -AXEL MAKONDA MBUTA

-MADELEYN OPPORTO

-ABAS MOHAMMED

Supervisor:

Mingkui Tan

Grade: Graduate

Student ID：

201722800013; 201722800070 and 201722800094

December 28, 2017

HANDWRITTEN DIGIT RECOGNITION BASED ON SHALLOW NEURAL NETWORK

Abstract—

This experiment use handwritten digital data set - MNIST, which contains 60,000 hand-written digital images for training and 10,000 hand-written digital images for validation. Each image is 28 × 28 pixels in size. Operations such as downloading dataset and reading images are done using pytorch. In short, “shallow” neural networks are a term used to describe NN that usually has only one hidden layer as opposed to deep NN which have several hidden layers, often of various types. A shallow network has less number of hidden layers. While there are studies that a shallow network can fit any function, it will need to be really fat. That causes the number of parameters to increase a lot.

KEYWORDS: SHALLOW NEURAL NETWORK,

HANDWRITTEN DIGIT

I. INTRODUCTION

A popular demonstration of the capability of deep learning techniques is object recognition in image data. The MNIST problem is a dataset developed by Yann LeCun, Corinna Cortes and Christopher Burges for evaluating machine learning models on the handwritten digit classification problem. The dataset was constructed from a number of scanned document dataset available from the National Institute of Standards and Technology (NIST). This is where the name for the dataset comes from, as the Modified NIST or MNIST dataset. Images of digits were taken from a variety of scanned documents, normalized in size and centered. This makes it an excellent dataset for evaluating models, allowing the developer to focus on the machine learning with very little data cleaning or preparation required. Each image is a 28 by 28 pixel square (784 pixels total). A standard spit of the dataset is used to evaluate and compare models, where 60,000 images are used to train a model and a separate set of 10,000 images are used to test it.

It is a digit recognition task. As such there are 10 digits (0 to 9) or 10 classes to predict. Results are reported using prediction error, which is nothing more than the inverted classification accuracy. Learn deep neural network construction process and use; Understand deep learning framework pytorch and preliminary use; Experience neural network training and testing process; Deepen understanding of convolution, pooling, ReLu, fully connected and other network layers. All these points are considered as our experiment motivation. The technology is leaping into so much advancement that image recognition will become part and parcel of our daily lives.

II. METHODS AND THEORY

We can do any symbol recognition using this methodology. but for our project we only chose to do numerical digits 1,2,3,4,5,6,7,8,9 and 0. Different approaches have been used for handwritten recognition, feature extraction , by using Fourier transformation , using support vector machine (SVM) and using classifier . On

the contrary in this research hand written digit recognition is done through giving a cognitive thinking process to a machine by developing a neural network based AI engine, which recognizes any handwritten digit. The same technique can be further used in any application for signature verification or hand writing recognition or other bio metric applications. The idea of using Neural network based AI engine is unique, and simple to use. It only requires one time training of the neural network where as in cited methodologies whenever there is an image to process all steps are repeated again and again for image pre-processing which uses important cycle time and takes longer time intervals to recognize each handwritten digit. AI Engine based approach is customizable, and adaptable to be used for any generic image recognition application. i.e signature recognition, face recognition or thumb print recognition .let's have a look at some related work:

Shallow networks

In this section, we announce our results in the context of shallow networks in two settings. One is the setting of neural networks using the ReLU function x 7→ |x| = x+ + (−x)+ (Sub-section 3.1), and the other is the setting of Gaussian networks using an activation function of the form x 7→ exp(−|x − w| 2) (Sub-section 3.2). It is our objective to generalize these results to the case of deep networks in Section 4. 6 Before starting with the mathematical details, we would like to make some remarks regarding the results in this section and in Section 4. 1. It seems unnatural to restrict the range of the constituent functions. Therefore, we are interested in approximating functions on the entire Euclidean space. 2. If one is interested only in error estimates analogous to those in Theorem 2.1, then our results need to be applied to functions supported on the unit cube. One way to ensure that that the smoothness is preserved is to consider a smooth extension of the function on the unit cube to the Euclidean space [25, Chapter VI], and then multiply this extension by a C∞ function supported on [−2, 2]q , equal to 1 on the unit cube. However, this destroys the constructive nature of our theorems. 3. A problem of central importance in approximation theory is to determine what constitutes the right smoothness and the right measurement of complexity. The number of parameters or the number of non–linear units is not necessarily the right measurement for complexity. Likewise, the number of derivatives is not necessarily the right measure for smoothness for every approximation process. In this paper, we illustrate this by showing that different smoothness classes and notions of complexity lead to satisfactory approximation theorems.

ReLU networks

In this section, we are interested in approximating functions on R q by networks of the form x 7→ Pn k=1 ak|x·vk+bk|, ak, bk ∈ R, x, vk ∈ R q . The set of all such functions will be denoted by Rn,q. Obviously, these networks are not bounded on the whole Euclidean space. Therefore, we will study the approximation in weighted spaces, where the norm is defined by kfkw,q = ess sup x∈Rq |f(x)| p |x| 2 + 1 . The symbol Xw,q will denote the set of all continuous functions f : R q → R for which (|x| 2 + 1)−1/2f(x) → 0 as x → ∞. We will define a "differential operator" D and smoothness classes Ww,γ,q in terms of this operator in Section 5.1. In the sequel, we will adopt the following convention. The notation A. B means A ≤ cB for some generic positive constant c that may depend upon fixed parameters in the discussion, such as γ, q, but independent of the target function and the number of parameters in the approximating network. By A ∼ B, we mean A. B and B. A. Our first main theorem is the following Theorem 3.1. We note two technical novelties here. One is that the activation function | · | does not satisfy the conditions of Theorem 2.1. Second is that the approximation is taking place on the whole Euclidean space rather than on a cube as in Theorem 2.1. Theorem 3.1 Let γ > 0, n ≥ 1 be an integer, f ∈ Ww,γ,q. Then there exists P ∈ Rn,q such that kf − Pkw,q . n −γ/qkfkw,γ,q

III. EXPERIMENT

1. Build shallow neural network LeNet5 (A class inheriting from nn.Module. You are supposed to at least implement init and forward function) in model.py. Network structure is as follows: Use pytorch to build this network,
2. Complete the network training and validation process in train.py:
   2.1Usetorch.utils.data.DataLoadertorchvision.datasets.MNIST and torchvision.transforms to load dataset.
   2.2 Instantiate torch.optim.SGD to get the optimizer
   2.3 Instantiate LeNet5 to get net.
   2.4 Input data, which size is batch size, into the network for forward propagation to get predict target.
   2.5 Use torch.nn.CrossEntropyLoss to calculate the loss between predict target and ground truth target.

2.6 Calculate gradient using loss.backward and optimize the net using optimizer.step.
2.7 Calculate accuracy of recognition.
2.8 Repeat step 2.4 to 2.7 until all data are inputed to the net.
2.9 Define the number of selected training epoch, repeat the training process 2.5--2.9
2.10 Save the best model as a .pkl file

3. Compared with the training , there is no optimization process in validation. The other processes between training and validation are basically the same.
4. Load the trained model (.pkl file) into the main.py application. The implementation of main.py can test the model's performance on the web page.

IV. CONCLUSION

Excellent results achieve a prediction error of less than 1%. State-of-the-art prediction error of approximately 0.2% can be achieved with large Convolutional Neural Networks. There is a listing of the state-of-the-art results and links to the relevant papers on the MNIST and other datasets on Rodrigo Benenson's webpage. You can also get very good results using a very simple neural network model with a single hidden layer.

Neural Networks have really created a new vision in the computer and industrial applications. For future work we plan to use the same technique to identify signatures for processing cheques in banking Industry and secondly to develop a face recognition system for HRM Department for student attendance system based on Computer Vision.

Finally, the output variable is an integer from 0 to 9. This is a multi-class classification problem. As such, it is good practice to use a one hot encoding of the class values, transforming the vector of class integers into a binary matrix.