Handwritten Digits Recognition using Deep Convolutional Neural Network An Experimental Study using EBlearn

Karim M. Mahmoud

Computer and Systems Engineering Department
Faculty of Engineering, Alexandria University
Alexandria, Egypt

Abstract. In this paper, results of an experimental study of a deep convolutional neural network architecture which can classify different handwritten digits using EBLearn library are reported. The purpose of this neural network is to classify input images into 10 different classes or digits (0-9) and to explore new findings. The input dataset used consists of digits images of size 32X32 in grayscale (MNIST dataset).

Keywords: EBlearn, Convolution Neural Network, Handwritten Digits Classification

1 Convolution Neural Network Architecture

Fig. 1 shows the architecture of the proposed convNet which consists of 6 layers. The first layer C1 is a convolution layer with 6 feature maps and a 5x5 kernel for each feature map. The second layer S1 is a subsampling layer with 6 feature maps and a 2x2 kernel for each feature map. The third layer C3 is convolution layer with 16 feature maps and a 6x6 kernel for each feature map. The fourth layer S4 is a subsampling layer with 16 feature maps and a 2x2 kernel for each feature map. The fifth layer C5 a convolution layer with 120 feature maps and a 6x6 kernel for each feature map. The sixth layer f7 is a fully connected layer.

In this experimental study the MNIST dataset is trained with the above mentioned ConvNet architecture, using 2000 training samples and 1000 testing samples. The number of training iterations is set to 20. The learning rate used in the Backpropagation algorithm is 0.0001.

2 Results of Demo and Discussion

Experimental results are available publicly. From the results of this experimental study, it is obvious that as the number of training iterations increases, the accuracy rate increases and error rate decreases. For example, after 20 iterations the accuracy rate converges to 96.5% and error rate converges to 3.4%. (Starting

1 http://sites.google.com/site/convnetexp/)
the first training iteration with accuracy rate of about 90.9% and error rate of 9%.

Also from the detailed results, it is noticed that among the 10 digits/classes (0-9), the digit 1 achieved the highest accuracy rate 99.2% and lowest error rate 0.79%. Perhaps this is because the digit 1 usually has a fixed figure or image and dont have a weird or unusual figure as the other digits.

The following sections discuss tuning different configuration parameters used in the experimental study. Also more extra experiments have been tested to show the effect of changing these parameters on the error and accuracy rates.

2.1 Tuning learning rate (ETA)

Trying to test the effect of changing the learning rate (ETA) input on the results, the learning rate used with different values other than the original one used in this experiment (0.0001).

Test Case ETA = 0.01: Increasing the learning rate to 0.01 with 20 iterations, it is noticed that the error rate and accuracy rate stopped improving after the 3rd or 4th iteration. Moreover, the results became worse as the error rate increased and the accuracy rate was decreasing after the 4th iteration till the 6th iteration. Starting from the 7th iteration till 20th iteration, the learning process was saturated as error rate stopped at 90% and the accuracy rate stopped at 10%.

Test Case ETA = 0.00001: Decreasing the learning rate to 0.00001, it is noticed that the speed of learning became slower. After 20 iterations the error rate was around 4.6% and the accuracy rate was around 95%. To test the effect of increasing the number of iterations with this decreased learning rate value, the following case is tested, in this case the number of training iterations increased to be 50 instead of 20.
**Test Case ETA = 0.00001 with 50 iterations:** Increasing the number of training iterations to 50, and after about 32 iterations the rates became fixed to the following values: error rate around 4.5% and the accuracy rate around 95.5%.

As a conclusion, the learning rate value has a great effect on the speed of the learning process and the accuracy of the final results. The larger the learning rate, the larger the weight changes on each iteration, and the quicker the network learns. However, the size of the learning rate can also influence whether the network achieves a stable solution. If the learning rate gets too large, then the weight changes no longer approximate a gradient descent procedure. Selection of appropriate or optimal learning rate is important. One of the suggestions is to use adaptive learning rate value. Authors in [3] followed a heuristic which is dividing the learning rate by 10 when error rate stops improving with the current learning rate.

2.2 Type of Non-linearity

In these experiments, the tanh function is used, which is a saturating non-linearity. In terms of training time with gradient descent, this nonlinearity is much slower than the non-saturating nonlinearity function \( \max(0,x) \) as suggested by [3,4]. Since, this function is not implemented in the current eblearn library, it is of my future work interest to implement it in eblearn and to test the effect of using it instead of using tanh function.

Meanwhile, another function is tested which is abs (absolute value of input). The following are the test results: High error rate of around 89% and accuracy rate of 10.7% (poor results).

Another test case is using standard sigmoid function (stdsig). The following are the test results: error rate of around 4.5% and accuracy rate of 95.6%, which is almost the same accuracy rate as using tanh, but with higher error rate.

2.3 Momentum and Decay time

In this experiment, the momentum (inertia variable) and decay time (anneal period variable) of the Backpropagation learning algorithm are not used. In the following test case, I set the momentum to 0.9 and decay time to 0.0005 to test the effect, following are the results: Error rate = 2.7%, Accuracy rate = 97.3% (Better results)

If we compared those values with the first mentioned in this report (error rate = 3.4% and accuracy = 95.5%), It will be clear that adding momentum and weight decay values for the training algorithm, reduce the error rate and improves the results.

2.4 Regularization

Regularization is a process of adding more information in order to prevent overfitting. Setting regularization parameter to 0.00001 or 0.0001 or 0.001 in the
original configuration file, did not improve the results for perhaps due to the limited number of input samples and number of iterations as the overfitting problem occurs due to the large number of parameters.

2.5 Classification Test

As a part of this experimental study, the trained convNet can be used to classify a single input handwritten digit image. In the configuration, input classification is activated by setting parameter runtype to detect, feeding the network with the weight file of the 20th iteration, and calling detect program.

3 Conclusion

In this experimental study, after running many experiments and tuning some parameters, the best results (error rate = 2.7%, accuracy rate = 97.3%) have been achieved using the following input parameters (For time sake, demo uses only 2000 training samples, 1000 test samples, and limited to 20 training iterations): Learning rate: = 0.0001, Momentum: = 0.9, and decay time: = 0.0005. Detailed results are available at http://sites.google.com/site/convnetexp/

References

1. : Eblearn library: http://eblearn.cs.nyu.edu:21991/doku.php?id=start
2. : Mnist dataset: http://yann.lecun.com/exdb/mnist/
3. Krizhevsky, A., Sutskever, I., Hinton, G.: Imagenet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems 25 (2012)
4. Nair, V., Hinton, G.: Rectified linear units improve restricted boltzmann machines. In Proc. 27th International Conference on Machine Learning (2010)