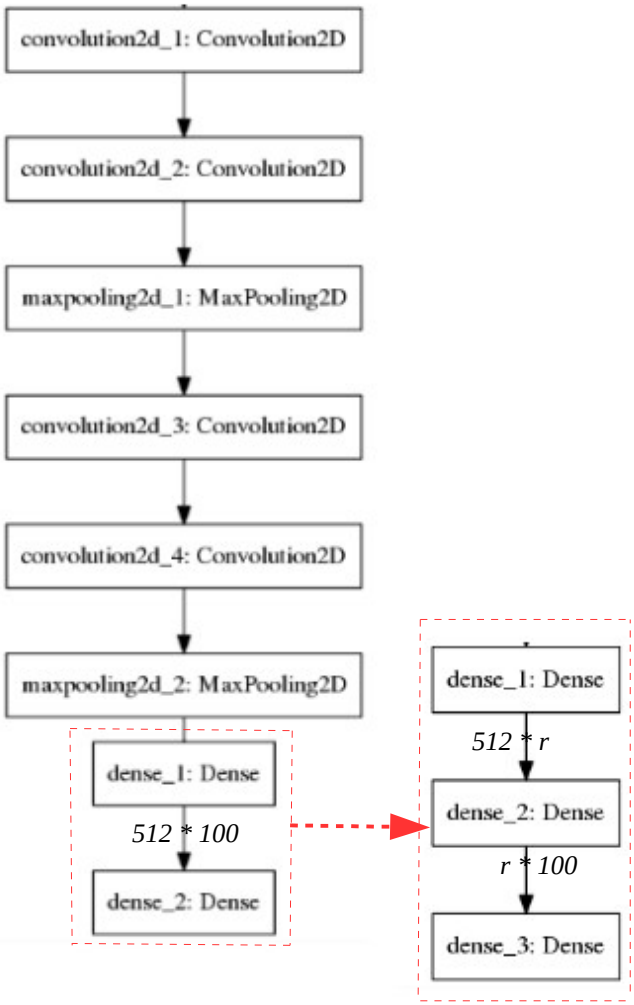
Low Rank Matrix Factorization for DNNs

Introduction

Despite Deep Neural Networks (DNNs) have achieved a great success in various applications, the training process of these models are slow because of the high number of model parameters. To achieve good performance with DNNs we usually use a large number of output targets, so many of these parameters are in the final layer. Furthermore, few output targets are actually active for a given input, and most of the time these active outputs are correlated. And this suggests that the last weight layer matrix is low level. So, we can represent last layer matrix by two small matrices and reducing the number of training parameters. So in this project, I demonstrate using Matrix Factorization to reduce the number of parameters in the final layer by a percentage around 40% without significant loss of accuracy. The low rank network has an extra matrix multiplication, so if we need to make use of the parameters reduction we need to parallelize the matrix multiplications in order to reduce the network training time to a percentage similar to the the parameters reduction percentage. The demonstration is carried for two image classification task with 10 and 100 target classes with MNIST and CIFAR100 dataset and the implementation is done using Keras.

Low Rank Matrix Factorization

Given that only few output targets are really active and this makes the last weight layer matrix a low rank matrix. We can represent the large matrix by two small ones. Let us donate the last weight matrix as matrix *A* with size of *m*n*, we can replace it with small two matrices *B* and *C* with size *m*r* and *n*100*. The right-hand side of Figure 1 illustrates replacing the weight matrix in the last weight layer by two matrices one of size *512*r* and one of size *r*100*. We can reduce the number of parameters of the system so long as the number of parameters in B and C is less than A. If we would like to reduce the number of parameters in A by a fraction p, we require the following to hold: $mr + rn < pmn$. Solving the equation for r in gives the following requirement needed to reduce overall parameters by fraction p: $r < (pmn) / (m+n)$.



Experiments

The initial experiments are conducted on MNIST dataset for digit recognition task. A neural network with 2 conventional layers with 32 filters 3*3 each, 1 max pooling layer 2*2 pool size, 1 fully connected layer with 256 units, and finally, an output layer with 10 softmax units. And for training, *adam* optimizer was used with 32 batch size and 10 epochs.

rank	error rate	# of parameters (reduction %)	execution time (seconds)
full rank	0.99 %	19,962	1170
8	1.18 %	19,536 (2.5%)	926
6	1.01 %	19,004 (5.0%)	885
4	1.14 %	18,472 (7.5%)	893
2	6.94 %	17,940 (10%)	875
1	7.33 %	17,674 (12.5%)	815

Table 1: Different Choices of Rank r on Softmax Layer

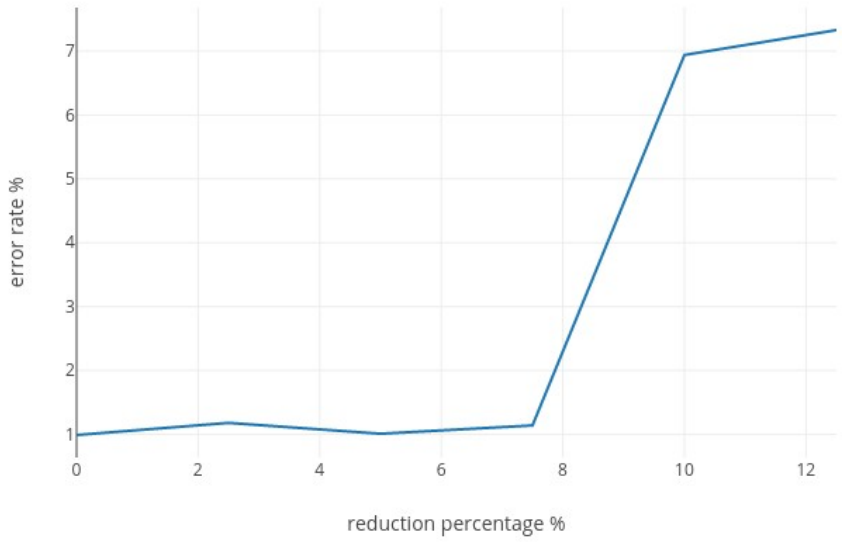


Illustration 1: Spectral Norm between Full and Rank r-Matrix

Analysis

Referring to the results in Table 1 and the Illustration 1, we can see that reducing the rank increases the error rate not so much until rank 4 then the error rate starts to increase more rapidly. The best rank from the results is rank 4 because we reduce the parameters up to 7.5% and 60% of the last layer parameters and still get just 0.15% error more than the full rank and this suggests that the softmax layer has only 4 uncorrelated active outputs. The execution time is improved by about 20% and this percentage can be further improved by parallelizing the matrices multiplications.

Further experiments are conducted to test the method generalization to a relatively larger scale task. In this task, I test the method on CIFAR100 dataset for an image classification task with 100 output targets.

rank	error rate	# of parameters (reduction %)	execution time (seconds)
full rank	56.38 %	148,480	8,174
75	58.11 %	143,180 (3.5%)	7,860
60	59.04 %	134,000 (6.5%)	7,656
40	61.44 %	121,760 (18%)	7,693
20	66.10 %	109,520 (26%)	7,788
10	75.62 %	103,400 (30%)	7,750
1	96.97 %	97,872 (35%)	7,668

Table 2: Different Choices of Rank r on Softmax Layer

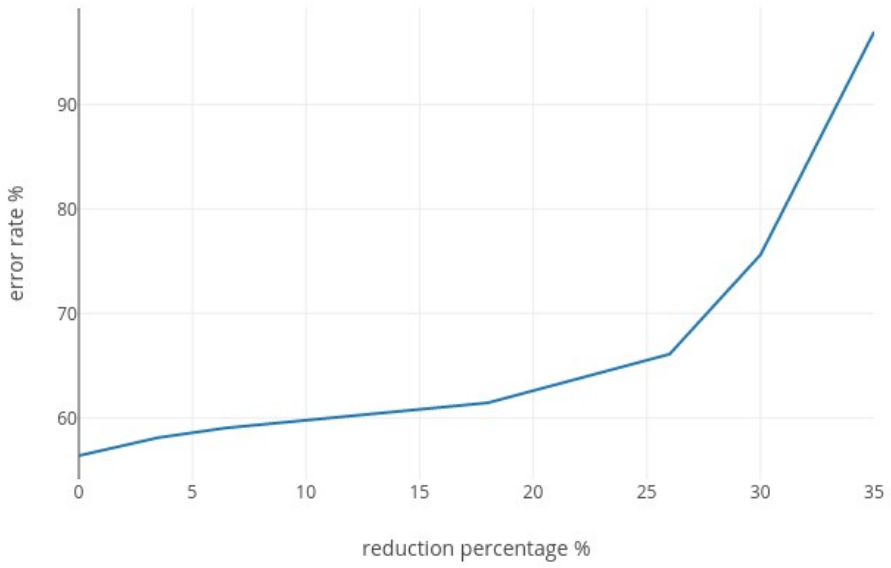


Illustration 2: Spectral Norm between Full and Rank r-Matrix

Analysis

Referring to the results in Table 2 and the Illustration 2, we can see that reducing the rank increases the error rate not so much until rank 60 then the error rate starts to increase more rapidly. The best rank from the results is rank 60 because we reduce the parameters up to 6.5% and nearly 30% of the last layer parameters and still get just 2.66% error more than the full rank and this suggests that the softmax layer has nearly only 60 uncorrelated active outputs. The execution time is improved by about 7% and this percentage also can be further improved by parallelizing the matrices multiplications.

Bibliography

[1] *LOW-RANK MATRIX FACTORIZATION FOR DEEP NEURAL NETWORK TRAINING WITH HIGH-DIMENSIONAL OUTPUT TARGETS: Tara N. Sainath, Brian Kingsbury, Vikas Sindhwani, Ebru Arisoy, Bhuvana Ramabhadran, IBM T. J. Watson Research Center, Yorktown Heights, NY 10598.*