A predictor-corrector method for the training of deep neural networks

Yatin Saraiya

847 Moana Court, Palo Alto, CA 94306, USA

Abstract

The training of deep neural nets is expensive. We present a *predictor-corrector* method for the training of deep neural nets. It alternates a predictor pass with a corrector pass using stochastic gradient descent with backpropagation such that there is no loss in validation accuracy. No special modifications to SGD with backpropagation is required by this methodology. Our experiments showed a time improvement of 9% on the CIFAR-10 dataset.

Keywords: Predictor, corrector, deep, neural, network

1. Introduction

Image recognition is performed mainly by deep convolutional neural networks [6, 7], and the depth of the network is crucial to accurate results [8]. However, very deep neural nets degrade near convergence unless some form of shortcutting [11] is used. We choose the deep residual nets of [1] as the basis for our work. These networks consist of stacked blocks with the same number of inputs as outputs[1]. The addition of a residual identity mapping counters the degradation problem.

2. Methodology

Assumption 1. Our basic hypothesis is that the weights and biases at the lowest layers of a deep neural network learn more slowly than those of the

Email address: yatinsaraiya12@gmail.com (Yatin Saraiya)

[1]except for 3 changes in input and output sizes.

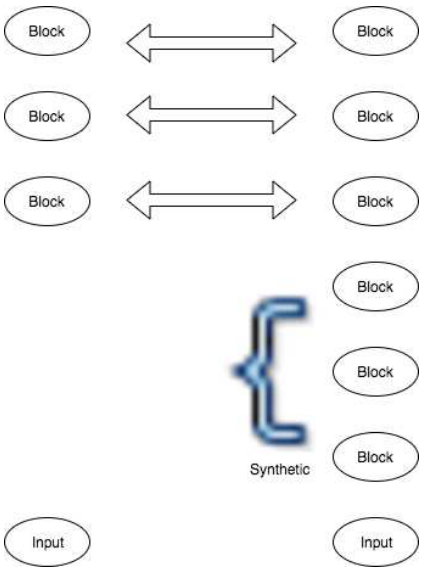

Figure 1: Predictor and corrector

upper layers. Hence, the parameters of the lower layers can be computed every other iteration with no appreciable loss of validation accuracy.

Assumption 2. We also assume that the weights and biases at the lower layers approximate the identity function. Hence, if network N_1 is obtained from N_2 by adding some layers at the bottom of the stack, then we expect there to be an approximate equivalence in the learned parameters of the common (upper) blocks.

Our methodology is to maintain and train 2 models, a shallower one (the *predictor*) and a deeper one (the *corrector*). At the end of the training, the corrector is the trained model to be used. The parameters on all blocks above the input layer in the predictor are maintained as equal to the parameters of the same number of upper blocks in the corrector. An alternation with this copy operation is done at the granularity of 1 epoch. Figure 1 shows the picture. The remaining blocks on the corrector are initialized to be the parameters of the input layer in the case of the input layer, and to the lowest non-input block of the predictor for the additional blocks of the corrector. Then SGD and backpropagation maintain these parameters (i.e they are not copied).

Let N_1 be an instance of a resdidual neural net version 1 of [1]. Assume

it has L blocks, 1 representing the input block and L representing the output block. Let B_l represent the lth block, with parameters P_l. This is the predictor. Let N_2 be the corrector, which is obtained by copying N_1 and modifying it as in Algorithm 1 below. Note that this is performed only once per training session. Algorithm 2 describes how to perform the training. Note that no special processing is required.

Algorithm 1 Construction of corrector

procedure ConstructCorrector(N_1, K) ▷ K is the number of blocks to add.
    copy N_1 to N_2
    for $i = 1, 2, \ldots, K$ **do**
        Add a copy of block B_2 in N_1 just under B_2 in N_2
    end for
end procedure

Algorithm 2 Training algorithm

procedure Train(N_1, N_2, K) ▷ K is the number of blocks added.
    for half the number of epochs **do**
        Perform one training epoch using the predictor N_1
        for l = 2, 3, . . . , L **do**
            Copy P_l in the predictor to P_{l+K} in the corrector
        end for
        Perform one training epoch using the corrector N_2
        for l = 2, 3, . . . , L **do**
            Copy P_{l+K} in the corrector to P_l in the predictor
        end for
    end for
    The corrector N_2 is the trained model.
end procedure

3. Experiments

We used `cifar10_resnet.py`, obtained from `https://github.com/fchollet/keras/blob/master/examples/`, to model the experimental framework of Section 4.2 of [1]. This software is under the

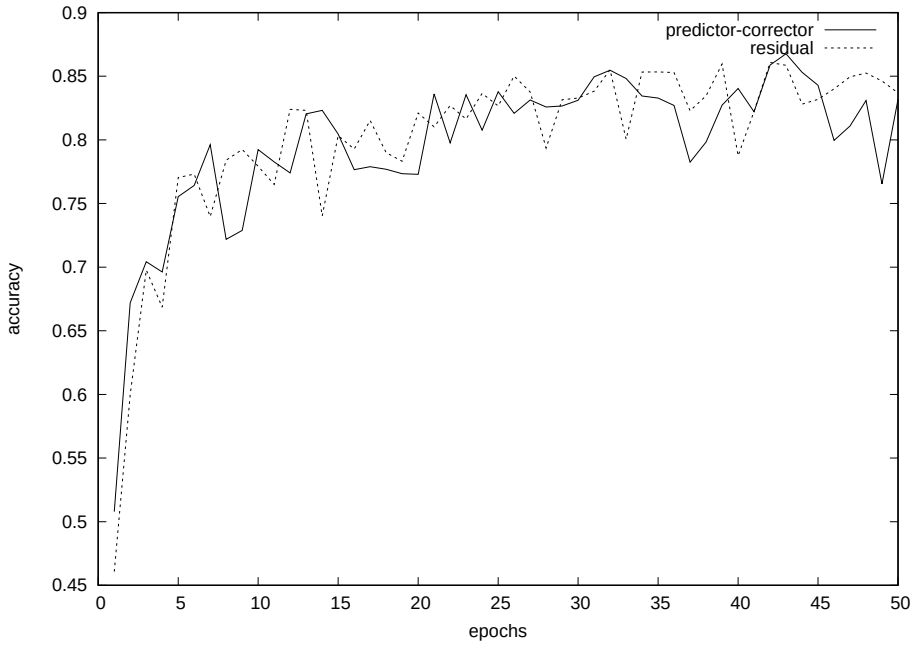

Figure 2: Validation accuracy

	Time savings
Predictor-corrector residual	9

Table 1: Time savings (%) over 50 epochs

	Min top-1 error
Residual	14.04
Predictor-corrector residual	13.24

Table 2: Top-1 validation error (%) over 50 epochs

MIT license. It is used as the predictor, with 116 layers. We modified it to create a deeper corrector model by adding 15 layers above the input layer. We ran both against the CIFAR-10 dataset [4] for 50 epochs.

The time savings were 9% (see Table 3).

Results. Our results are contained in Table 2 and Figure 2. Note that the predictor-corrector top-1 validation error is lower than that of the residual net, although marginally so.

4. Conclusions

We presented a predictor-corrector methodology for training a deep neural net using alternating epochs with a shallower and less expensive model. We gained a time savings of 9% on the CIFAR-10 dataset with no loss in validation accuracy.

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun, Deep Residual Learning for Image Recognition, CoRR (2015)

[2] Sergey Zagoruyko and Nikos Komodakis, Wide Residual Networks, CoRR (2017)

[3] Saining Xie, Ross Girshick, Piotr Dolla, Zhuowen Tu and Kaiming He, Aggregated Residual Transformations for Deep Neural Networks, CoRR (2017)

[4] A. Krizhevsky, Learning multiple layers of features from tiny images, Tech Report (2009)

[5] François Chollet et al, `https://github.com/fchollet/keras`, GitHub (2015)

[6] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012.

[7] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. Neural computation, 1989.

[8] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In ICLR, 2015.

[9] M.D.Zeiler and R.Fergus. Visualizing and understanding convolutional neural networks. In ECCV, 2014.

[10] W. L. Briggs, S. F. McCormick, et al. A Multigrid Tutorial. Siam, 2000.

[11] C. M. Bishop. Neural networks for pattern recognition. Oxford university press, 1995.