Using accumulation to optimize deep residual neural nets

Yatin Saraiya

847 Moana Court, Palo Alto 94306, CA, USA

Abstract

Residual Neural Networks [1] won first place in all five main tracks of the ImageNet and COCO 2015 competitions. This kind of network involves the creation of pluggable modules such that the output contains a residual from the input. The residual in that paper is the identity function. We propose to include residuals from all lower layers, suitably normalized, to create the residual. This way, all previous layers contribute equally to the output of a layer. We show that our approach is an improvement on [1] for the CIFAR-10 dataset.

Keywords: Residual, neural, net, accumulate

1. Introduction

Deep convolutional neural networks [6, 7] form the basis for image recognition. It has been shown [8] that depth is critical in classification accuracy. The stacked layers of such nets provide features at different granularities [9]. However, very deep neural nets suffer from degradation of training error as the networks start converging. Proposed solutions to this degradation problem include shortcutting [11], of which the use of residuals as in [1] is a modification. These networks consisted of stacked blocks with the same input-output characteristics[1], with residuals from the input added to the output via the identity function.

Figure 1 illustrates one such block. The results of [1] showed that this modular design would mitigate degradation even in very deep networks. The

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

[1] modulo a small number of changes in the layer's input and output sizes.

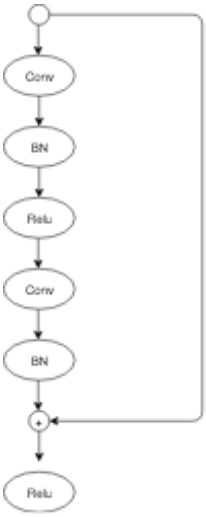

Figure 1: Residual block

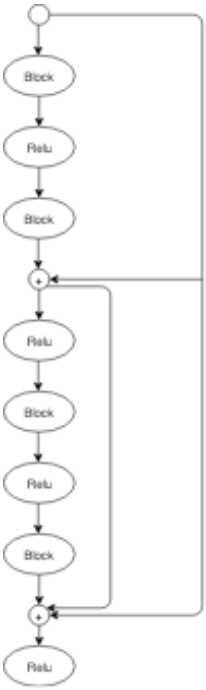

Figure 2: Accumulated residual block

fundamental improvement was the addition of residuals using the identity function. That is, if F is the function computed by a block, x is the input and y is the output,

$$y = \sigma(F(x) + x) \tag{1}$$

where σ is the rectified linear unit. The intuition is that later layers perform fine tuning on the results of the earlier layers.

We replace the identity residual with the sum of the normalizations of the inputs to each block, which necessitates just one extra variable that accumulates the residue, and one extra addition per block[2]. If the model consists of blocks $B_1, B_2, \ldots, B_L$, F_i is the function computed by block B_i, x_i is the input to block B_i and y_i is the output of this block, we have

$$y_i = \sigma(F_i(x_i) + \Sigma_{j=1}^{i}\mathrm{BN}(x_i)) \tag{2}$$

where $\mathrm{BN}(x_i)$ is the batch normalization of x_i. The intuition is that each block computes feature sets at a different granularity, so each block's output should weigh equally in the result. Figure 2 presents the architecture. We call such neural nets *accumulated residual neural nets*.

2. Experiments

We used `cifar10_resnet.py`, obtained from `https://github.com/fchollet/keras/blob/master/examples/`, which bears the MIT license, as a representation of the network described in [1]. We modified it to define the network of this paper.

We ran both against the CIFAR-10 dataset [4] with the depth at 32. The experimental setup was that of Section 4.2 of [1]. Note that the same setup was used for both the residual network and our network.

Results. Our results are contained in Table 2 and Figures 4, 5, 6 and 3.

Table 2 presents the minimum and average validation errors per epoch over 50 epochs. In each case, the net with history was at leasts 1% better than the residual net. That is, it generalizes better on the CIFAR-10 dataset.

Figure 3 compares the validation accuracy of the residual net with and without accumulation.

[2]We reinitialize whenever the input to a block changes shape, which removes the necessity for the addition.

	Min top-1 error	Avg top-1 error
ResNet	13.92	19.9
Accumulated	12.65	18.1

Table 1: Top-1 validation error over 50 epochs

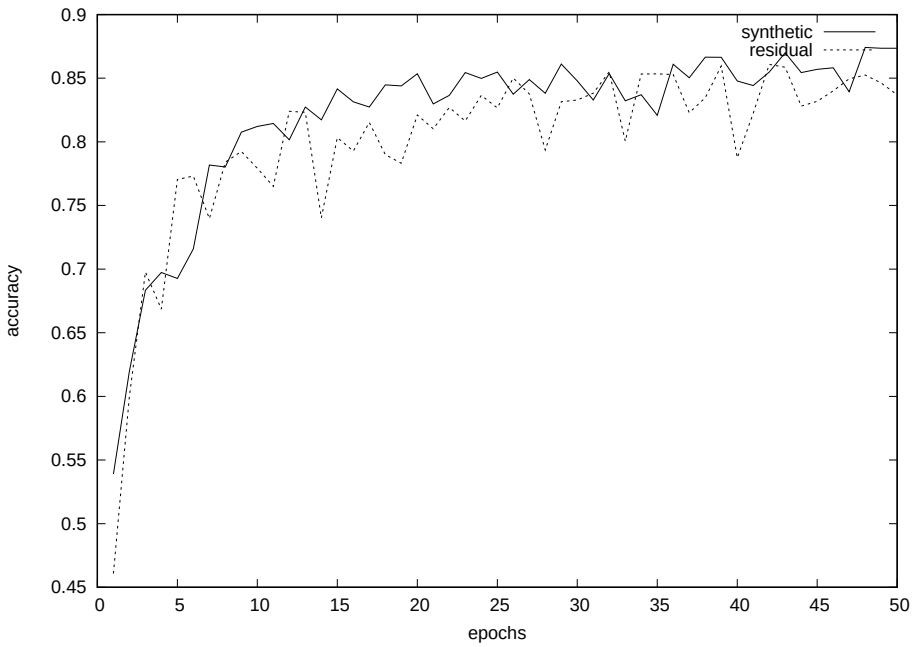

Figure 3: Validation accuracy

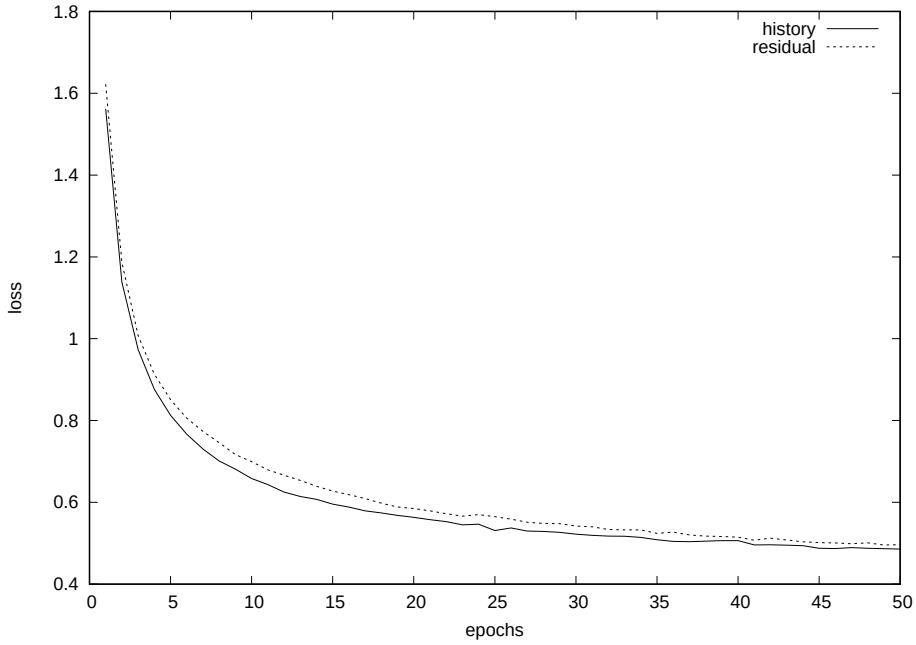

Figure 4: Training loss

Figure 4 compares the training loss of the residual net with and without accumulation. Figure 5 compares the training accuracy of the residual net with and without accumulation. Figure 6 compares the validation loss of the residual net with and without accumulation.

3. Conclusions

We presented an augmentation of residual neural networks where the residuals accumulate along the depth of the neural net. This permits the output of each layer to play an equal role in the classification. We showed that these networks outperform the residual networks of [1]. It is of interest to see whether this approach extends to the wide networks of [2] and the aggregated networks of [3].

[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)

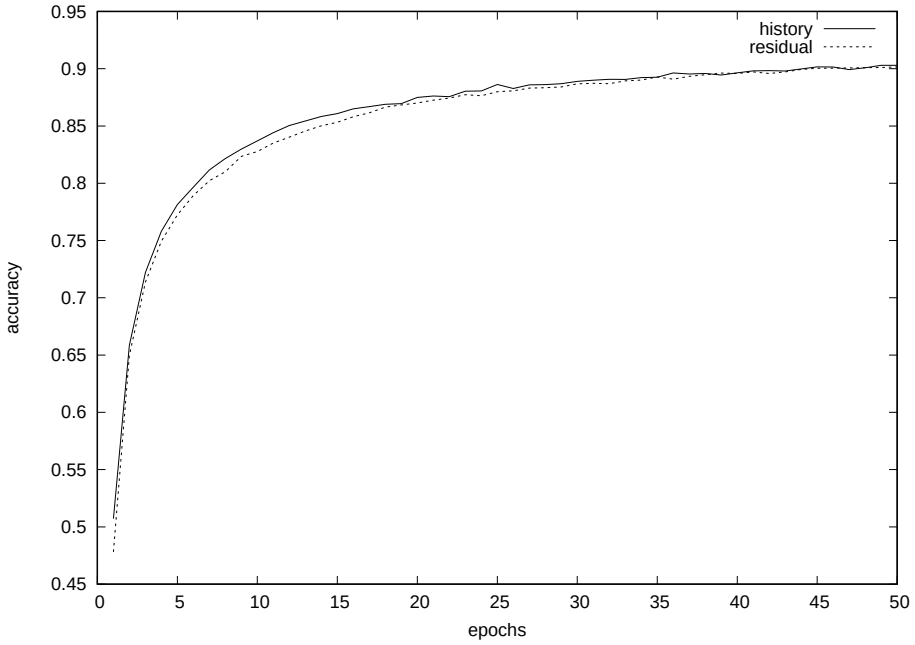

Figure 5: Training accuracy

[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.

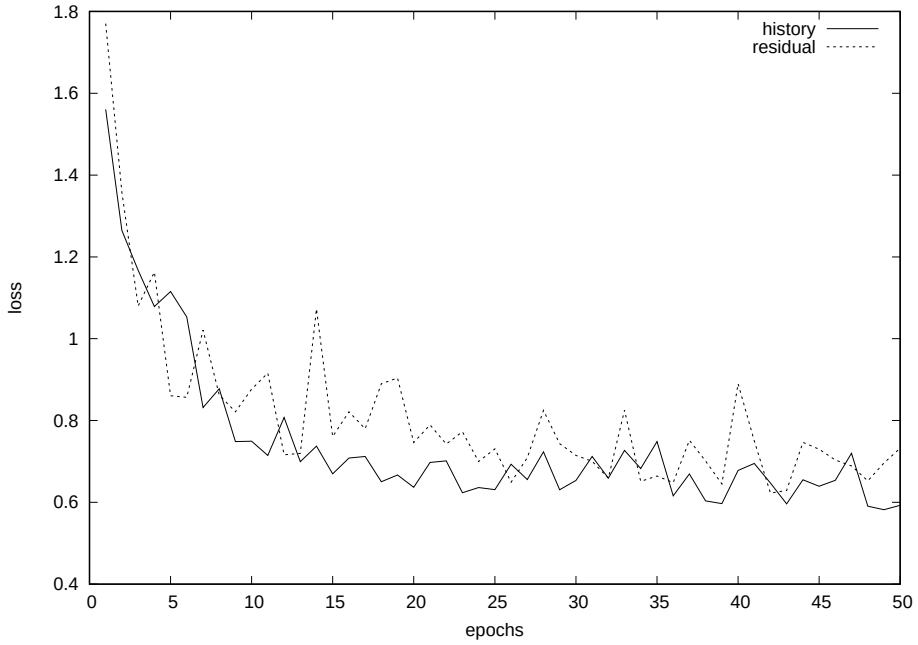

Figure 6: Validation loss

[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.