Poster Summary - Precomputing Outputs of Hidden Layers to Speed Up Deep Neural Network Training

Sohil Lal Shrestha
Computer Science and Engineering Department
The University of Texas at Arlington
Arlington, TX, USA
sohil.shrestha@mavs.uta.edu

***Abstract*—Deep learning has recently emerged as a powerful technique for many tasks including image classification. A key bottleneck of deep learning is that the training phase takes a lot of time, since state-of-the-art deep neural networks have millions of parameters and hundreds of hidden layers. The early layers of these deep neural networks have the fewest parameters but take up the most computation. In this work, we reduce training time by progressively freezing hidden layers, pre-computing their output and excluding them from training in both forward and backward paths in subsequent iterations. We compare this technique to the most closely related approach for speeding up the training process of neural network. Through experiments on two widely used datasets for image classification, we empirically demonstrate that our approach can yield savings of up to 25% wall-clock time during training with small loss in accuracy.**

***Index Terms*—Deep Learning, Artificial Neural Network, Convergence, Precomputation**

I. Introduction and motivation

Deep Learning is credited with improving many state of the art problems that Artificial Intelligence and Machine Learning community has been facing over the years. A practical instance is Face Verification by Facebook which uses DeepFace [1] to automatically tag people in uploaded photos.

Many state-of-the-art deep neural networks have millions of parameters and hundreds or perhaps thousands of hidden layers. Deep Learning models typically tend to work well with large dataset. However, even with a clean dataset and a suitable model to train the dataset on, getting a trained model that generalizes the underlying behavior well is a time consuming and resource intensive process. Generally large scale training of deep learning models are carried out in a distributed environment with multiple GPUs [2], [3].

One way to reduce training time is to fine tune the hyper parameters of the model. One such hyper parameter is the learning rate of the hidden layers. General practices in fine tuning the learning rate sets a higher learning rate initially and reduces it based on some functional policy such as cubic decay, exponential decay, etc.

To optimize training time, Brock et al. [4] proposed the FreezeOut technique. In this technique, each layer L_i starts with a single fixed learning rate, which anneals to zero at t_i, where t_i is linearly spaced between a user-selected t_0 and the total number of iterations. Once the layer's learning rate anneals to zero, it is frozen by excluding it from backward passes. The technique is motivated by an observation that early layers of many deep neural networks have comparatively fewer parameters, take most of the time while training, but converge quickly, suggesting early layers need less time to train in comparison to later layers. For example, in image classification, early layers of a deep neural network try to learn simple features such as edges or contours while later layers learn complex features of the entire image (e.g., face, car).

Brock et al. proposed to progressively freeze layers by excluding from backward passes. He reported 3% loss in accuracy, saving up to 20% training time in image classification using a state-of-the-art deep neural net model. Adding to his work, we completely exclude frozen layer by pre-computing the output and assess how much we can gain on training time. Our experimental evaluation suggests improvement in the training time of up to 25 % with small loss in accuracy compared to the work by Brock et al.

II. Background and related work

In general, training a neural net involves two steps: (1) The forward pass propagates input in forward direction through the network to compute a output label. Error is calculated comparing the actual labels with the model's predicted labels. (2) The backward pass then propagates the error in a backward direction, to compute weight gradients to update the weight parameters, minimizing the model loss or error by means of cross entropy. The process is carried out with multiple passes of training data until the desired accuracy is achieved or for a fixed number of iterations. The goal of training the model is to search for weight parameters, keeping the error or loss as low as possible.

Many approaches have been developed to speed up the training of neural networks, which can be broadly divided into two categories: (1) Algorithmic approaches exploit the training algorithm to accelerate training, whereas (2) system approaches focus on deploying enhanced resources such as super computers to achieve parallel training.

Recent work by Priya Goyal et al. [3] adopted a hyper-parameter-free linear scaling rule for adjusting learning rates as a function of mini-batch size and developed a new warm-up scheme that overcome optimization challenges early in training. Their Caffe2-based system trained ResNet-50 [5] on the ImageNet dataset in one hour using the distributed

setting. Many projects based on data parallelism have reduced the training time of deep neural network[[6]–[10]

III. Overview and Design

Figure-1 and Figure-2 give a general overview of training a neural network with both FreezeOut and PreFast. FreezeOut's approach tweaks training by fine tuning the learning rate of each layer based on user defined hyper parameter t_0. Once the learning rate of a layer anneals to zero, it excludes the layer from the backward pass saving on computation, hence reducing training time. Our PreFast approach further builds on this technique by pre-computing all frozen layer's output and saving the output in memory such that the layer can be entirely removed from training. This approach further reduces computation and thus improves on the training time.

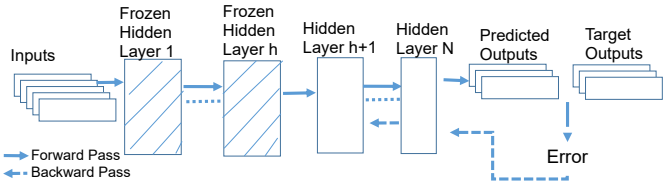

Fig. 1. FreezeOut. When a layer is frozen, the frozen layer is excluded from backward pass. Thus weights are not updated for the frozen layer.

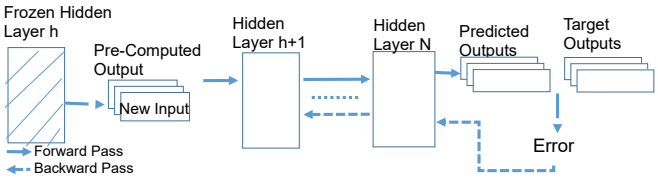

Fig. 2. PreFast. This technique computes output of the frozen layer and saves it in memory. It then uses the computed output as an input to subsequent hidden layers.

IV. Experimental setup and Evaluation

In our work, we used FreezeOut's open source implementation[1] with default hyper parameter settings and incorporated our technique to empirically compare the results. Our open source code can be found in Git hub[2]. The original Freezeout work validates its claim using the CIFAR-10 and CIFAR-100 datasets on the state-of-the-art DenseNet [11] model. We compared our technique using both of the datasets along with the MNIST dataset.

Output pre-computation of the frozen layers required large memory size. Due to resource constraints, we took multiple random samples of 2000 training images for the CIFAR datasets and 4000 training images for MNIST to train the DenseNet model and record the training time. We then tested the DenseNet model trained on both techniques on the respective test dataset to compute validation errors. The empirical results suggest that for a user-defined parameter t_0 of 0.6, we could save additional 20-25% training time without losing on accuracy in comparison to FreezeOut.

Although PreFast improves on the training time, there is trade-off between accuracy and training time compared to the base case where layers are not frozen. The acceptability of such trade-off is entirely up to the user. Thus, this technique is beneficial for the user who wants to prototype different designs of neural network and quickly observe how they compare.

[1]github.com/ajbrock/FreezeOut

[2]https://github.com/50417/PreFast

V. Conclusions

In this work, we presented the PreFast technique which extends the work of the FreezeOut technique aiming to reduce the training time of deep neural networks.

References

[1] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, ser. CVPR '14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 1701–1708. [Online]. Available: https://doi.org/10.1109/CVPR.2014.220

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[3] P. Goyal, P. Dollár, R. B. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, large minibatch SGD: training imagenet in 1 hour," *CoRR*, vol. abs/1706.02677, 2017. [Online]. Available: http://arxiv.org/abs/1706.02677

[4] A. Brock, T. Lim, J. Ritchie, and N. Weston, "Freezeout: Accelerate training by progressively freezing layers," *OPTML 2017 : 10th NIPS Workshop on Optimization for Machine Learning*, 12 2017.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[6] A. Krizhevsky, "One weird trick for parallelizing convolutional neural networks," *CoRR*, vol. abs/1404.5997, 2014. [Online]. Available: http://arxiv.org/abs/1404.5997

[7] H. Zhang, Z. Hu, J. Wei, P. Xie, G. Kim, Q. Ho, and E. P. Xing, "Poseidon: A system architecture for efficient gpu-based deep learning on multiple machines," *CoRR*, vol. abs/1512.06216, 2015. [Online]. Available: http://arxiv.org/abs/1512.06216

[8] S. Zou, C. Chen, J. Wu, C. Chou, C. Tsao, K. Tung, T. Lin, C. Sung, and E. Y. Chang, "Distributed training large-scale deep architectures," *CoRR*, vol. abs/1709.06622, 2017. [Online]. Available: http://arxiv.org/abs/1709.06622

[9] T. Ben-Nun and T. Hoefler, "Demystifying parallel and distributed deep learning: An in-depth concurrency analysis," *CoRR*, vol. abs/1802.09941, 2018. [Online]. Available: http://arxiv.org/abs/1802.09941

[10] I.-H. Chung, T. N. Sainath, B. Ramabhadran, M. Picheny, J. Gunnels, V. Austel, U. Chauhari, and B. Kingsbury, "Parallel deep neural network training for big data on blue gene/q," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '14. Piscataway, NJ, USA: IEEE Press, 2014, pp. 745–753. [Online]. Available: https://doi.org/10.1109/SC.2014.66

[11] G. Huang, Z. Liu, and K. Q. Weinberger, "Densely connected convolutional networks," *CoRR*, vol. abs/1608.06993, 2016. [Online]. Available: http://arxiv.org/abs/1608.06993