Training BatchNorm and Only BatchNorm: Affine parameter effects in MLP's

July 15, 2022

Marcus Tiedemann Oekland Henriksboe

Abstract

This report is part of a project in Deep Learning Applied AI at the University of Sapienza the spring of 2022. The starting point was the paper by (Frankle et al., 2020) where they investigated the effects of training only batch normalization layers on residual neural nets. In this project similar experiments were conducted on MLP's of varying depth and a shallow CNN. The findings largely concur with the general idea that the affine parameters in batch normalization can be more expressive than equivalent random weights in the network. The code repository can be found on Github: `https://github.com/Marcusntnu/mlp_lenet_bathnorm`.

1. Introduction

BatchNorm normalizes according to the mean and standard deviation on a mini-batch. In addition to the normalization, the affine parameters of γ and β have also shown promising results in training more accurate models and being able to reach non-trivial test accuracy when trained alone in ResNets (Frankle et al., 2020). In this project we try try this on MLP's and a shallow CNN. Our dataset is the MNIST dataset, chosen for its simpleness and lack of complexity considering the scope of this project.

Models using layers approximately as wide as the input layer (Wikipedia, 2022b) have been found to produce good results (99%+) and this size (784 units) per layer was therefore the main width of interest. For this project we produced MLP's ranging from 2 to 14 hidden layers, in intervals of 2. Other widths (10, 20, 40, 80, 100, 400) were also used in experiment producing similar results. Slimmer layers with size 10 and 100 still performed well (90%) when training all parameters and we therefore compare these results with the wider (784 units) results.

Email: Marcus Tiedemann Oekland Henriksboe <henriksboe.2036178@studenti.uniroma1.it>.

Deep Learning and Applied AI 2022, Sapienza University of Rome, 2nd semester a.y. 2021/2022.

2. Related work

(Rosenfeld & Tsotsos, 2018) investigates training only a subset of parameters and finds that it can achieve performance on par with training all. This tells us that it should not be surprising that a subset of random parameters can perform non-trivially well.
We also see that BatchNorm can make it possible to train deeper networks (He et al., 2015), that it makes the optimization landscape significantly smoother which induces a more predictive and stable behavior of the gradients, allowing for faster training (Santurkar et al., 2018)
We are on the other hand interested in what (Frankle et al., 2020) concludes which is that the affine parameters have expressive power in neural networks and that this is done through shifting and rescaling random features. In addition this performance is said to be partly due to learning to disable around a third of these random features.

3. Methodology

MLP's: We train fully connected feed forward layers on the MNIST dataset. MLP's were of interest because of their ubiquity and because they differ somewhat from convolutional neural networks while still being able to perform similar tasks. **LeNet CNN:** The LeNet-5 CNN was one of the earliest CNN's proposed and are considered simple (Wikipedia, 2022a). For our experiment we merely wanted a shallower non-residual CNN for experimenting with in this computationally resource limited project.

Hyperparameters: For BatchNorm we initialize β to 0 and sample γ uniformly between -1 and 1. Beta was deactivated for the hidden layers. Random initializations are done with the he-normal distribution. Batch sizes are also set to 128 and the models are trained for 100 epochs with early stopping at patience 3 epochs. Stochastic gradient descent is used with learning rates set at 0.01. Momentum set at 0.9 as done in (Frankle et al., 2020). **Replicates:** Experiments shown are the mean of running models two times with different random initializations.

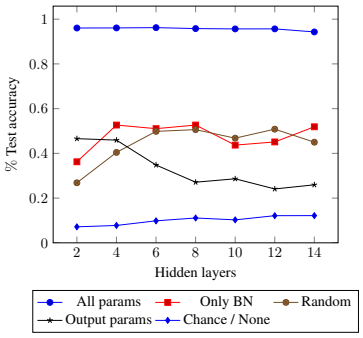

Figure 1. Test accuracy for MLP's at varying depth when width set at 10 units with different trainable parameters

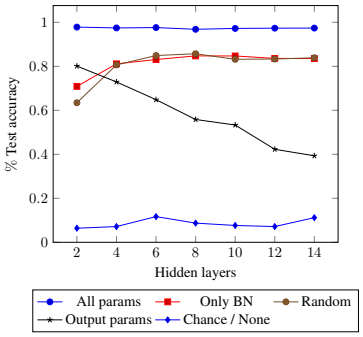

Figure 2. Test accuracy for MLP's at varying depth when width set at 100 units with different trainable parameters

4. Results

We see that when training all params the models benefit from BatchNorm, especially at deeper levels. We also see that training only BatchNorm yields non-trivial results and consistently higher than by chance.

The experiments from the LeNet CNN showed test accuracy around 30% percent. Random weights did the same, although slightly lower by a few (2-3) percent. The results for MLP's at varying depth and width are visualized in figures 1, 2 and 3.

Training on varying levels of depth for MLP's showcases that BatchNorm can reach non-trivial accuracy on MLP's too (above 90% for widths at 784) and also that the effect scales with the amount of parameters, although more visibly when scaling up width in this case rather than depth, but this applies for all model configurations. On the MNIST dataset training only BatchNorm on wide (784) and deeper (4-14 hidden layers) MLP's reach around (0.95%) accuracy while an equivalent number of random parameters training in the fully connected layers result in considerably lower (more than 10% less) test accuracy by varying amounts.

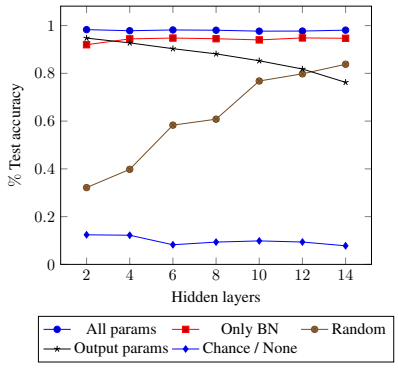

Figure 3. Test accuracy for MLP's at varying depth when width set at 784 units with different trainable parameters

5. Discussion

The results we get concur with research on other architectures like ResNets from (Frankle et al., 2020). BatchNorm and freezing all other parameters at their original initializations can result in quite high accuracy also for MLP's through their power to shift and rescale random features. Our conclusion, with some caveats, is that at larger dimensions these affine features can perform better than training random weights in the network. Contrary to the deeper ResNets (REF) trained only with BatchNorm, the benefit of deeper models for more affine parameters seem to taper off more in our experiment with BatchNorm stabilizing after around four hidden layers.

The randomness and distribution of parameters are still somewhat up for discussion. We did not find that the random or non-random distribution in a single layer mattered for performance, as was the case with ResNets for (Frankle et al., 2020). We chose to have random parameters evenly distributed in the model (the same in each layer) because the BatchNorms parameters were, but this is not necessarily the most optimal configuration for random parameters. We not know if uneven distribution among layers in the MLP model can affect performance.

Further work could include other architectures, possibly transformer based architectures or those operating on non-euclidean data, like GNN's. Hyperparametrization is not explored in this report, although we did to some tuning and include this in our delivered notebooks. It would be especially interesting to combine batch sizing, further width tuning of layers, other random initialization distributions, non-standard learning rates etc. In addition, if affine parameters are of interest it could be interesting to look at them isolated and not in conjunction with normalization, so to replace the BatchNorm layer with another function instead of comparing it only to units in the previous layer.

References

Frankle, J., Schwab, D. J., and Morcos, A. S. Training batchnorm and only batchnorm: On the expressive power of random features in cnns. *CoRR*, abs/2003.00152, 2020. URL https://arxiv.org/abs/2003.00152.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

Rosenfeld, A. and Tsotsos, J. K. Intriguing properties of randomly weighted networks: Generalizing while learning next to nothing. *CoRR*, abs/1802.00844, 2018. URL http://arxiv.org/abs/1802.00844.

Santurkar, S., Tsipras, D., Ilyas, A., and Madry, A. How does batch normalization help optimization?, 2018. URL https://arxiv.org/abs/1805.11604.

Wikipedia. LeNet — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=LeNet&oldid=1087003467, 2022a. [Online; accessed 13-July-2022].

Wikipedia. MNIST database — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=MNIST%20database&oldid=1092287929, 2022b. [Online; accessed 13-July-2022].