Designing Normalizer-Free EfficientNets

Author: Pieter Dewyse
Supervisor: Günter Klambauer
Assistant Supervisor: Richard Freinschlag

Introduction

- This study investigates how Neural Networks, EfficientNet-b0 in particular, can be designed without batch normalization (BN).
- BN is a popular normalization technique known for increasing performance, stability, and convergence time[1].
- However, it increases computational overhead, loses effectiveness in small batches, and creates problems during inference due to the reliance on batch statistics[1].
- EfficientNet is highly dependent on BN for stability, but how BN achieves this is largely unknown.
- Signal Propagation Plots (SPPs)[2] will be used to empirically record the (pre)activations, weights, and gradients to gain a better understanding of how BN affects the training.

- Question 1: Can the effect of BN be empirically measured on the forward pass and backward pass using SPPs?
- Question 2: Can a stable EfficientNet be designed without BN? If so, what is the performance impact?

Methods

- Expand the SPP framework[2] to capture and extract (pre)activations, weights, and gradients during training.
- Evaluate SPPs from training a simple Feed-Forward Neural Network (FFNN).
- Design a normalizer-free EfficientNet based on Self-Normalizing Neural Networks[3] and magnitude preserving modules. The Architecture is termed Self-Normalizing (SN) EfficientNet
- Compare the test results and SPPs between vanilla EfficientNet and its normalizer-free variant.
- Testing and training is performed on the CIFAR100 dataset.

Results & discussion

- In the FFNN, BN normalizes the forward pass but also scales the gradients in the backward pass using its learnable scale and shift parameters.
- The normalizer-free EfficientNet trains stably but underperforms compared to vanilla EfficientNet.
- Signal Propagation Plots are a useful empirical visualization tool during training, particularly for simpler test cases with fewer hyperparameters that may confound the signals.

Conclusion

- BN is more complex than just stabilizing the forward pass. It also stabilizes gradients during the backward pass by scaling.
- SPPs effectively visualize parameters, as well as the forward and backward pass dynamics, during training.
- While achieving stable training without additional normalization is valuable for a baseline, regularization is essential to achieve competitive performance.

Future directions

- Explainable AI using SPPs.
- Stabilization of the gradients in the backward pass.

Literature

[1] Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016, July 21). *Layer Normalization*.
[2] Brock, A., De, S., & Smith, S. L. (2021). *Characterizing signal propagation to close the performance gap in unnormalized ResNets*.
[3] Klambauer, G., Unterthiner, T., Mayr, A., & Hochreiter, S. (2017). *Self-Normalizing Neural Networks*.

Batch normalization stabilizes the **backward-pass** through **gradient scaling**

Enable the learnable scale ($\boldsymbol{\gamma}$) and shift ($\boldsymbol{\beta}$) parameters

$$\mathrm{BN}(x) = \boldsymbol{\gamma} \odot \frac{x - \mu_B}{\sigma_B} + \boldsymbol{\beta}$$

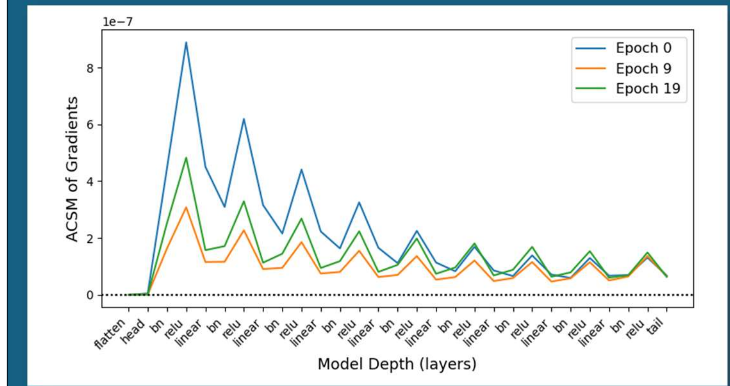


Figure 1: Gradient propagation plot for a ReLU activated Fully Connected Neural Network. Batch Normalization (BN) is used, but the learnable scale (γ) and shift (β) parameters are disabled. Gradients are measured as the Average Channel Squared Mean (ACSM)

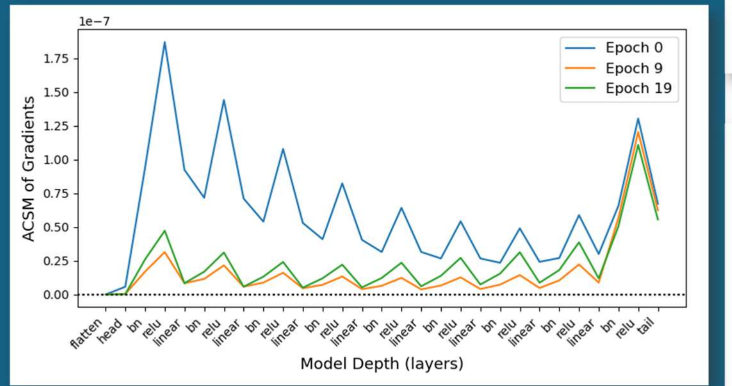


Figure 2: Gradient propagation plot for a ReLU activated Fully Connected Neural Network. Batch Normalization (BN) is used, and the learnable scale (γ) and shift (β) parameters are enabled. Gradients are measured as the Average Channel Squared Mean (ACSM)

Batch normalization can be removed but **reduces performance**

Learning rate	EfficientNet (No BN)	EfficientNet (BN)	SN-EfficientNet (No MP)	SN-EfficientNet (MP)
0.1	1.0 ± 0.0	1.2 ± 0.4	1.0 ± 0.0	1.0 ± 0.0
0.01	1.0 ± 0.0	13.9 ± 3.4	1.0 ± 0.0	1.0 ± 0.0
0.001	1.0 ± 0.0	33.5 ± 0.8	30.0 ± 0.2	29.2 ± 0.5

Table 1: Test accuracy (%) on CIFAR100 after training EfficientNet (with and without Batch Normalization (BN)) and Self-Normalizing (SN) EfficientNet (With and without magnitude preserving modules (MP)). SN-EfficientNet is built by replacing Swish with SELU activations, adding MP modules on the merging of skip connections with the main path, using LeCun initialization, and removing BN

Magnitude Preserving (MP) Modules

To merge Skip connections and main path

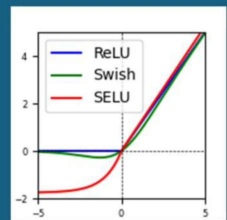
$$\frac{(1-w)\cdot \boldsymbol{x}_{main} + w \cdot \boldsymbol{x}_{skip}}{\sqrt{(1-w)^2 + w^2}}$$

To scale Squeeze-and Excitation output

$$\frac{\boldsymbol{x} \cdot \boldsymbol{w}_{\boldsymbol{SE}}}{\sqrt{(1-\boldsymbol{w}_{\boldsymbol{SE}})^2 + \boldsymbol{w}_{\boldsymbol{SE}}^2}}$$

SELU activations[3]

$$SELU(x) = \lambda \begin{cases} x & if\ x > 0 \\ \alpha e^x - \alpha & if\ x \le 0 \end{cases}$$



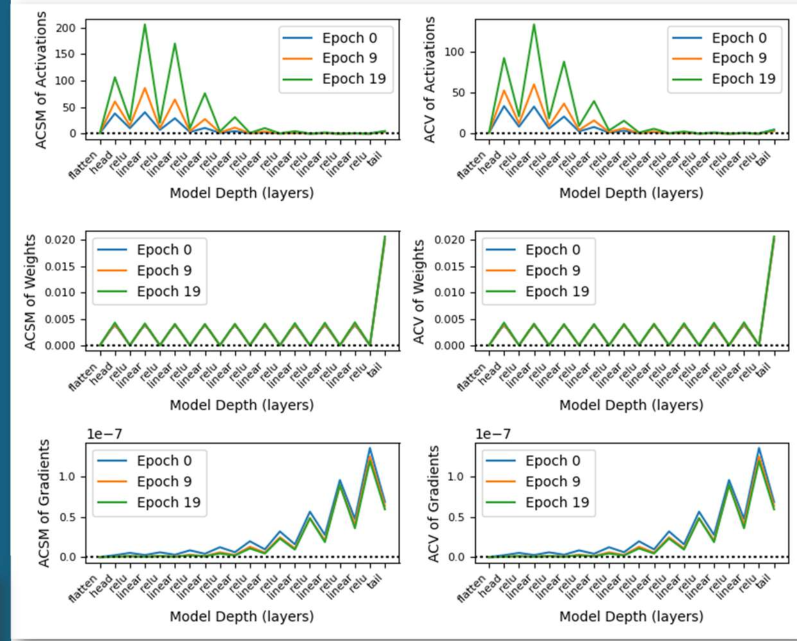


Figure 3: Signal propagation plots for a ReLU activated FFNN.

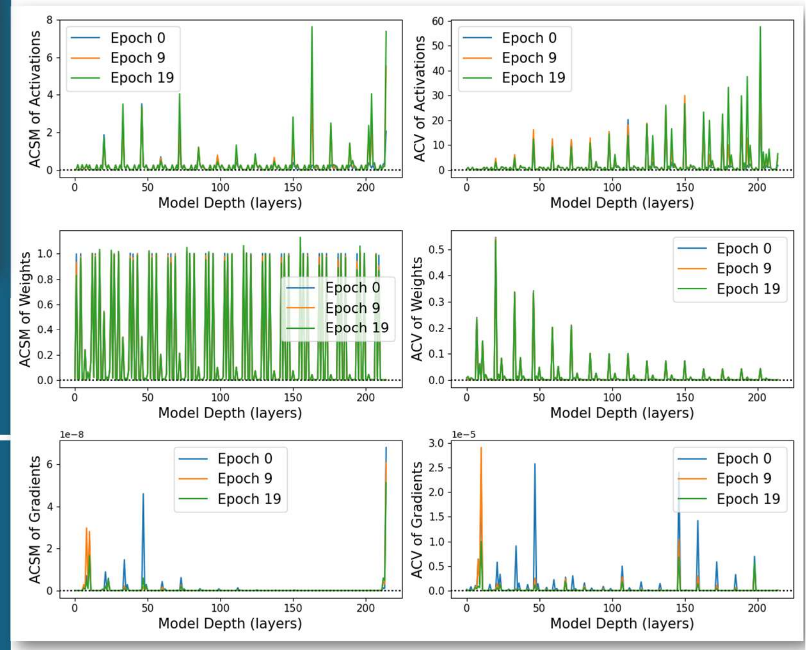


Figure 4: Signal propagation plots for Vanilla EfficientNet.

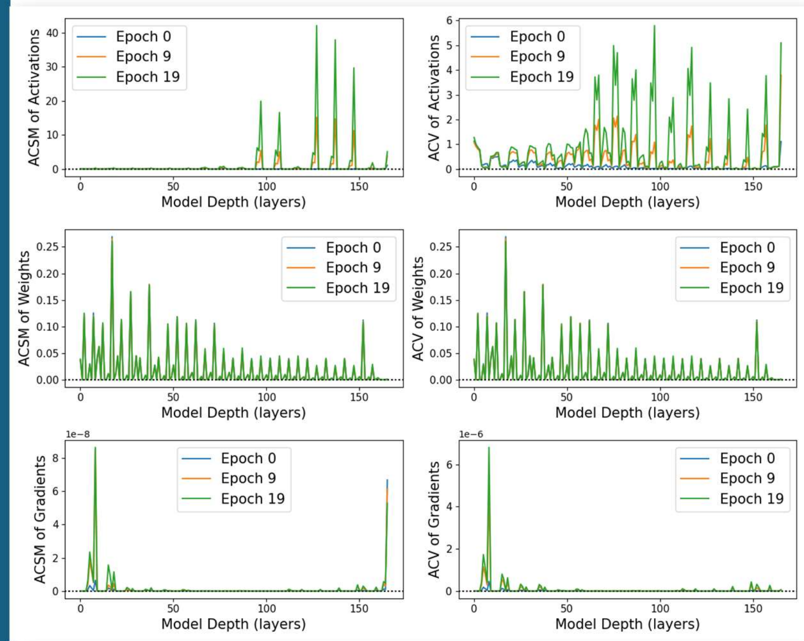


Figure 5: Signal propagation plots for SN-EfficientNet with Magnitude preserving modules

JOHANNES KEPLER UNIVERSITY LINZ