ulm university universität
uulm

Faculty of Engineering, Computer Science and Psychology
Institute of Neural Information Processing

Group Normalization

Examination Project at Ulm University

Presented by:
Carolin Schindler
carolin.schindler@uni-ulm.de
978616

Primary Instructor:
Prof. Dr. Heiko Neumann

Secondary Instructor:
M. Sc. Christian Jarvers

DeepVision - Deep Learning and Convolutional Neural Networks in Computational Vision (2021)

Last updated May 31, 2021

1 Overview

The following work deals with Group Normalization as presented by Wu and He [21]. We first position Group Normalization in the context of the literature and hence also in the context of the lecture. Afterwards, we introduce Group Normalization, outline the results of the experiments conducted by Wu and He [21] and summarize their main findings which indicate the relevance of Group Normalization. Finally, we conclude with Future Work.
In the jupyter notebook included in this submission, we implemented a Group Normalization layer and investigated to which extend Figure 1 in Wu and He's [21] work can reproduced on the Fashion-MNIST [22] dataset with a smaller network.

2 Normalization

Group Normalization [21] is, as its name implies, a normalization technique. Normalization layers [13, esp. chap. 4, pp. 32-43] are important in order to speed up the training procedure [19], especially in case of deep neural networks.

Former methods, such as Local Response Normalization (LRN) [10], enhance contrasts in the features and are motivated by the principle of lateral inhibition in neurobiology. LRN performs squared-amplitude normalization in a local neighbourhood without any learnable parameters: Inter-channel LRN defines the neighbourhood at a single position in the feature map over different channels, intra-channel LRN uses a single channel with the neighbourhood being spatially close pixels in this channel.

Recent methods, among them Group Normalization, prevent internal covariate shifts (variations in the statistics of the data distribution at each layer) that arise due to updated parameters of previous layers in the network. When the distribution of the input data to a layer keeps changing, training cannot be performed effectively. Hence in a network without normalization, layers are converging sequentially from the input layer to the output layer, while in a network with normalization, the layers can learn and converge simultaneously.

Covariate shifts can be tackled by different kinds of normalization layers that mathematically perform the same statistical computation (1), but the mean μ_i and standard deviation σ_i are estimated over a differently defined set of pixels S_i [2, 13, 21].

$$\hat{x}_i = \frac{x_i - \mu_i}{\sigma_i} \tag{1}$$

$$\mu_i = \frac{1}{|S_i|} \sum_{k \in S_i} x_k \tag{1a}$$

$$\sigma_i = \sqrt{\frac{1}{|S_i| - 1} \sum_{k \in S_i} (x_k - \mu_i)^2 + \epsilon} \tag{1b}$$

where x_i is an input feature of the specific layer with $i = (i_N, i_H, i_W, i_C)$ indexing the feature in the axes (batch size, height, width, channel) and ϵ is a small constant preventing division by zero in Equation (1). The uncertainty in the estimate of the mean and standard deviation additionally introduces a regularization effect into the network: The smaller $|S_i|$, the larger the uncertainty and hence also the regularization. Further, a linear transformation (2) is applied on a per-channel basis to compensate for possible losses in representational ability due to the normalization.

$$y_i = \gamma_{i_C} \cdot \hat{x}_i + \beta_{i_C} \tag{2}$$

where the scaling factor γ and the offset β are trainable parameters. In the following we discuss the choice of S_i (visualized in Figure 1) for some well-known normalization layers.

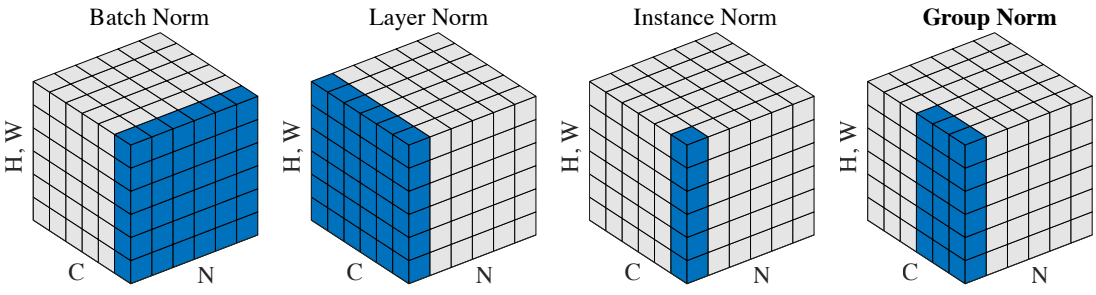

Figure 1: Visualisation of S_i for different types of normalization
N: batch axis, H: height axis, W: width axis, C: channel axis
Figure from Wu and He [21]

Batch Normalization (BN) [8] exploits the batch dimension for the calculation, the set $S_i = \{k|k_C = i_C\}$ contains all features of a specific channel. As we normalize over the batch dimension, the error in the estimate of the mean and standard deviation increases with decreasing batch size. Thus, the deployment of BN requires a sufficiently large batch size which induces constraints on the memory capacity. In order to improve the performance degradation with small batch sizes, one can apply Batch Renormalization [7] (two additional hyper-parameters to constrain the mean and standard deviation estimates) or use Synchronized Batch Normalization [14] (demands more GPUs). Furthermore, the concept of batch size is not always applicable. While there is a batch dimension during the training of a model, there is non at inference time. Therefore, pre-computed mean and standard deviation values from the training need to be used, leading to inconsistencies between training, transferring and testing.

Layer Normalization (LN) [1] computes the mean and standard deviation for each sample individually with $S_i = \{k|k_N = i_N\}$.

Instance Normalization (IN) [20] estimates the parameters of the data distribution with respect to each sample and each channel: $S_i = \{k|k_N = i_N, k_C = i_C\}$. This is comparable to BN with batch size 1.

LN and IN do not normalize over the batch dimension and hence avoid the above named drawbacks of BN. They are effective for training sequential and generative models. However, they are not able to achieve the accuracy of BN in many visual recognition tasks and hence the introduction of Group Normalization [21].

3 Group Normalization

Group Normalization (GN) [21] defines $S_i = \{k|k_N = i_N, \lfloor \frac{k_C}{C/G} \rfloor, \lfloor \frac{i_C}{C/G} \rfloor\}$, where the hyper-parameter G (default: $G = 32$) states the number of groups. The normalization is performed per sample and per group with $\lfloor C/G \rfloor$ channels. GN estimates the mean and the standard deviation independent of the batch size, such as LN and IN, and thus does not face the batch size related issues of BN. Actually, LN and IN are extreme cases of GN: GN becomes LN for $G = 1$ (group contains all channels)

and IN for $G = C$ (one channel per group). The representational power of GN is improved compared to LN and IN as the channels are not entirely independent of each other (which is disregarded by IN) but also do not necessarily share the same distributional characteristics amongst them all (which is assumed by LN and can be less valid when convolutional layers are deployed).

Wu and He [21] investigated the performance of Group Normalization in several experimental setups.
First, they trained models with different normalization layers over the batch sizes 32, 16, 8, 4, 2 (other than that the training procedure was exactly the same) for the task of image classification on the ImageNet [16] dataset. For the ResNet-50 [6] model and batch size 32, BN performs best on the validation set followed by GN (+0.5% error), LN and IN. Interestingly, GN has a lower training error than BN, which implies that GN is effective for improving the optimization but does lack some of the regularization ability provided by BN. However, these results could not be reproduced by the authors with the VGG-16 [17] model, where GN performs 0.4% better than BN on the validation data. GN, LN and IN are similarly robust to changes in the batch size, whereas BN gets worse with smaller batch sizes (esp. for batch sizes 4 and 2). This leads to GN having a 10.6% higher accuracy than BN for batch size 2. Even Batch Renormalization (BR) cannot reach the error rate of GN but performs better than the standard BN (error for batch size 4: GN 24.2%, BR 26.3%, BN: 27.3%). Similar results have been found when comparing BN and GN in the ResNet-101 [6] model. GN still performs well when varying the hyper-parameter $G = 32, 16, 8, 4, 2$; only for $G = 1$ (when GN becomes LN) the error increases. When fixing the number of channels $\lfloor C/G \rfloor$ per group, GN has a considerably higher accuracy than IN until GN equals IN with $\lfloor C/G \rfloor = 1$. Applying large-batch distributed training [4] (total batch size changes: 256, 512, 1024, 2048; per-GPU batch size is fixed: 32), shows that GN, LN and IN are more sensitive to larger total batch sizes than BN. The main difference is that for the former methods the gradient of a sample is independent of all other samples, whereas for BN the gradient of one sample is dependent of all other samples in the same GPU. The authors suggest that this hierarchical batching property might be essential in order for large-batch distributed training to work sufficiently.
Second, object detection and segmentation was investigated with the COCO [12] dataset and the Mask R-CNN [5] model. Usually, smaller batch sizes are utilized

as these tasks benefit from higher-resolution images which require more memory. When applying transfer-learning with the previously trained ResNet-50 models, GN performs better than BN (with pre-computed mean and standard deviation) and the authors show that this is mainly due to the normalizations in the head of the model: BN suffers from the non-iid (independent and identically distributed) data created by the Region-of-Interest layer, however this is not an issue for GN. GN also performs better than LN in the transfer-learning task. When training the Mask R-CNN from scratch, GN was able to achieve the best from-scratch results reported at the time of publication (41.0 box AP and 36.4 mask AP).
Finally, the Kinetics [9] dataset was used to investigate video classification with Inflated 3D convolutional networks [3]. Due to the 3D spatial-temporal dimension, the memory is constraint and small batch sizes are common practice, there is a trade-off between temporal length and batch size. When using 32-Frames inputs, BN is slightly better than GN for batch size 8 but worse for batch size 4. With 64-Frames inputs, only batch size 4 could be tested: The increased temporal length has a positive effect on both BN and GN, but for BN this effect is diminished compared to GN caused by it suffering from small batch sizes.

All in all, normalization is an essential part of neural networks in order to compensate for covariate shifts. Wu and He [21] conclude that Group Normalization is a "simple and competitive alternative to Batch Normalization" that can effectively replace BN in a variety of visual recognition tasks (image and video classification, object detection and segmentation) and importantly even outperforms Layer Normalization and Instance Normalization in these tasks. As GN is independent of the batch size (unlike BN), it can be applied the same way to training, transferring and testing and it does not suffer from small batch sizes. Hence, it can improve memory consumption by using smaller batch sizes in favour of higher-resolution data or larger models without reducing the performance. Moreover, GN is not affected by non-iid data and can substantially improve the from-scratch training of object detectors. The only limitation of GN compared to BN found by the authors was its reduced performance in large-batch distributed training and that it might improve the optimization of the training while at the same time loosing some generalization ability.

4 Future Work

Based on the work by Wu and He [21], the following opportunities for future work arise: Due to the immense influence of BN, many state-of-the-art models are designed for this normalization method. Thus, the authors propose to re-design the architectures for GN and to search for better hyper-parameter settings such that the results with GN can be improved, for example by applying more suitable regularizers. The relationship between GN, LN and IN suggests to further investigate the deployment of GN in sequential and generative models for which LN and IN provide successful results. The authors also announced that they will investigate how GN performs in reinforcement learning tasks where BN is of importance for training deep models.

Since then, new normalization methods were proposed that exploit the principle idea of GN, e. g. combining Batch and Group Normalization to "General Batch Group Normalization" [18] ($S_i = \{k | \lfloor \frac{k_C}{C/G} \rfloor, \lfloor \frac{i_C}{C/G} \rfloor\}$) or to "Batch Group Normalization" [23] (groups are defined over the (channel $\times$ height $\times$ width) dimension). Furthermore, Qiao et al. [15] found that combining GN with Weight Standardisation, which implements some benefits of BN (smoothing effects, avoidance of elimination of singularities) into the network, outperforms BN and GN in both small- and large-batch training. This effect can be even improved when replacing GN by their introduced "Batch Channel Normalization" (BN and GN at the same time).

Moreover, we think that it would be interesting to consider the application of GN (or any of its above named variants) to Continual Learning [11] tasks where the data distributions are non-iid and hence BN would suffer.

Bibliography

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. “Layer Normalization”. In: (2016). arXiv: 1607.06450 [stat.ML].

[2] Allan G. Bluman. *Elementary Statistics: A Step by Step Approach*. 7th ed. McGraw-Hill Higher Education, 2009.

[3] João Carreira and Andrew Zisserman. “Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 4724–4733. DOI: 10.1109/CVPR.2017.502.

[4] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. *Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour*. 2018. arXiv: 1706.02677 [cs.CV].

[5] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. “Mask R-CNN”. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2017.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep Residual Learning for Image Recognition”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[7] Sergey Ioffe. “Batch Renormalization: Towards Reducing Minibatch Dependence in Batch-Normalized Models”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS’17. Long Beach, California, USA: Curran Associates Inc., 2017, 1942—1950.

[8] Sergey Ioffe and Christian Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, 2015, pp. 448–456. URL: http://proceedings.mlr.press/v37/ioffe15.html.

[9] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. “The Kinetics Human Action Video Dataset”. In: (2017). arXiv: 1705.06950 [cs.CV].

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger. Vol. 25. Curran Associates, Inc., 2012.

[11] Timothée Lesort. “Continual Learning: Tackling Catastrophic Forgetting in Deep Neural Networks with Replay Processes”. PhD thesis. 2020. arXiv: 2007.00487 [cs.LG].

[12] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. “Microsoft COCO: Common Objects in Context”. In: *Computer Vision – ECCV 2014*. Springer International Publishing, 2014, pp. 740–755. DOI: 10.1007/978-3-319-10602-1_48.

[13] Heiko Neumann. “DeepVision - Deep Learning and Convolutional Neural Networks in Computational Vision”. Lecture. Ulm University, 2021.

[14] Chao Peng, Tete Xiao, Zeming Li, Yuning Jiang, Xiangyu Zhang, Kai Jia, Gang Yu, and Jian Sun. “MegDet: A Large Mini-Batch Object Detector”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 6181–6189. DOI: 10.1109/CVPR.2018.00647.

[15] Siyuan Qiao, Huiyu Wang, Chenxi Liu, Wei Shen, and Alan Yuille. “Micro-Batch Training with Batch-Channel Normalization and Weight Standardization”. In: (2020). arXiv: 1903.10520 [cs.CV].

[16] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. “ImageNet Large Scale Visual Recognition Challenge”. In: *International Journal of Computer Vision* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[17] Karen Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *International Conference on Learning Representations*. 2015.

[18] Cecilia Summers and Michael J. Dinneen. “Four Things Everyone Should Know to Improve Batch Normalization”. In: (2020). arXiv: 1906.03548 [cs.LG].

[19] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S. Emer. “Efficient Processing of Deep Neural Networks: A Tutorial and Survey”. In: *Proceedings of the IEEE* 105.12 (2017), pp. 2295–2329. DOI: 10.1109/JPROC.2017.2761740.

[20] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. “Instance Normalization: The Missing Ingredient for Fast Stylization”. In: (2017). arXiv: 1607.08022 [cs.CV].

[21] Yuxin Wu and Kaiming He. “Group Normalization”. In: *International Journal of Computer Vision* 128.3 (2020), pp. 742–755. DOI: 10.1007/s11263-019-01198-w.

[22] Han Xiao, Kashif Rasul, and Roland Vollgraf. “Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms”. In: (2017). arXiv: 1708.07747 [cs.LG].

[23] Xiao-Yun Zhou, Jiacheng Sun, Nanyang Ye, Xu Lan, Qijun Luo, Bo-Lin Lai, Pedro Esperanca, Guang-Zhong Yang, and Zhenguo Li. “Batch Group Normalization”. In: (2020). arXiv: 2012.02782 [cs.LG].