LocalNorm: Robust Image Classification Through Dynamically Regularized Normalization

Yin, B.; Scholte, H.S.; Bohté, S.

DOI
10.1007/978-3-030-86380-7_20

Publication date
2021

Document Version
Final published version

Published in
Artificial Neural Networks and Machine Learning – ICANN 2021

License
Article 25fa Dutch Copyright Act

Link to publication

Citation for published version (APA):
Yin, B., Scholte, H. S., & Bohté, S. (2021). LocalNorm: Robust Image Classification Through Dynamically Regularized Normalization. In I. Farkaš, P. Masulli, S. Otte, & S. Wermter (Eds.), *Artificial Neural Networks and Machine Learning – ICANN 2021: 30th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 14–17, 2021 : proceedings* (Vol. IV, pp. 240-252). (Lecture Notes in Computer Science; Vol. 12894). Springer. https://doi.org/10.1007/978-3-030-86380-7_20

LocalNorm: Robust Image Classification Through Dynamically Regularized Normalization

Bojian Yin[1(✉)], H. Steven Scholte[2], and Sander Bohté[1,2]

[1] CWI, Machine Learning Group, Amsterdam, The Netherlands
byin@cwi.nl
[2] University of Amsterdam, Amsterdam, The Netherlands

Abstract. While modern convolutional neural networks achieve outstanding accuracy on many image classification tasks, they are, once trained, much more sensitive to image degradation compared to humans. Much of this sensitivity is caused by the resultant shift in data distribution. As we show, dynamically recalculating summary statistics for normalization over batches at test-time improves network robustness, but at the expense of accuracy. Here, we describe a variant of Batch Normalization, LocalNorm, that regularizes the normalization layer in the spirit of Dropout during training, while dynamically adapting to the local image intensity and contrast at test-time. We show that the resulting deep neural networks are much more resistant to noise-induced image degradation, while achieving the same or slightly better accuracy on non-degraded classical benchmarks and where calculating single image summary statistics at test-time suffices. In computational terms, LocalNorm adds negligible training cost and little or no cost at inference time, and can be applied to pre-trained networks in a straightforward manner.

1 Introduction

Methods that reduce internal covariate shift via learned rescaling and recentering neural activation, like Batch Normalization [8], have been an essential ingredient for successfully training deep neural networks (DNNs). In Batch Normalization, neural activation values are rescaled with trainable parameters, where summary neural activity is typically computed as mean and standard deviation over training samples. Such summary statistics however are sensitive to the input distribution, failing to generalize when novel images are outside this distribution, for example when faced with different and unseen lighting or noise conditions [5].

Where the original Batch Normalization computed statistics across the activity in a single feature map (or *channel*) [8], trainable normalizations have been proposed along a number of dimensions of deep neural network layers [2,13,18,20]. Other normalization approaches focus specifically on domain generalizability where distributions shift between source and target domains [4,12,16]. While these methods each have their merits, they do not resolve the sensitivity

I. Farkaš et al. (Eds.): ICANN 2021, LNCS 12894, pp. 240–252, 2021.
https://doi.org/10.1007/978-3-030-86380-7_20

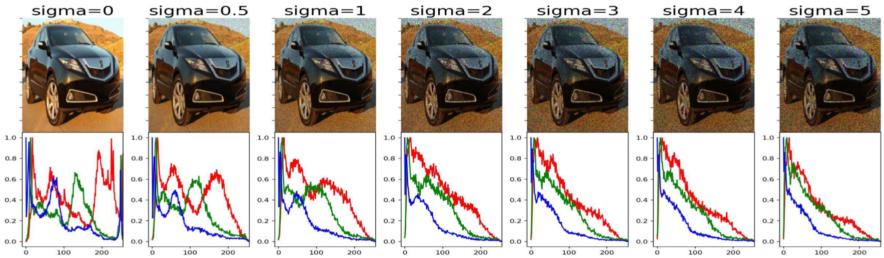


Fig. 1. RGB-histogram for increasing additive Gaussian noise (Color figure online)

of DNNs to image-degradation and the resultant distributional shifts in unseen images, which can be considered within-domain, dynamic and continuous distribution shifts.

Here, we propose a variant of Batch Normalization (BatchNorm), Local Normalization (LocalNorm): we observe that the mean and variance in channel activity changes when images are subjected to noise-related degradation. And the pixel distribution will significantly affect the network performance. Figure 1 shows an example of how the addition of Gaussian Noise flattens the color distribution for each channel in an image - other types of noise similarly affect the summary statistics. As we show, simply computing summary statistics at test-time similar to [12] – which we term *Dynamic BatchNorm* – does not resolve the problem: for large sized images, robustness increases, but accuracy decreases; for smaller sized images all accuracy is lost as the summary statistics computed from a single test-image prove too volatile.

To increase robustness of trained networks for variable summary statistics, LocalNorm regularizes the normalization parameters during training by splitting the Batch into Groups, each with their own normalization scaling parameters. At test-time, the summary statistics are then computed on the fly, either over a single image or a set (batch) of images from the test-set. For even single large images, this approach increases accuracy over and beyond both standard and dynamic Batch Normalization accuracy while retaining robustness; for single small images we demonstrate how a simple data augmentation strategy similarly fixes the accuracy while drastically increasing robustness to image degradation: the trained networks exhibit strong performance for unseen images with noise conditions that are not in the training set.

We also find that LocalNorm improves classification of distorted images in general, as measured on the CIFAR10-c and ImageNet datasets [7]. LocalNorm is straightforward to implement, also for networks already trained with standard BatchNorm - we demonstrate how a large pretrained ResNet152 network retrained further with LocalNorm significantly improves accuracy. We further show that LocalNorm achieves competitive performance on image classification benchmarks at little additional computational expense.

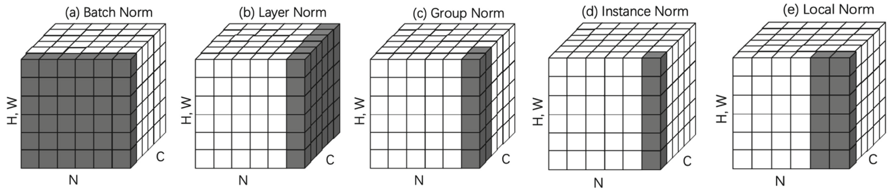


Fig. 2. Variants of Normalization Methods. Each cube corresponds to a feature map tensor, with N as the batch axis, C as the channel axis, and (H, W) as the spatial axes – height and width. The pixels in gray are normalized by the same mean and variance, computed by aggregating the values of these pixels.

2 Related Work

Lighting and noise conditions can vary wildly over images, and various preprocessing steps are typically included in an image-processing pipeline to adjust color and reduce noise. In traditional computer vision, different filters and probabilistic models for image denoising are applied [14]. More recent approaches for noise removal include deep neural networks, like Noise2Noise [11], DURR [22], and a denoising AutoEncoder [19] where the network is trained on a combination of noisy and original images to improve its performance on noisy dataset thus increasing the networks' robustness to image noise and also to train a better classifier. However, as noted in [5], training on images that include one type of noise in DNNs does not generalize to other types of noise.

Neural Normalizing Techniques. Normalization is typically used to rescale the dynamic range of input. This idea has been applied to deep learning in various guises, and notably Batch Normalization (**BatchNorm**) [8] was introduced to renormalize the mean and standard deviation of neural activations using an end-to-end trainable parametrization. **Normalization** is generally computed as

$$\hat{x_i} = \frac{x_i - \mu_i}{\sigma_i + \epsilon} * \gamma + \beta$$

where the x_i is a feature tensor of input $X = \{\cup x_i\}$ computed by the previous layer and γ and β are the (trainable) scaling parameters. For normal RGB or GBR images, $i = (i_N, i_W, i_H, i_C)$ is a 4D vector indexing the feature in $[N, W, H, C]$ order where N is the batch size(number of images per batch), H and W are the spatial height and width axes, and C is the channel axis.

The space spanned by N, H, W, C can be subdivided and subsequently normalized in multiple ways. We call the subdivision, the elements on which this normalization is performed, a group G_k: different forms of input normalization can be described as dealing with different groups. The mean μ_k and std σ_k of the certain computation group G_k are computed as:

$$\mu_k = \frac{1}{m} \sum_{x_j \in G_k} x_j; \quad \sigma_k = \sqrt{\frac{1}{m} \sum_{x_j \in G_k} (x_j - \mu_i)^2 + \epsilon}$$

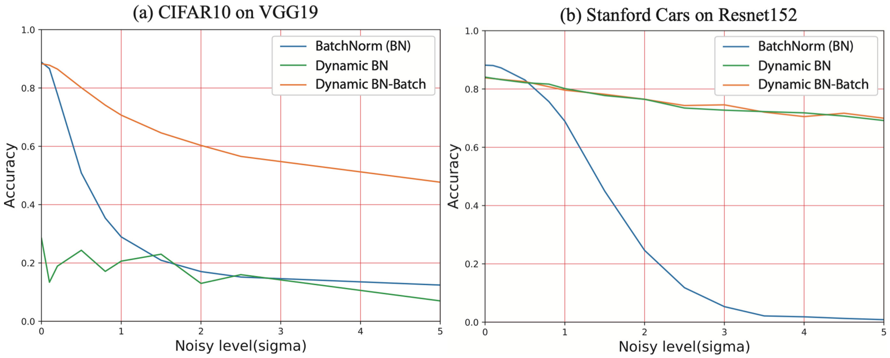


Fig. 3. Performance of VGG19 network applied to CIFAR10 (a) and a ResNet152 network to the stanford Cars dataset (b) where the test-images are subjected to increasing amounts of image degradation, here in the form of additive Gaussian noise. Blue: accuracy for standard batch normalization. Green: accuracy on dynamic batch normalization evaluated on single images. Orange: accuracy on dynamic batch normalization with summary statistics computed over a batch of test-images. (Color figure online)

where ϵ is a small constant. The computation group G_k (where $X = \{\cup G_k \mid k = 1, 2, \ldots K\}$) is a set of pixels which shares the mean μ_k and std σ_k, and m is the size of the group G_k. BatchNorm and its variants, like Layer Normalization [2], Instance Normalization [17,18] and Switchable Normalization [13], can be mapped to a computational group along various axes (Fig. 2).

Dynamic Batch Normalization. For BatchNorm, the mean μ and standard deviation (std) σ are calculated along all training samples in a channel and then fixed for evaluation on test images; as noted however, when the (test) image distribution changes, these statistical parameters will drift. As a result, DNNs with BatchNorm layers are sensitive to input that deviates from the training distribution, including noisy images. Simply computing the summary statistics on-the-fly at test-time [12], to account for potential drift, only partly solves the problem: in Fig. 3, we show what happens when the mean μ and std σ are computed as dynamical quantities also at test time for standard benchmarks CIFAR10 and Stanford Cars, using modern deep neural networks (for details, see below). For each test image (or batch of test images) we compute (μ, σ), for increasing noise (here for additive Gaussian noise).

For CIFAR10, we find that using single test images when evaluating gives poor results (Fig. 3a; Dynamic BN), as the small (32×32) images do not result in channel activity sufficient for effective summarizing statistics. However, computing these statistics over a batch shows a marked improvement (Fig. 3a; Dynamic BN-Batch): then, test accuracy exceeds standard BatchNorm for noisy images, at the expense of a slight decrease in accuracy for noiseless images. For the large images in Stanford Cars, we see that dynamically computing (μ, σ) at test time even for single images drastically improves accuracy for noise-degraded images (Fig. 3b); the classification accuracy absent noise however drops. While computing summary statistics over a batch at test-time is feasible for benchmarking purposes, real world application would correspond to for example using a video stream, which would however substantially increase computational cost and latency, and decrease accuracy on noiseless data.

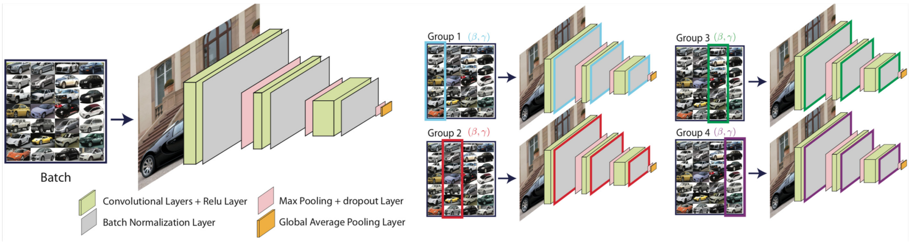

Batch
Group 1 (β, γ)
Group 2 (β, γ)
Group 3 (β, γ)
Group 4 (β, γ)
Convolutional Layers + Relu Layer
Batch Normalization Layer
Max Pooling + dropout Layer
Global Average Pooling Layer

Fig. 4. LocalNet. A deep network with standard batch normalization computes single summary statistics over the entire batch. In LocalNorm, summary statistics are computed over groups, where each group k is associated with its own scaling parameters β_k, γ_k (while sharing the all other network parameters), and summary statistics (μ_k, σ_k) are dynamically computed also at test-time on the test-images.

3 Local Normalization (LocalNorm)

We develop LocalNorm, which dynamically uses the local summary statistics in a group to improve the robustness of DNNs to various noise conditions and show how resolves the loss of accuracy when computing summary statistics at test-time in Dynamic BatchNorm, while also improving the robustness to image degradation and distributional shifts at test-time.

In LocalNorm, we regularize the normalization layer for variations in μ and σ. The aim is to make the trained architecture less sensitive to changes in these statistics at test-time, such that we can dynamically recompute μ and σ on test-images. We divide the Batch into separate Groups G_k for which we each compute summarizing statistics μ_k, σ_k and associate separate scaling parameters γ_k and β_k with each Group, as illustrated in Fig. 4. For LocalNorm the computational group is defined along the $(N/K, H, W)$ axes (Fig. 2(e)):

$$G_k = \left\{ p | p_c = i_c, \lfloor \frac{p_n}{N/K} \rfloor = \lfloor \frac{i_n}{N/K} \rfloor \right\}.$$

BatchNorm computes fixed summary statistics μ, σ from the training set; for Dynamic BatchNorm and LocalNorm, we recompute these statistics at test-time.

Since LocalNorm provides both multiple independent groups and computes summary statistics at test-time, there are different variants for classifying a novel image at test-time. Given L Groups of size M each, we can evaluate an image M_i at test-time as follows: we can pass M_i through a randomly selected group L_r, use the activations of only this image M_i to compute this group's summary statistics μ_r, σ_r, and then obtain a classification c_i^r (**LN-Single**); we can pass M_i through all L groups, obtaining a set of classifications $\{c_i^l\}$ for each image, and then for each image select the class with the most votes (**LN-Single-Vote**; randomly breaking ties). We can also fill all random groups L_r with M different test-images into a batch, compute the summary statistics on the collective activations, and obtain classifications c_M^r for all M images and $M * L$ images are classified at once (**LN-Batch**), and finally we can pass the batch with M test-images through all L groups obtaining classification $\{c_M^l\}$ for each image, where voting then determines the overall classification (**LN-Voting**). For benchmark testing, **LN-Batch** is the fastest evaluation method, whereas **LN-Voting** will be most accurate; **LN-Single** will be fastest for single-image real-world application and **LN-Single-Voting** most accurate.

Implementation. LocalNorm is easily implemented in auto-different-iation frameworks like **Keras** [3] and **Tensorflow** [1] by adapting a standard batch normalization implementation[1]. For multi-GPUs, LocalNorm can map computational groups on separate GPUs which can accelerate training and allow the training of larger networks. It is moreover straightforward to adapt a model pre-trained with BatchNorm by replacing all BatchNorm layers with LocalNorm layers initialized with the BatchNorm parameters, and then continue training.

4 Image Noise

We test LocalNorm in a Noisy-object classification task where synthetic Gaussian, Poisson and Bernoulli noise is added to images, as in Noise2Noise [11]. All three kinds of independent noise ξ are added on each channel of the image x_c as follows. For **Additive Gaussian Noise (AGN)**, Gaussian noise with zero mean is added to the image on each channel, defined as $\hat{x_c} = x_c(1+\xi), \xi \sim Gaussian(0, \sigma_n)$. **Additive Poisson Noise (APN)** is one of the most dominating noise sources in photographs, and is easily visible in low-light images. APN is a type of zero-mean noise and is hard to remove by pre-processing because it is distributed independently at each channel. Mathematically, APN is computed as $\hat{x_c} = x_c + 255\xi$ or $\hat{x_c} = x_c(1+\xi)\ \xi \sim Poisson(0, \sigma_n)$, where $\sigma_n \in [0, 1]$. **Multiplicative Bernoulli Noise (MBN)** removes some random pixels from the image with probability σ_n. MBN defined by $\hat{x} = x\xi, \xi \sim Bernoulli(\sigma_n)$.

5 Experimental Results

Benchmark Accuracy. We apply LocalNorm to a number of classical benchmarks: CIFAR10 [10], and Stanford Cars [9]. Where useful, we evaluate the benchmarks using all four different types of LocalNorm evaluation methods; when not explicitly mentioned otherwise, the application of LocalNorm refers

[1] Code available at https://github.com/byin-cwi/LocalNorm1.

to LN-Batch evaluation. LocalNorm has as a parameter the number of groups which, for a given batch size, determines the number of images in each group. While we did not extensively optimize for group number, we found that a smallish number of images per group, 4–8, performed best in practice for the batch sizes used in this study.

Results for all three normalization methods (BatchNorm, Dynamic BatchNorm and LocalNorm) are shown in Table 1 using otherwise identical network architectures, where we evaluate LocalNorm with LN-Single, LN-Batch and LN-Voting. We achieve high or near state-of-the-art accuracy on the original datasets, where in 3 our of 4 cases, LN-Voting and LN-Batch outperform BatchNorm and Dynamic BatchNorm. The improvement for CIFAR10 using the VGG architecture with LN-Voting in particular stands out, as accuracy improves from 88.8% to 95.3%; no such improvement is observed for the ResNet32 architecture, and only a slight improvement for the ResNet152 applied to Stanford Cars. We also observe that for the small CIFAR10 images, evaluating test-images using only a single image at a time (LN-Single) gives poor results. Comparing training time, we find that LocalNorm incurs only a small computational cost (10–20%) compared to BatchNorm.

Table 1. The accuracy on original test dataset of each network with various types of normalization on each dataset and for different LocalNorm evaluation methods.

	CIFAR10-VGG	CIFAR10-ResNet	Stanford-cars
BatchNorm	88.83%	**91.74**%	88.17%
Dyn. BatchNorm	87.64%	89.34%	85.34%
LN-Single	65.88%	32.33%	88.39%
LN-Batch	92.07%	91.15%	89.34%
LN-Voting	**95.29**%	91.65%	**89.58**%

For CIFAR10, we use two classical network architectures – VGG19 and ResNet32. The classical **VGG19** network architecture [15] is often used as a baseline to test new network architectures. **Residual Networks**, or **ResNets** [6] achieve robust and scalable accuracy on many machine learning datasets, and ResNet32 (a ResNet with 32 Layers) achieves competitive results on the CIFAR10 dataset [21]. We use a batch size of 128; for LocalNorm, we divide the batch into 8 computational groups with 16 images per group by default.

The Stanford Cars dataset contains 16,185 images of 196 classes of cars taken under various conditions, and each image is large, 224 × 224 pixels, allowing us to compare LocalNorm to BatchNorm and Dynamic BatchNorm when applied to large networks and large images. We use a large ResNet152For ResNet152, we use a pre-trained network[2] and continue training this network with BatchNorm, Dynamic BatchNorm or LocalNorm. For LocalNorm, 16 images are trained as a batch and divided into 4 groups.

[2] https://gist.github.com/flyyufelix/7e2eafb149f72f4d38dd661882c554a6.

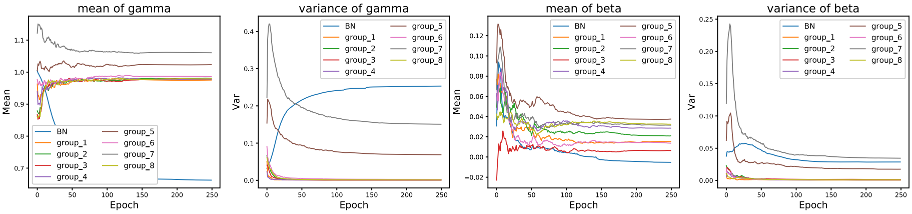


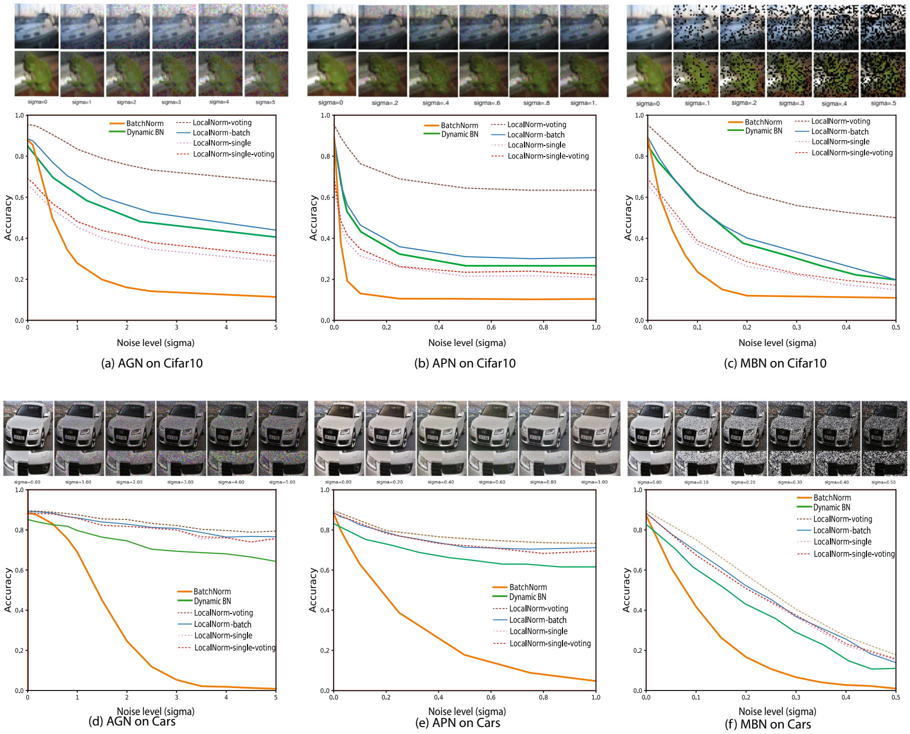
Fig. 5. Development of mean and variance of the scaling parameters γ and β for LocalNorm Groups (group_x) and BatchNorm (BN) during training on CIFAR10. The group-specific scaling parameters do not converge during training, maintaining a diverse scaling regime.

In Fig. 5 we plot the development of mean and variance of the normalization scaling parameters γ and β for LocalNorm and BatchNorm (averaged over all channels) when training VGG19 on CIFAR10 using 8 Groups for LocalNorm. We see that LocalNorm converges to a spread of γ and β values during training, maintaining a diverse normalization regime.

Noisy Image Degradation. To measure noise robustness and noise generalization, we use the networks trained with various normalization methods and the original training dataset, and test them on images degraded with different levels of noise. We evaluated the CIFAR10 and Stanford Cars dataset for all variants of LocalNorm, both where a batch of images is used at test-time (LN-Batch and LN-Voting) to obtain summary statistics, and where only a single image at a time is used at test-time to obtain summary statistics (LN-Single and LN-Single-Voting).

CIFAR10. We tested VGG19 trained on CIFAR10 with various normalization methods on noisy test images degraded with AGN. Figure 6a shows that the accuracy when using BatchNorm decreases rapidly, achieving only 29% accuracy for $\sigma_n = 1$. For the different types LocalNorm evaluation, we find that LN-Batch and LN-Voting drastically improve over BatchNorm and substantially over Dynamic Batchnorm, where for LN-Voting the network accuracy is 83% at $\sigma_n = 1$, almost three times better than the BatchNorm-based network. Evaluation using only single images, LN-Single and LN-Single-Voting, while being more robust to noise, clearly underperform for noiseless data.

Similar observations apply for the other types of noise. For APN, both BatchNorm and LocalNorm's accuracy curve dropped sharply, while LocalNorm still substantially outperforms BatchNorm and Dynamic BatchNorm in general (Fig. 6b). For MBN in Fig. 6c, BatchNorm accuracy drops exponentially and converges to random choice, while LocalNorm's performance decreases slower. Similar findings apply for a ResNet32 network (not shown).



(a) AGN on Cifar10

(b) APN on Cifar10

(c) MBN on Cifar10

(d) AGN on Cars

(e) APN on Cars

(f) MBN on Cars

Fig. 6. Noise effect on CIFAR10 (a–c) and Stanford Cars datasets (d–f). (a–c) Top row illustrates noise-degraded CIFAR10 images for different amounts of AGN, AGN and MBN respectively. Bottom row, line graphs plot corresponding network accuracy on degraded CIFAR10 images using a VGG19 network architecture; (d–f) same noise-degradations applied to the Stanford Cars images using a ResNet152 network.

Stanford Car Dataset. For the images in the Stanford Cars dataset, we find that when testing on noisy images (Fig. 6d), all LocalNorm variants perform very similar, demonstrating that here, a single large image is sufficient to dynamically compute the summary statistics at test-time. LocalNorm maintains a test accuracy over 74% under any tested level of AGN and substantially outperforms Dynamic BatchNorm, while standard BatchNorm accuracy declines sharply to $< 20\%$ for $\sigma_n > 2.5$; similar behavior is observed for APN and BBN (Fig. 6e, f).

CIFAR10-c. The Cifar10-C dataset was published specifically to test network robustness to image corruption [7]. It contains 19 types of algorithmically generated corruptions from noise, blur, weather, and digital categories. To evaluate robustness, the networks are trained on the original CIFAR10 dataset, and evaluated on the corrupted dataset using LN-Voting. The result are shown in Fig. 7b: we find that LocalNorm outperforms both standard BatchNorm and Dynamic BatchNorm everywhere, with the largest improvements observed for those image

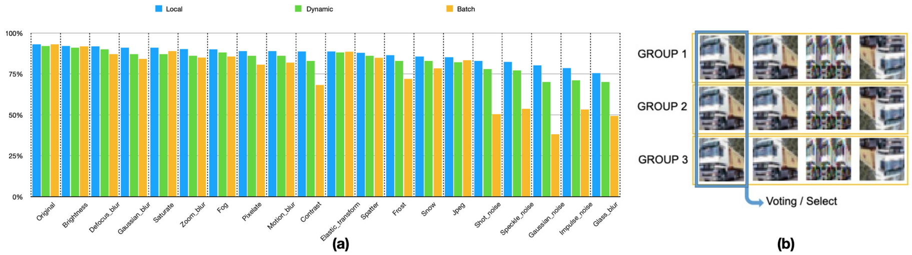


Fig. 7. (a) Comparison of LocalNorm LN-Voting to other normalization on the Cifar10-C dataset, for both Resnet32, for all the different image corruption categories. (b) Single image using data augmentation of the summary statistics through group expansion with rotated images. For a single image in each group (bracketed in green), rotated versions are created and added to the group. Summary statistics are computed for the whole group, while classification is computed for the original single image only. (Color figure online)

corruptions that incur the largest performance drop (Noise, Blur). We also see that LocalNorm improves the accuracy of the VGG-19 network much more than for the ResNet32 network, to the point that VGG becomes substantially more accurate than ResNet32.

ImageNET. We include the ImageNet dataset to test the generalization-ability improvement of VGG16 network with LocalNorm. The network was optimized based on a pre-trained network. The VGG16 with BN achieved 68% top-1 test accuracy, and LocalNorm obtained 67%. In [7], the mCE(mean of Corruption Error) was introduced to represent the generalizability of the network on various corruptions and perturbations. In [7] an mCE value of VGG16-BN of 0.8597 was reported, while the use of LocalNorm-batch improved this to 0.8036 and LocalNorm-voting achieved 0.7045.

Single Image Data Augmentation at Test-time. To improve the performance of LocalNorm-Single and LocalNorm-Single-Voting evaluation on small images, a simple suggestion is to enrich the summary statistics. Here, we propose to augment the data by adding rotated versions of the image along the width- and channel-axis (ROT90 clockwise) to the computation group to enrich the summary statistics, as shown in Fig. 7b. Adding such data will increase stability of the mean and variance derived from the computational group. During classification, the rotated images are only used to compute the summary statistics and the classification is determined for the original image only; classification can be done by either voting the prediction of each group or selecting a prediction randomly as the final result.

We find that this approach drastically improves LN-Single and LN-Single-Voting for the small images of CIFAR10. As show in Fig. 8 for AGN, thus enhancing the summary statistics for single image evaluation improves robustness and noiseless accuracy to the same level as LN-Batch - similar observations apply for

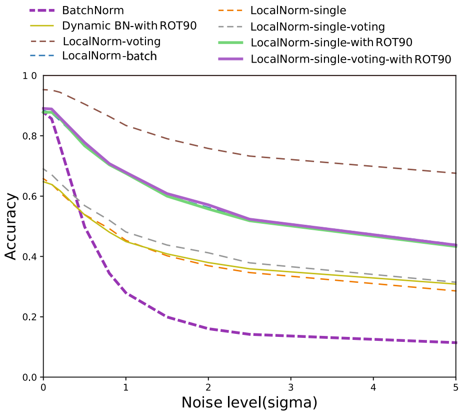


Fig. 8. Rotation-augmented summary statistics. The solid 'ROT90' lines indicate rotation data-augmented evaluation; the LN-Single-ROT90 and LN-Single-Voting-ROT90 curves now overlap with the LN-batch curve. Rotation-augmentation for Dynamic Batch Normalization on single images is insufficient (solid yellow line). (Color figure online)

APN and MBN image degradation (not shown). Applying rotation-augmentated Dynamic BatchNorm to single CIFAR10 images improves performance evaluation only modestly (DynamicBN in Fig. 8). While adding rotated images at test-time to the groups implies a substantial increase in computational cost, there is no cost to training, and evaluation on such small images tends to be very fast.

6 Conclusion

We develop an effective and robust normalization layer–LocalNorm. LocalNorm regularizes the Normaliation layer during training, and includes a dynamic computation of the Normalization layer's summary statistics during test-time. The key insight here is that out-of-sample conditions, like noise degradation, will shift the summary statistics of an image, and the LocalNorm approach makes a DNN more robust to such shifts.

We demonstrate the effectiveness of the approach on classical benchmarks, including both small and large images, and find that LocalNorm decisively outperforms classical Batch Normalization and dynamic variants like Dynamic Batch Normalization. We show that computing LocalNorm only has a limited computational cost with respect to training time, of order 10–20%. LocalNorm furthermore can be evaluated on batches of test-images, and, for large enough images, also on single images passed through only a single group, then incurring the same evaluation cost as Batch Normalization. To enable the evaluation of small images one-at-a-time at test-time, we demonstrated the addition of rotated images to the groups as a form of data augmentation to improve the summary statistics. For more general type of image distortions, we find that using LocalNorm also makes networks substantially more robust, as evidenced by the results on the CIFAR10-c dataset.

Intriguingly, the ability of regularized normalization layers to calculate sufficient summary statistics from single large-enough inputs rather than batches also suggests that correlates may be found in biological systems, where access to batch-statistics seem implausible but multiple parallel pathways may facilitate multiple independent normalizations as in LocalNorm.

Acknowledgments. BY is funded by the NWO-TTW Programme "Efficient Deep Learning" (EDL) P16-25.

References

1. Abadi, M., et al.: Tensorflow: a system for large-scale machine learning. OSDI. **16**, 265–283 (2016)
2. Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. arXiv preprint arXiv:1607.06450 (2016)
3. Chollet, F., et al.: Keras. https://github.com/fchollet/keras (2015)
4. Du, Y., Zhen, X., Shao, L., Snoek, C.G.: Metanorm: learning to normalize few-shot batches across domains. In: International Conference on Learning Representations (2020)
5. Geirhos, R., Temme, C.R., Rauber, J., Schütt, H.H., Bethge, M., Wichmann, F.A.: Generalisation in humans and deep neural networks. In: NeurIPS, pp. 7549–7561 (2018)
6. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, vol. 2106, pp. 770–778 (2016)
7. Hendrycks, D., Dietterich, T.: Benchmarking neural network robustness to common corruptions and perturbations (2018)
8. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)
9. Krause, J., Stark, M., Deng, J., Fei-Fei, L.: 3D object representations for fine-grained categorization. In: 4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13), Sydney, Australia (2013)
10. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Tech. rep, Citeseer (2009)
11. Lehtinen, J., et al.: Noise2noise: learning image restoration without clean data. CoRR (2018)
12. Li, Y., Wang, N., Shi, J., Hou, X., Liu, J.: Adaptive batch normalization for practical domain adaptation. Pattern Recogn. **80**, 109–117 (2018)
13. Luo, P., Ren, J., Peng, Z.: Differentiable learning-to-normalize via switchable normalization. arXiv preprint arXiv:1806.10779 (2018)
14. Motwani, M.C., Gadiya, M.C., Motwani, R.C., Harris, F.C.: Survey of image denoising techniques. In: Proceedings of GSPX, pp. 27–30 (2004)
15. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR https://arxiv.org/abs/1409.1556 (2014)
16. Summers, C., Dinneen, M.J.: Four things everyone should know to improve batch normalization. In: International Conference on Learning Representations (2019)

17. Ulyanov, D., Vedaldi, A., Lempitsky, V.S.: Improved texture networks: maximizing quality and diversity in feed-forward stylization and texture synthesis. CoRR https://arxiv.org/abs/1701.02096 (2017)
18. Ulyanov, D., Vedaldi, A., Lempitsky, V.S.: Instance normalization: the missing ingredient for fast stylization. CoRR https://arxiv.org/abs/1607.08022 (2016)
19. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.A.: Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. JMLR **11**(Dec), 3371–3408 (2010)
20. Wu, Y., He, K.: Group normalization. arXiv preprint arXiv:1803.08494 (2018)
21. Zhang, G., Wang, C., Xu, B., Grosse, R.: Three mechanisms of weight decay regularization (2018)
22. Zhang, X., Lu, Y., Liu, J., Dong, B.: Dynamically unfolding recurrent restorer: a moving endpoint control method for image restoration. CoRR https://arxiv.org/abs/1805.07709 (2018)