Comparing normalization in conditional computation tasks

Anonymous Authors[1]

Abstract

Batch normalization has been widely used to improve optimization in deep neural networks. While the uncertainty in batch statistics can act as a regularizer, using these dataset statistics specific to the training set, hurts generalization in certain tasks. Recently, group normalization has been shown to yield similar or better performance than batch normalization. In this work, we study whether the conditional formulation of group normalization can improve generalization compared to conditional batch normalization. We compare performance on the tasks of visual question answering, few-shot learning, and conditional image generation.

1. Introduction

Recent studies propose different benchmarks for evaluating task specific models for their generalization capacity. While in this paper, we only focus on visual question answering (VQA), few-shot learning and generative models, any improvement in this direction can be beneficial for other domains. Here specifically, we explore the effect of activation normalization techniques used in deep neural networks, in particular Batch Normalization (BN) and Group Normalization (GN). We compare the performance of deep neural networks using different normalization schemes in each task. We wish for the community to take advantage of the results of this exploration for the best choice of normalization, in terms of the benefit and disadvantage of each type.

2. Background

Several normalization methods have been proposed to address the problem of covariate shift in deep neural networks (Ioffe & Szegedy, 2015; He et al., 2016; Lei Ba et al., 2016).

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the ICML 2019 Workshop "Understanding and Improving Generalization in Deep Learning". Do not distribute.

To stabilize the range of variation of network activations x_i, methods such as BN first normalize the activations:

$$\hat{x}_i = \frac{1}{\sigma_i}\left(x_i - \mu_i\right), \tag{1}$$

with mean μ_i and standard deviation σ_i:

$$\mu_i = \frac{1}{m}\sum_{k\in\mathcal{S}_i} x_k, \quad \sigma_i = \sqrt{\frac{1}{m}\sum_{k\in\mathcal{S}_i}\left(x_k - \mu_i\right)^2 + \epsilon} \tag{2}$$

Then, to increase representational power, the normalized activations are scaled and shifted using learnable parameters γ and β:

$$y_i = \gamma\hat{x}_i + \beta \tag{3}$$

In the case of 2D feature maps, i is a four-dimensional index $i = (i_N, i_C, i_H, i_W)$, whose elements index the features along the batch, channel, height and weight axis, respectively. In BN, $\mathcal{S}_i$ in Equation 2 is defined as

$$\text{BN} \implies \mathcal{S}_i = \{k | k_C = i_C\}, \tag{4}$$

the set of all pixels sharing the same channel axis, resulting in μ and σ being computed along the (N, H, W) axes.

However, (Lei Ba et al., 2016) points out that BN is highly affected by batch size, and introduces Layer Normalization (LN), which normalizes activations within each layer, without any dependence on batch statistics. (Ulyanov et al., 2016) introduce Instance Normalization (IN) in the context of image stylization. It is essentially normalization at every channel in each layer, a more extreme version of LN. Then (Wu & He, 2018) introduced GN as a trade-off between LN and IN. It aptly summarizes LN, IN and GN using the set notation as:

$$\text{GN} \implies \mathcal{S}_i = \{k | k_N = i_N, \lfloor \frac{k_C}{C/G} \rfloor = \lfloor \frac{i_C}{C/G} \rfloor\}, \tag{5}$$

i.e. normalization is performed per sample, within groups of size G along the channel axis. For LN, $G =$ channel width, while in IN, $G = 1$.

Conditional Batch Normalization (CBN) (De Vries et al., 2017) is a conditional variant of BN, where the learnable

re-normalization parameters γ and β are functions of some condition to the network, such as the class label. De Vries et al. introduced CBN for the task of VQA in (De Vries et al., 2017; Perez et al., 2018). (Dumoulin et al., 2017) introduce Conditional Instance Normalization (CIN), a conditional variant of IN similar to CBN, replacing BN with IN. In our experiments, we also use the conditional variants of LN and GN, i.e. Conditional Layer Normalization (CLN), and Conditional Group Normalization (CGN).

It was recently concluded that internal covariance shift has little to do with the success of BN (Santurkar et al., 2018). Instead, BN smooths the optimization landscape. Hence, it is useful to understand whether the advantages of BN carry over to other normalization techniques.

Recently, (Kurach et al., 2018) conducted an empirical study on only image generation in an adversarial setting. We conduct a more specific study on normalization techniques in a greater variety of tasks.

2.1. Visual Question Answering

The task here is to answer a question based on an image. The way this task is usually solved is that the question is fed to a parametric model conditioned on the image, which then provides an answer.

One recent successful model for VQA is the Feature-wise Linear Modulation (FiLM) architecture (Perez et al., 2018). The most useful feature introduced here to solve the problem was the conditional normalization technique. We experiment with replacing BN with the recently proposed GN (Wu & He, 2018), to investigate how their performance differs.

2.2. Few-Shot Learning

The field of few-shot learning, i.e. learning to classify given only a small set of samples, has recently received a lot of interest. In episodic M-way, k-shot classification tasks, meta-learning models, learn to adapt a classifier given multiple M-class classification tasks, with k support samples for each class. The meta-learner thus has to solve the problem of generalizing between these tasks given the limited number of training samples. In this work, we experiment with the recently proposed Task dependent adaptive metric (TADAM) (Oreshkin et al., 2018), which leverages CBN, which we replace with CGN to compare performances.

2.3. Conditional Image Generation

One of the most successful ways of generating images in recent times is using Generative Adversarial Networks (GAN)s (Goodfellow et al., 2014). More recently, the generators that worked best stack multiple ResNet-style (He et al., 2016) architectural blocks, involving two ReLU-CBN-Conv operations with an upsampling operation. These blocks are followed by a ReLU-BN-Conv operation to transform the last features into the shape of an image (with 3 channels).

Such models include Wasserstein GAN using gradient penalty (WGAN-GP) (Gulrajani et al., 2017), Self-Attention GAN (SAGAN) (Zhang et al., 2018) and BigGAN (Brock et al., 2019). In all our experiments, we simply replaced the BN modules in the networks with the other types of normalization, and compared.

However, we found it difficult to reproduce the results of the SAGAN and BigGAN. Since the architectures of the three models are similar, and all we changed was the normalization technique, we expect our results on WGAN-GP to carry over to these models as well.

3. Experiments

3.1. Visual Question Answering

To study how CGN performs in VQA tasks, we experimented with several small variations of the FiLM architecture (Perez et al., 2018) on three VQA datasets: CLEVR Compositional Generalization Test (CLEVR-CoGenT) (Johnson et al., 2017a), Figure Question Answering (FigureQA) (Kahou et al., 2017). and the recently introduced Spatial Queries On Object Pairs (SQOOP) (Bahdanau et al., 2018).

The original architecture in (Perez et al., 2018) consists of a sequence of four FiLM-ed residual block (He et al., 2016). The scale and shift parameters are generated as an affine transform of a gated recurrent unit (GRU) question embedding. The output of the last residual block is fed to a classifier, which consists of a layer of 512 1×1 convolutions, global max-pooling, followed by a fully-connected ReLU layer using BN and the softmax layer, which outputs the probability of each possible answer. We train the three variants with CGN:

- all conditional and regular BN layers are replaced with corresponding conditional or regular GN layers.
- all CBN layers are replaced with CGN, regular BN layers are left unchanged.
- all CBN layers are replaced with CGN, regular BN layers are left unchanged, except the fully-connected hidden layer in the classifier, which doesn't use normalization.

For all datasets, the input to the residual network are image features extracted from layer *conv4* of a ResNet-101 (He et al., 2016), pretrained on ImageNet(Russakovsky et al., 2015), as in FiLM (Perez et al., 2018) and other works (Johnson et al., 2017a;b).

All models are trained using the Adam optimizer (Kingma & Ba, 2014) with a learning rate of $3e{-}4$, ϵ of $1e{-}5$ and weight decay coefficient of $1e{-}5$, with all other hyperparameters the same as in the original papers.

3.2. Few-Shot Learning

In this section, we compare CGN with CBN in the few-shot classification model TADAM (Oreshkin et al., 2018). We experiment with two episodic five-shot, five-class classification tasks, as in Oreshkin et al. (2018). TADAM is a metric-based few-shot classifier. It uses a similarity metric based on embeddings from a residual network to compare data samples to class templates. The residual network employs CBN conditioned on a task embedding, which is simply the average of class embeddings for the given task. For the GN version we replaced all conditional and regular BN layers with their corresponding GN version. For a complete description of the experimental setup, including all other hyperparameters, we refer the reader to Oreshkin et al. (2018).

3.3. Conditional Image Generation

We check the effect of CBN, CGN, CLN and CIN on image generation. We use WGAN-GP (Gulrajani et al., 2017) on the CIFAR10 dataset (Krizhevsky, 2009). We replicated the architecture mentioned in the original paper. This is the case of CBN. For the other cases, we replace every BN in the model with the respective normalization technique. We use 4 groups in the case of CGN. We use the same optimization mentioned in the original paper - learning rate of $2e{-}4$ for both Generator and Discriminator, 5 discriminator updates per generator update, and Adam optimizer on a single GPU (NVIDIA P100) with a batch size of 64.

For each case of normalization, we calculate two scores that are widely used in the community to check the quality of image generation — Inception Score (IS) (Salimans et al., 2016) and Fréchet Inception Distance (FID) (Heusel et al., 2017). We use public code to calculate IS [1] and FID[2]. Numbers for true data may differ from original papers since these are PyTorch (Paszke et al., 2017) implementations, while the papers use TensorFlow (Abadi et al., 2015). But, we compare the same metric for true and generated data.

We first calculate the IS of the true images of CIFAR-10, for each class separately. Then, during training of the model, we sample images from the generator at regular intervals, and calculate the IS and FID of those images for each class separately. This allows us to see the effect of the different normalization techniques on the conditional generation process. We average our results from multiple runs.

[1]https://github.com/sbarratt/inception-score-pytorch
[2]https://github.com/mseitzer/pytorch-fid

4. Results

4.1. Visual Question Answering

Tables 1, 2 3 list our results in VQA. We see that in all cases, models using GN perform very similar to models using BN. We can perhaps conclude that using GN instead of BN has no critical effect on the performance. Hence, it might be preferable to use GN in cases where the batch statistics are unreliable to compute, for example in cases of smaller batch sizes.

4.2. Few-Shot Learning

Table 4 shows our results for few-shot learning using TADAM and it's CGN version. We see here too that using GN instead of BN yields relatively similar performance.

Table 1. Classification accuracy on CLEVR-CoGenT *val2* averaged over three runs.

Model	Accuracy (%)
FiLM (Perez et al., 2018)	75.5 ± 0.7
CGN (all GN)	75.7 ± 0.3
CGN (BN in stem, classifier no norm)	75.7 ± 0.6
CGN (BN in stem and classifier)	$\mathbf{75.8 \pm 0.5}$

Table 2. Classification accuracy on FigureQA *validation2* averaged over three runs.

Model	Accuracy (%)
FiLM (Perez et al., 2018)	$\mathbf{91.6 \pm 0.1}$
CGN (all GN)	91.3 ± 0.4
CGN (BN in stem, classifier no norm)	91.1 ± 0.2
CGN (BN in stem and classifier)	91.3 ± 0.5

4.3. Conditional Image Generation

We computed the IS and FID of the samples generated by the model during training for each class separately, to compare the effects of the different normalization techniques. We see in Figure 1 that it is not conclusive whether any one normalization technique is strictly better than the others.

We also see that it is easier for the model to generate certain classes such as automobile and truck, than other classes, and that this is true for all cases of normalization. We conclude that the type of normalization does not have a critical effect on the conditional generation process.

This would imply that, at least in the space of generative models such as GANs operating at lower batch sizes, using BN or GN would not lead to significant change in the quality of generated images. So it might be advantageous to use GN instead of BN, since GN is more computationally efficient, and not affected by batch size.

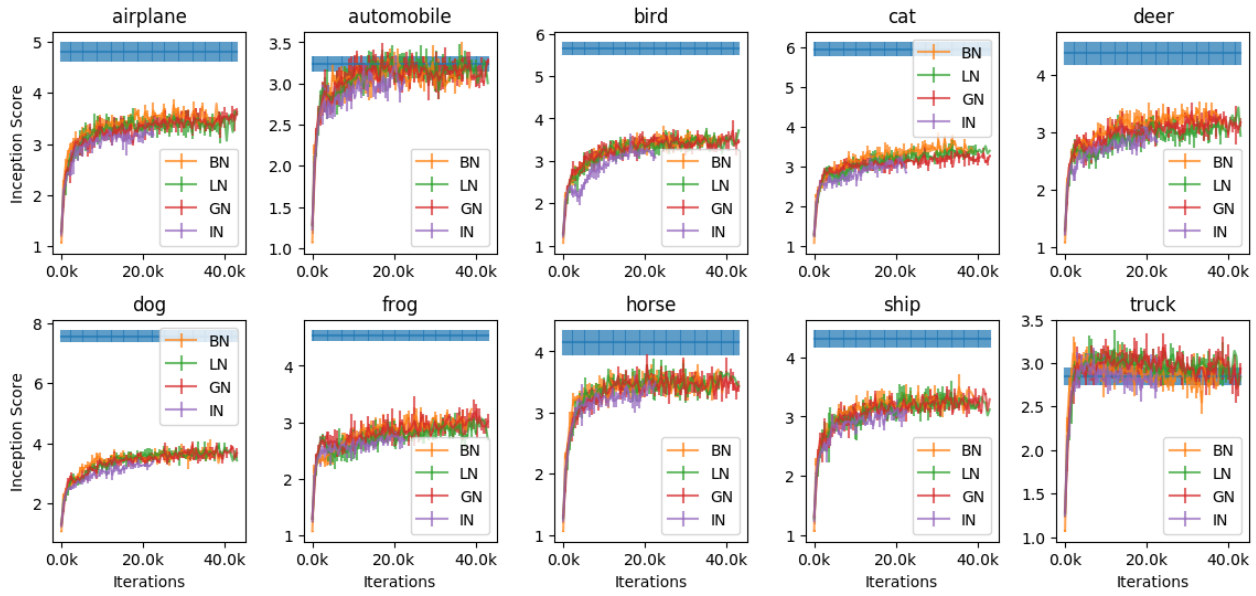

Figure 1. Class-wise Inception Score (IS) of samples generated by WGAN-GP model trained on CIFAR-10. (Blue: IS of true data)

Table 3. Classification accuracy on SQOOP averaged over three runs.

Dataset	Model	Accuracy (%)
1 rhs/lhs	FiLM (BN)	72.4 ± 0.5
	CGN (all GN)	74.0 ± 2.8
	CGN (BN — no norm)	73.8 ± 0.3
	CGN (BN — classifier)	$\mathbf{74.9 \pm 3.9}$
2 rhs/lhs	FiLM (BN)	85.0 ± 4.2
	CGN (all GN)	$\mathbf{86.7 \pm 6.3}$
	CGN (BN — no norm)	83.1 ± 0.4
	CGN (BN — classifier)	85.9 ± 5.3
4 rhs/lhs	FiLM (BN)	97.0 ± 1.9
	CGN (all GN)	91.4 ± 0.3
	CGN (BN — no norm)	91.6 ± 1.9
	CGN (BN — classifier)	$\mathbf{99.5 \pm 0.2}$
35 rhs/lhs	FiLM (BN)	$\mathbf{99.8 \pm 0.1}$
	CGN (all GN)	99.8 ± 0.1
	CGN (BN — no norm)	99.8 ± 0.1
	CGN (BN — classifier)	99.8 ± 0.2

Table 4. Five-way five-shot classification accuracy on Fewshot-CIFAR100 (Oreshkin et al., 2018) and Mini-Imagenet (Vinyals et al., 2016) averaged over three runs, where TADAM (BN) is from (Oreshkin et al., 2018)

Dataset	Model	Accuracy (%)
FC100	TADAM (BN)	$\mathbf{53.0 \pm 0.6}$
	TADAM (GN)	52.8 ± 0.5
Mini-Imagenet	TADAM (BN)	$\mathbf{76.4 \pm 0.5}$
	TADAM (GN)	74.1 ± 0.4

5. Conclusion

We presented a set of experiments for VQA, few-shot learning and image generation tasks using models that rely on BN for conditional computation. As the performance of BN heavily depends on the batch size, and on how well training and test statistics match, we replaced BN with GN. We experimentally showed that CGN can be a good replacement for CBN for conditional computation, without losing performance in some cases. We hope that this study serves as a reference for research in simple general purpose methods for conditional computation . In most experiments, we simply replaced BN with GN and did not use extensive hyper-parameter optimization. We also did not experiment with different batch sizes, so as to have a fair comparison. In settings with large input spaces and conditioning information, such as video question answering (Mun et al., 2017; Tapaswi et al., 2016), CGN might find an application highlighting its strengths compared with CBN. Studying such tasks remains as future work. As the authors of Wu & He (2018) point out, GN lacks the regularization effect that BN has due to noisy batch statistics. They propose to explore combinations of GN with an appropriate regularization scheme to improve performance. We see an investigation of regularization of both GN and CGN as a worthwile research direction.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL http://tensorflow.org/. Software available from tensorflow.org.

Bahdanau, D., Murty, S., Noukhovitch, M., Nguyen, T. H., de Vries, H., and Courville, A. Systematic generalization: What is required and can it be learned? *arXiv preprint arXiv:1811.12889*, 2018.

Brock, A., Donahue, J., and Simonyan, K. Large scale gan training for high fidelity natural image synthesis. *CoRR*, abs/1809.11096, 2019.

De Vries, H., Strub, F., Mary, J., Larochelle, H., Pietquin, O., and Courville, A. C. Modulating early visual processing by language. In *Advances in Neural Information Processing Systems*, pp. 6594–6604, 2017.

Dumoulin, V., Shlens, J., and Kudlur, M. A learned representation for artistic style. *CoRR*, abs/1610.07629, 2017.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. C., and Bengio, Y. Generative adversarial nets. In *NIPS*, 2014.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pp. 57695779, 2017.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NIPS*, 2017.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Johnson, J., Hariharan, B., van der Maaten, L., Fei-Fei, L., Zitnick, C. L., and Girshick, R. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *CVPR*, 2017a.

Johnson, J., Hariharan, B., van der Maaten, L., Hoffman, J., Fei-Fei, L., Lawrence Zitnick, C., and Girshick, R. Inferring and executing programs for visual reasoning. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2989–2998, 2017b.

Kahou, S. E., Michalski, V., Atkinson, A., Kádár, Á., Trischler, A., and Bengio, Y. Figureqa: An annotated figure dataset for visual reasoning. *arXiv preprint arXiv:1710.07300*, 2017.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Krizhevsky, A. Learning multiple layers of features from tiny images. 2009.

Kurach, K., Lucic, M., Zhai, X., Michalski, M., and Gelly, S. The gan landscape: Losses, architectures, regularization, and normalization. *CoRR*, abs/1807.04720, 2018.

Lei Ba, J., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Mun, J., Hongsuck Seo, P., Jung, I., and Han, B. Marioqa: Answering questions by watching gameplay videos. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2867–2875, 2017.

Oreshkin, B., López, P. R., and Lacoste, A. Tadam: Task dependent adaptive metric for improved few-shot learning. In *Advances in Neural Information Processing Systems*, pp. 721–731, 2018.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017.

Perez, E., Strub, F., De Vries, H., Dumoulin, V., and Courville, A. Film: Visual reasoning with a general conditioning layer. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3): 211–252, 2015.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X., and Chen, X. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, 2016.

Santurkar, S., Tsipras, D., Ilyas, A., and Madry, A. How does batch normalization help optimization? In *NeurIPS*, 2018.

Tapaswi, M., Zhu, Y., Stiefelhagen, R., Torralba, A., Urtasun, R., and Fidler, S. Movieqa: Understanding stories in movies through question-answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4631–4640, 2016.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. S. Instance normalization: The missing ingredient for fast stylization. *CoRR*, abs/1607.08022, 2016.

Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pp. 3630–3638, 2016.

Wu, Y. and He, K. Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 3–19, 2018.

Zhang, H., Goodfellow, I. J., Metaxas, D. N., and Odena, A. Self-attention generative adversarial networks. *arXiv:1805.08318*, 2018.