An Empirical Study of Batch Normalization and Group Normalization in Conditional Computation

Vincent Michalski
Mila, Université de Montréal
michals@mila.quebec

Vikram Voleti
Mila, Université de Montréal

Samira Kahou
Mila, McGill University

Anthony Ortiz
University of Texas, El Paso

Pascal Vincent
Mila, Université de Montréal

Chris Pal
Mila, Polytechnique Montreal

Doina Precup
Mila, McGill University

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, impairs generalization in certain tasks. Recently, alternative methods for normalizing feature activations in neural networks have been proposed. Among them, group normalization has been shown to yield similar performance and in some domains it also outperforms batch normalization. All these methods utilize a learned affine transformation after normalization to increase representational power. Methods used in conditional computation instead define these affine transformation parameters as learnable functions of conditional information. 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

In machine learning, the parameters of a model are typically optimized using a fixed training set. The model is then evaluated on a separate partition of the data to estimate its generalization capability. In practice, even under the i.i.d. assumption, the distribution of these two finite sets can *appear* quite different to the learning algorithm, making it challenging to achieve strong and robust generalization. This difference is often the result of the fact that a training set of limited size cannot adequately cover the cross-product of all relevant factors of variation. This issue can be addressed by making strong assumptions that simplify discovering a family of patterns from limited data. For instance Bahdanau et al. [1] present a synthetic relational reasoning task, in which a Neural Module Network (NMN) [2] with fixed tree structure clearly outperforms models without such a prior.

Recent studies propose different benchmarks for evaluating task specific models for their generalization capacity [3, 4, 1]. While in this paper, we focus on visual question answering (VQA), few-shot learning and generative models, any improvement in this direction can be beneficial for other domains such as reinforcement learning. Here specifically, we explore the effect of activation normalization techniques used in deep neural networks, in particular Batch Normalization (BN) [5] and Group Normalization (GN) [6]. Many of the best-performing models for each of the considered tasks are deep neural networks, with normalization layers at regular intervals. We compare the performances of models using different normalization schemes for each task.

BN, proposed by Ioffe and Szegedy [5], normalizes the features of a neural network by the mean and variance computed within a (mini-)batch. It has been shown that BN eases optimization and enables very deep networks to converge in part by acting as a regularizer [7]. Most normalization schemes, including BN, learn a per-channel affine transformation, where scaling and shifting parameters are learned during training, to compensate for the possible loss of representational capacity.

Conditional Batch Normalization (CBN) for the task of VQA was introduced in De Vries et al. [8], Perez et al. [9]. Instead of learning scaling and shifting parameters directly from the minibatch training statistics, CBN defines them as learned functions of a conditioning input. Scale and shift parameters for each feature are generated and applied to each feature via an affine transformation. This type of feature transformation has been successfully applied to neural style transfer [10] as well. Wu and He recently proposed GN [6] as a new normalization scheme. GN divides feature maps into groups and normalizes the features within each group across the entire spatial dimensions.

Recently, Kurach et al. [11] conducted a more general empirical study on image generation in an adversarial setting. We conduct a more specific study on normalization techniques, also involving experiments in conditional image generation.

The contribution of the present work is a careful study of feature-wise context-based modulation on normalization methods in a variety of tasks. 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

2.1 Normalization Layers

Several normalization methods have been proposed to address the problem of covariate shift in deep neural networks [5, 12, 6]. To stabilize the range of variation of network activations x_i, methods such as BN first normalize the activations:

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

of an input x_i using 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 width 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. [12] 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. [13] 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 and He [6] introduced GN as a trade-off between LN and IN; and succinctly summarize LN, IN and GN using the set notation as:

$$\text{LN} \implies \mathcal{S}_i = \{k | k_N = i_N\}, \tag{5}$$

i.e. normalization is performed per sample, within each layer,

$$\text{IN} \implies \mathcal{S}_i = \{k | k_N = i_N, k_C = i_C\}, \tag{6}$$

i.e. normalization is performed per sample, per channel,

$$\text{GN} \implies \mathcal{S}_i = \{k | k_N = i_N, \lfloor \frac{k_C}{C/G} \rfloor = \lfloor \frac{i_C}{C/G} \rfloor \}, \tag{7}$$

i.e. normalization is performed per sample, within groups of size G along the channel axis.

CBN [8] is a conditional variant of BN, where the learnable "de-normalization" parameters γ and β are functions of some conditioning input to the network, such as a class label. Dumoulin et al. [10] introduce Conditional Instance Normalization (CIN), a conditional variant of IN similar to CBN, replacing BN with IN. In our experiments, we also explore conditional variants of LN and GN.

It was recently concluded that "internal covariate shift" has little to do with the success of BN [7]. Instead, BN smooths the optimization landscape. Hence, it is useful to understand whether some of the advantages of BN carry over to other normalization techniques.

2.2 Visual Question Answering

In VQA [14, 15], the task is to answer a question about an image. The way this task is usually approached 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 [9]. The crucial ingredient in that model is that the affine transformation parameters of BN layers of a ResNet are generated by a network reading the question. Here, we experiment with replacing BN with the recently proposed GN [6] to compare their behavior in this application of conditional computation [16].

2.3 Few-Shot Classification

The task of few-shot classification consists in the challenge of classifying data given only a small set of support samples. In episodic M-way, k-shot classification tasks, meta-learning models [17] 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) [18]. It belongs to the family of meta-learners, that employ nearest neighbor classification within a learned embedding space. In the case of TADAM, the network providing this embedding uses CBN, which we replace with Conditional Group Normalization (CGN) to compare performances.

2.4 Conditional Image Generation

One of the most successful ways of generating images in recent times is using Generative Adversarial Networks (GAN)s [19]. Subsequently, several variants of GAN [20, 21] have been proposed to condition the image generation process on the class label. More recently, the generators that worked best stack multiple ResNet-style [22] 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 three channels).

Such models are typically trained as Wasserstein GANs using gradient penalty (WGAN-GP) [23]. This paper gave mathematically sound arguments for an optimization framework, which we have followed as well. More recently, Self-Attention GAN (SAGAN) [24] and BigGAN [25] also use similar architectures as described above, with some important changes. SAGAN inserts a self-attention mechanism to attend over the important parts of features during the generation process. In addition, it uses spectral normalization [26] to stabilize training. BigGAN uses a similar architecture to SAGAN, but increases the batch size and channel widths, and makes some architectural changes to save memory and time. Both these models have been tested on generating natural images of larger size. In all our experiments, we simply replaced the BN modules in the networks with other types of normalizations, and compared the performance metrics.

3 Experiments

We perform experiments on datasets in several domains to compare the conditional versions of BN and GN[1]. Section 3.1 describes the setup of our VQA experiments, Section 3.2 gives a high-level description of the few-shot classification experiments and Section 3.3 provides the context for the conditional image generation experiments.

3.1 Visual Question Answering

To study how CGN performs in VQA tasks, we experimented with several small variations of the FiLM architecture [9] on four VQA datasets: Compositional Language and Elementary Visual Reasoning Dataset (CLEVR), CLEVR Compositional Generalization Test (CLEVR-CoGenT) [3], Figure Question Answering (FigureQA) [4] and the recently introduced Spatial Queries On Object Pairs (SQOOP) [1].

The original architecture in Perez et al. [9] consists of a sequence of four residual blocks [22] with CBN layers. The scale and shift parameters of these layers are generated as an linear transformation of a gated recurrent unit (GRU) as the 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 following three variants that include CGN[2]:

1. all conditional and regular BN layers are replaced with corresponding conditional or regular GN layers.
2. all CBN layers are replaced with CGN, regular BN layers are left unchanged.
3. 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.

Besides the described changes in the normalization layers, the architecture and hyperparameters are the same as used in Perez et al. [9] for all experiments, except for SQOOP where they are the same as in Bahdanau et al. [1]. The only difference is that we set the constant ϵ in to $1\mathrm{e}{-5}$ to improve numerical stability[3]. For SQOOP, the input to the residual network are the raw image pixels. For all other networks, the input is features extracted from layer *conv4* of a ResNet-101 [22], pre-trained on ImageNet[27], as was done in FiLM [9] and other works [3, 28].

Three of the datasets, CLEVR-CoGenT, FigureQA and SQOOP, require models to generalize to novel compositions of objects or attributes not seen during training. For instance, materials and colors of rendered objects in CLEVR-CoGenT are sampled from mutually exclusive sets for training (*condition A*) and testing (*condition B*). We perform early stopping on the respective validation sets which share attributes with the training set and report performances on data from condition B. For CLEVR, we did not perform early stopping and instead report validation accuracy after 80 epochs[4].

3.2 Few-Shot Learning

In this section, we compare CGN with CBN in the few-shot classification model TADAM [18]. We experiment with two episodic five-shot, five-class classification tasks, as in Oreshkin et al. [18]. 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. Class embeddings are the average image embeddings associated with the respective class inside the sample set. 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. [18].

[1]The code for all our experiments will be released soon

[2]We always set the number of groups to 4, as the authors of Wu and He [6] showed that this hyperparameter does not have a large influence on the performance. This number was selected using uniform sampling from the set $\{2, 4, 8, 16\}$.

[3]The authors of Perez et al. [9] confirmed occasional gradient explosions with the original setting of $1\mathrm{e} - 8$

[4]We didn't receive test results on time for the submission.

3.3 Conditional Image Generation

We check the effect of CBN, CGN, Conditional Layer Normalization (CLN) and CIN on image generation. We use WGAN-GP [23] on the CIFAR-10 dataset [29]. 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. As in the other experiments, we set the number of groups for CGN to 4. We use the same optimization setup as 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, in addition to a qualitative check of the samples generated, we calculate two scores that are widely used in the community to check the quality of image generation: Inception Score (IS) [30] and Fréchet Inception Distance (FID) [31]. We use publicly available code to calculate IS[5] and FID[6]. Numbers for true data may differ from the original papers since these are PyTorch [32] implementations, while the original papers used TensorFlow [33]. But, we compare the same metric for true and generated data. IS is meant to measure the natural-ness of an image by checking the embedding of the generated images on a pre-trained Inception network. Although the suitability of the IS for this purpose has been rightfully put into question [34], it continues to be used frequently. FID measures how similar two sets of images are, by computing the Fréchet distance between two multivariate Gaussians fitted to the embeddings of the images from the two sets. The embeddings are obtained from a pre-trained Inception network. In this case, we measure the distance between the real CIFAR-10 images, and the generated ones. This is a better metric than IS, since there is no constraint on the images being natural, and it is able to quantify diversity in the generated images along with similarity to the real images.

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 with different seeds.

We performed the same experiments using the more recent SAGAN [24] and BigGAN [25] on the ImageNet dataset [35], as these architectures could be implemented with a few small changes to our basic WGAN-GP code. However, we were unable to reproduce the results reported in the papers due our limited batch size and GPU capability. Notably, BigGAN used batch sizes from 256 to 2048 with the help of multiple TPUs. We limited our experiments to single-GPU models with a batch size of 64. Recall that our goal is not to compete with state-of-the-art large models, but to study the effect of normalization techniques. Nevertheless, due to similarity in structure, it is likely that the conclusions we will draw from WGAN-GP experiments may transfer to larger SAGAN and BigGAN.

4 Results

4.1 Visual Question Answering

Tables 1, 2, 3 and 4 list our results in VQA. We see that in all cases, models using GN achieve a very similar performance to models using BN. We observe that the performance does not differ significantly between CGN and CBN. 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 5 show our results for few-shot learning using TADAM and its CGN version. We see here too that using GN instead of BN yields relatively similar performance. Note, that we simply reuse the hyperparameters from Oreshkin et al. [18], which were tuned for CBN.

[5]https://github.com/sbarratt/inception-score-pytorch

[6]https://github.com/mseitzer/pytorch-fid

4.3 Conditional Image Generation

We compute 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 Figures 1 and 2 that it is not conclusive whether any one normalization technique is strictly better than the others.

Table 1: Classification accuracy on CLEVR *val* averaged over three runs.

Model	Accuracy (%)
FiLM [9]	$\mathbf{97.527 \pm 0.248}$
CGN (all GN)	97.430 ± 0.392
CGN (BN in stem, classifier no norm)	97.242 ± 0.239
CGN (BN in stem and classifier)	97.432 ± 0.521

Table 2: Classification accuracy on CLEVR-CoGenT *val2* averaged over three runs.

Model	Accuracy (%)
FiLM [9]	75.539 ± 0.671
CGN (all GN)	75.758 ± 0.356
CGN (BN in stem, classifier no norm)	75.703 ± 0.571
CGN (BN in stem and classifier)	$\mathbf{75.807 \pm 0.511}$

Table 3: Classification accuracy on FigureQA *validation2* averaged over three runs.

Model	Accuracy (%)
FiLM [9]	$\mathbf{91.618 \pm 0.132}$
CGN (all GN)	91.343 ± 0.436
CGN (BN in stem, classifier no norm)	91.080 ± 0.166
CGN (BN in stem and classifier)	91.317 ± 0.514

Table 4: Test accuracies on several versions of SQOOP averaged over three runs.

Dataset	Model	Accuracy (%)
1 rhs/lhs	FiLM (BN)	72.369 ± 0.529
	CGN (all GN)	74.020 ± 2.814
	CGN (BN in stem, classifier no norm)	73.824 ± 0.334
	CGN (BN in stem and classifier)	$\mathbf{74.929 \pm 3.888}$
2 rhs/lhs	FiLM (BN)	84.966 ± 4.165
	CGN (all GN)	$\mathbf{86.689 \pm 6.308}$
	CGN (BN in stem, classifier no norm)	83.109 ± 0.381
	CGN (BN in stem and classifier)	85.859 ± 5.318
4 rhs/lhs	FiLM (BN)	97.043 ± 1.958
	CGN (all GN)	91.404 ± 0.318
	CGN (BN in stem, classifier no norm)	91.601 ± 1.937
	CGN (BN in stem and classifier)	$\mathbf{99.474 \pm 0.254}$
35 rhs/lhs	FiLM (BN)	$\mathbf{99.841 \pm 0.043}$
	CGN (all GN)	99.755 ± 0.025
	CGN (BN in stem, classifier no norm)	99.815 ± 0.122
	CGN (BN in stem and classifier)	99.782 ± 0.155

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.

Table 5: Five-way five-shot classification accuracy on Fewshot-CIFAR100 [18] and Mini-Imagenet [36] averaged over ten runs.

Dataset	Model	Accuracy (%)
FC100	TADAM (BN) [18]	$\mathbf{52.996 \pm 0.610}$
	TADAM (GN)	52.807 ± 0.509
Mini-Imagenet	TADAM (BN) [18]	$\mathbf{76.414 \pm 0.499}$
	TADAM (GN)	74.032 ± 0.373

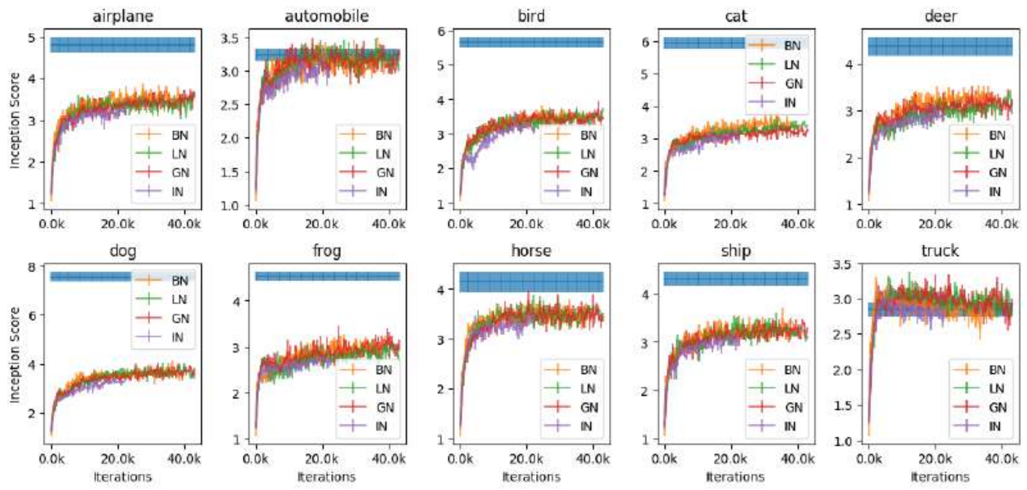


Figure 1: Inception score (IS) of samples generated by WGAN-GP model trained on CIFAR-10, sorted by class. (Blue is the IS of true data)

However, BN's outputs are highly dependent on the batch size, since there would be significant difference in the variance in the data. Therefore, it might be advantageous to use GN instead of BN, since GN is more computationally efficient, and not affected by batch size.

Moreover, we see in Figure 3 that the discriminator loss, which is a very informative metric to check if a GAN is converging, eventually saturates to the same values for all evaluated normalization techniques. This further suggests that BN may not provide any clear optimization advantage over GN.

Figure 4 shows samples from the same model architecture trained using each normalization type. Qualitatively as well, we cannot see a lot of differences in the images generated by any of the 4 types of normalization used.

With SAGAN or BigGAN, although we were able to replicate the architectures in code, we were not able to obtain the excellent quality of images as in the original papers. However, the training graphs and samples did not provide conclusive evidence of any one of the normalization techniques being critically better than the others.

5 Conclusion

We considered a set of experiments for VQA, few-shot learning and image generation tasks in which some of the best models rely on BN for conditional computation. As the performance of BN, unsatisfactorily, heavily depends on the batch size and on how well training and test statistics match, we investigated replacing BN by GN. We experimentally showed that CGN can be a good replacement for CBN for conditional computation, without negatively impacting performance in many cases. We hope that this study can serve as a reference for research in simple general purpose methods for conditional computation.

Note that in most experiments, we simply replaced BN with GN and did not perform extensive hyper-parameter optimization. We also did not experiment with different batch sizes, so as to have a

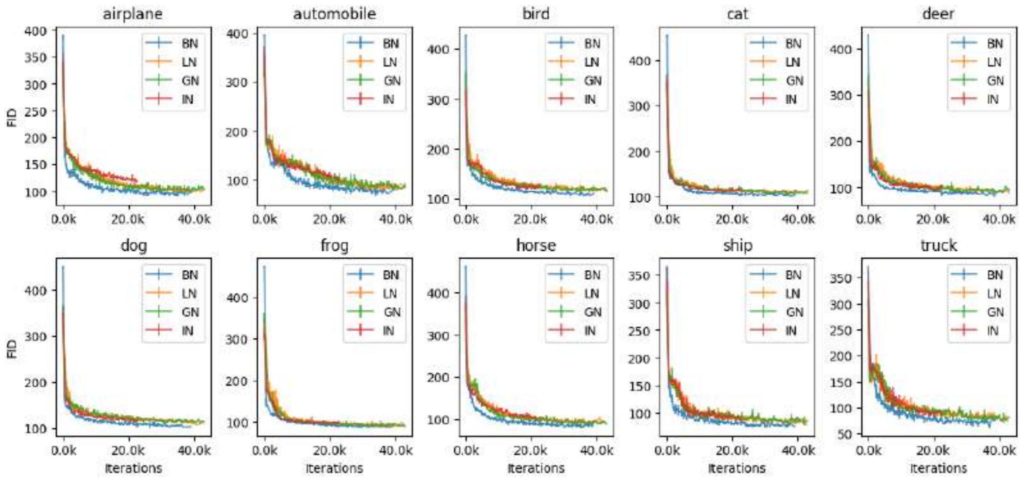


Figure 2: FID of samples generated by WGAN-GP model trained on CIFAR-10, sorted by class

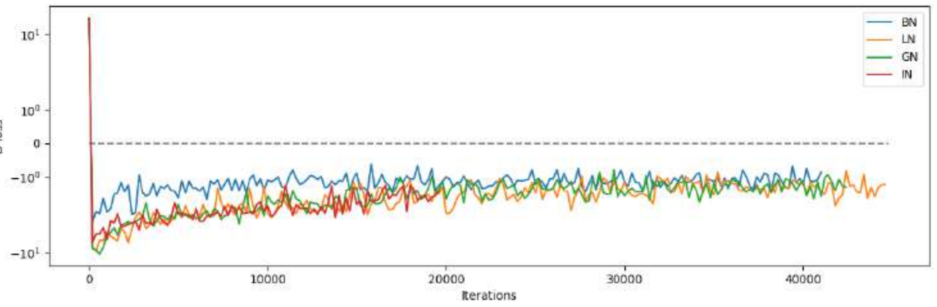


Figure 3: Discriminator loss for different normalization techniques

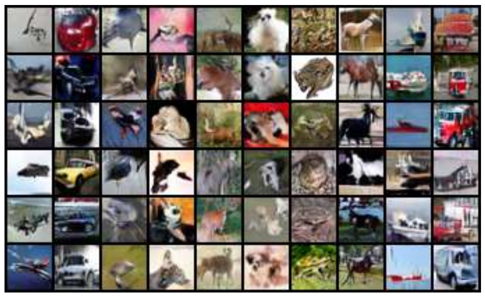
fair comparison. More extensive hyper-parameter tuning is likely to further improve the performance of the GN variant. In settings with large input spaces and conditioning information, such as video question answering [37, 38], where large batch sizes may be impractical computationally, CGN could find an application highlighting its strengths compared to CBN. Studying such tasks remains as future work. As the authors of Wu and He [6] point out, GN lacks the regularization effect that BN has thanks to noisy batch statistics. They propose to explore combinations of GN with an appropriate regularization scheme to improve performance. Similarly, recent successful attempts for replacing (unconditional) BN with careful network initialization in Zhang et al. [39] need to rely on additional regularization [40] to match up generalization performance. We see an investigation of regularization schemes for both GN and CGN as an important and promising future research direction.

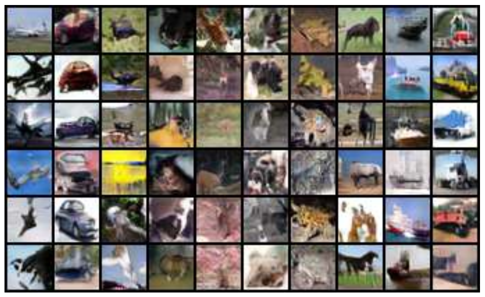
References

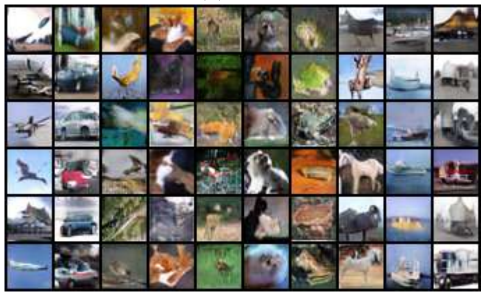
[1] Dzmitry Bahdanau, Shikhar Murty, Michael Noukhovitch, Thien Huu Nguyen, Harm de Vries, and Aaron Courville. Systematic generalization: What is required and can it be learned? *arXiv preprint arXiv:1811.12889*, 2018.

[2] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 39–48, 2016.

[3] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.


(a) CBN


(b) CGN


(c) CLN

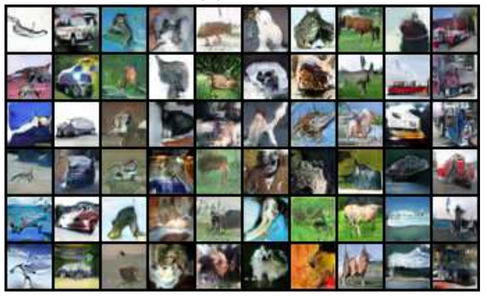

(d) CIN

Figure 4: Samples from models trained with different normalization techniques. The images in each column belong to the same class, ordered as 'airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck'. (Samples are not cherry-picked)

[4] Samira Ebrahimi Kahou, Vincent Michalski, Adam Atkinson, Ákos Kádár, Adam Trischler, and Yoshua Bengio. Figureqa: An annotated figure dataset for visual reasoning. *Workshop in the International Conference on Learning Representations*, 2017.

[5] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, 2015.

[6] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–19, 2018.

[7] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In *Advances in Neural Information Processing Systems*, 2018.

[8] Harm De Vries, Florian Strub, Jérémie Mary, Hugo Larochelle, Olivier Pietquin, and Aaron C Courville. Modulating early visual processing by language. In *Advances in Neural Information Processing Systems*, pages 6594–6604, 2017.

[9] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[10] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. *International Conference on Learning Representations (ICLR)*, 2017.

[11] Karol Kurach, Mario Lucic, Xiaohua Zhai, Marcin Michalski, and Sylvain Gelly. The gan landscape: Losses, architectures, regularization, and normalization. *CoRR*, abs/1807.04720, 2018.

[12] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[13] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *CoRR*, abs/1607.08022, 2016.

[14] Mateusz Malinowski and Mario Fritz. A multi-world approach to question answering about real-world scenes based on uncertain input. In *Advances in neural information processing systems*, pages 1682–1690, 2014.

[15] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2425–2433, 2015.

[16] Vincent Dumoulin, Ethan Perez, Nathan Schucher, Florian Strub, Harm de Vries, Aaron Courville, and Yoshua Bengio. Feature-wise transformations. *Distill*, 3(7):e11, 2018.

[17] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2016.

[18] Boris Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. In *Advances in Neural Information Processing Systems*, pages 721–731, 2018.

[19] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, 2014.

[20] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014.

[21] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *International Conference on Machine Learning (ICML)*, 2017.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 770–778, 2016.

[23] Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, page 5769–5779, 2017.

[24] Han Zhang, Ian J. Goodfellow, Dimitris N. Metaxas, and Augustus Odena. Self-attention generative adversarial networks. *International Conference on Learning Representations (ICLR)*, 2018.

[25] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *International Conference on Learning Representations*, 2019.

[26] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *International Conference on Learning Representations (ICLR)*, 2018.

[27] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[28] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Judy Hoffman, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Inferring and executing programs for visual reasoning. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2989–2998, 2017.

[29] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[30] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, 2016.

[31] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, 2017.

[32] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NeurIPS Autodiff Workshop*, 2017.

[33] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`. Software available from tensorflow.org.

[34] Shane Barratt and Rishi Kant Sharma. A note on the inception score. *CoRR*, abs/1801.01973, 2018.

[35] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[36] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638, 2016.

[37] Jonghwan Mun, Paul Hongsuck Seo, Ilchae Jung, and Bohyung Han. Marioqa: Answering questions by watching gameplay videos. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2867–2875, 2017.

[38] Makarand Tapaswi, Yukun Zhu, Rainer Stiefelhagen, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. Movieqa: Understanding stories in movies through question-answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 4631–4640, 2016.

[39] Hongyi Zhang, Yann N. Dauphin, and Tengyu Ma. Fixup initialization: Residual learning without normalization. In *International Conference on Learning Representations*, 2019.

[40] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=r1Ddp1-Rb`.