Conditional Group Normalization

ANONYMOUS AUTHOR(S)

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 competitive performance to batch normalization. All these methods utilize a learned affine transformation after the normalization operation to increase representational power. Methods used in conditional computation define the parameters of these transformations as learnable functions of conditioning information. In this work, we study whether and where the conditional formulation of group normalization can improve generalization compared to conditional batch normalization. We evaluate performances on the tasks of visual question answering and conditional image generation. Our experiments indicate that conditional group normalization is a reasonable replacement for conditional batch normalization and can achieve improved performance in certain tasks.

CCS Concepts: • **Computing methodologies → Neural networks**.

Additional Key Words and Phrases: conditional computation, normalization, generalization

ACM Reference Format:

Anonymous Author(s). 2018. Conditional Group Normalization. In *Proceedings of KDD '20*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/1122445.1122456

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[1], 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. In other cases, the i.i.d. assumption is dropped on purpose, to study whether a model can capture regularity that generalize *out-of-distribution*. Several benchmarks for evaluating task-specific models for their generalization capacity [6, 21, 22] have been proposed recently. The problem of out-of-distribution generalization can in some cases be addressed by making strong assumptions that simplify discovering a family of patterns from limited data. Bahdanau et al. [6], for example, show that their proposed synthetic relational reasoning task can be solved by a Neural Module Network (NMN) [2] with fixed tree structure, while models without this structural prior fail. At the other end of the spectrum are more "generic" models as Feature-wise Linear Modulation (FiLM) [34], which are able to uncover some of the compositionality in the tasks they are trained for.

[1] All data samples are assumed to be drawn independently from an identical distribution (i.i.d.).

In this paper, we focus on such generic models, that leverage conditional normalization methods for applications in visual question answering (VQA) and generative modeling of images. Despite our focus on these tasks, any improvement in this area can also benefit other domains such as deep reinforcement learning or metalearning, where conditional normalization layers are also being used [5, 20, 31]. We study strong deep neural network models for these tasks that employ Conditional Batch Normalization (CBN) [12] for modulating normalized activations with contextual information.

Since Batch Normalization (BN) normalizes activations with statistics computed across multiple training samples, one has to precompute activation statistics over the training set to be used during inference. Due to this reliance on dataset statistics, it seems that BN [19] (and thus also CBN) may be vulnerable to significant domain shifts between training and test data. To train models with BN one has to use a sufficiently large mini-batch size to limit the noise of activation statistics. Further potential issues with BN include limited diversity of samples generated by Generative Adversarial Networks (GANs) involving BN [41] and vulnerability to adversarial examples [14].

The recently proposed Group Normalization (GN) [40] normalizes across groups of feature maps instead of across samples in mini-batches. Here, we explore whether a conditional formulation of GN is a viable alternative for CBN. GN is conceptually simpler than BN, as its function is the same during training and inference. Further, GN can be used with small batch sizes, which may help in applications with particularly large feature maps, such as medical imaging or video processing, in which the available memory can be a constraint.

Our contribution is an extensive empirical study of two conditional normalization techniques over multiple tasks and benchmarks. To the best of our knowledge, the present work is the first to introduce Conditional Group Normalization (CGN). Our experiments show that it has advantages for out-of-distribution generalization.

2 BACKGROUND

2.1 Normalization Layers

Several normalization methods have been proposed to stabilize and speed-up the training of deep neural networks [4, 19, 38, 40]. To stabilize the range of variation of network activations x_i, methods such as BN [19] first normalize the activations by subtracting mean μ_i and dividing by standard deviation σ_i:

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

The distinction between different methods lies in how exactly these statistics are being computed. Wu and He [40] aptly summarize several methods using the following notation. Let $i = (i_N, i_C, i_H, i_W)$ be a four-dimensional vector, whose elements index the features along the batch, channel, height and width axes, respectively. The computation of the statistics can then be written as

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

where the set $\mathcal{S}_i$ of size m is defined differently for each method and ϵ is a small constant for numerical stability. BN, for instance, corresponds to:

$$\mathcal{S}_i = \{k | k_C = i_C\}, \tag{3}$$

i.e. $\mathcal{S}_i$ is the set of all pixels sharing the same channel axis, resulting in μ_i and σ_i being computed along the (N, H, W) axes.

As Ba et al. [4] point out, the performance of BN is highly affected by the batch size hyperparameter. This insight led to the introduction of several alternative normalization schemes, that normalize per sample, i.e. not along batch axis N. For instance, Layer Normalization (LN) [4], which normalizes activations within each layer, corresponds to the following set definition:

$$\mathcal{S}_i = \{k | k_N = i_N\}. \tag{4}$$

Ulyanov et al. [38] introduce Instance Normalization (IN) in the context of image stylization. IN normalizes separately for each sample and each channel along the spatial dimensions:

$$\mathcal{S}_i = \{k | k_N = i_N, k_C = i_C\}. \tag{5}$$

Recently, Wu and He [40] introduced GN, which draws inspiration from classical features such as Histogram of Oriented Gradients (HOG) [11]. It normalizes features per sample, separately within each of G groups, along the channel axis:

$$\mathcal{S}_i = \{k | k_N = i_N, \lfloor \frac{k_C}{C/G} \rfloor = \lfloor \frac{i_C}{C/G} \rfloor\} \tag{6}$$

GN can be seen as a way to interpolate between the two extremes of LN (corresponding to $G = 1$, i.e. all channels are in a single group) and IN (corresponding to $G = C$, i.e. each channel is in its own group).

After normalization, all above mentioned methods insert a scaling and shifting operation using learnable per-channel parameters γ and β:

$$y_i = \gamma \hat{x}_i + \beta \tag{7}$$

This "de-normalization" is done to restore the representational power of the normalized network layer [19].

CBN [12, 34] is a conditional variant of BN, in which the learnable parameters γ and β in Equation 7 are replaced by learnable functions

$$\gamma(c_k) = W_\gamma c_k + b_\gamma, \quad \beta(c_k) = W_\beta c_k + b_\beta \tag{8}$$

of some per-sample conditioning input c_k to the network with parameters $W_\gamma, W_\beta, b_\gamma, b_\beta$. In a VQA model, c_k would for instance be an embedding of the question [34]. Before CBN was introduced, Dumoulin et al. [13] proposed Conditional Instance Normalization (CIN), a conditional variant of IN very similar to CBN, using IN instead of BN. In our experiments, we explore a conditional variant of GN, i.e. CGN.

2.2 Visual Question Answering

In VQA [3, 25], the task is to answer a question about an image. This task is usually approached by feeding both image and question to a parametric model, which is trained to predict the correct answer, for instance via classification among all possible answers in the dataset. One recent successful model for VQA is the FiLM architecture [34], which employs CBN to modulate visual features based on an embedding of the question.

2.3 Conditional Image Generation

Some of the most successful models for generating images are GANs [15]. This approach involves training one neural network (Generator) to generate an image, while the only supervisory signal is that from another neural network (Discriminator) which indicates whether the image looks real or not. Several variants of GANs [26, 30] have been proposed to condition the image generation process on a class label. More recently, the generators that work best stack multiple ResNet-style [17] architectural blocks, involving two CBN-ReLU-Conv operations and an upsampling operation. These blocks are followed by a BN-ReLU-Conv operation to transform the last features into the shape of an image. Such models can be trained as Wasserstein GANs using gradient penalty (WGAN-GP) as proposed by Gulrajani et al. [16], which gives mathematically sound arguments for an optimization framework.

More recently, Spectral Norm GAN (SNGAN) [27] uses the aforementioned architecture with spectral normalization on the weights to stabilize training at each iteration. Two noteworthy GAN models that use architectures based on SNGAN are Self-Attention GAN (SAGAN) [44] and BigGAN [8]. SAGAN inserts a self-attention mechanism [9, 32, 39] to attend over important parts of features during the generation process. The architecture of BigGAN is the same as for SAGAN, with the exception of an increase in batch size and channel widths, as well as some architectural changes to improve memory and computational efficiency. Both of these models have been used successfully in generating high quality natural images. In our experiments, we use SNGAN and compare performance metrics of two types of normalization — CBN and CGN.

3 EXPERIMENTS

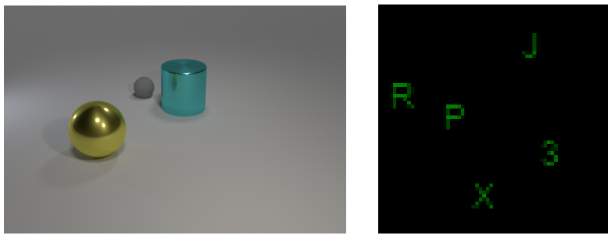
3.1 Visual Question Answering

We study whether CGN in the VQA architecture FiLM [34] yields performance improvements over CBN. We run experiments on the following recently proposed benchmarks for compositional generalization:

CLEVR-CoGenT. CLEVR Compositional Generalization Test (CLEVR-CoGenT) [21] is a variant of the popular Compositional Language and Elementary Visual Reasoning (CLEVR) dataset [21] that tests for compositional generalization. See Figure 1 (a) for an example from this dataset. The images consist of rendered three-dimensional scenes containing several shapes (small and large cubes, spheres and cylinders) of differing material properties (*metal* or *rubber*) and colors. Questions involve *queries* for object attributes, *comparisons*, *counting* of sets and combinations thereof. In contrast to the regular CLEVR dataset, the training set of CLEVR-CoGenT explicitly combines some shapes only with different subsets of four out of eight colors, and provides two validation sets: one with the same combinations (*valA*) and one in which the shape-color assignments are swapped (*valB*). To perform well on *valB*, the model has to generalize to unseen combinations of shapes and colors, i.e. it needs to capture the compositionality of the task.

SQOOP. Spatial Queries On Object Pairs (SQOOP) [6] is a recently introduced dataset that tests for systematic generalization. Figure 1 (b) shows an example from the training set. It consists of images containing five randomly chosen and arranged objects (digits and characters). Questions concern the four spatial relations *LEFT OF*, *RIGHT OF*, *ABOVE* and *BELOW* and the queries are all of the format "X R Y?", where X and Y are left-hand and right-hand objects and R is a relationship between them, e.g. "nine LEFT OF a?". To test for systematic generalization, only a limited number of combinations of each left-hand object with different right-hand objects Y are shown during training. In the hardest version of the task (1 rhs/lhs), only a single right-hand side object is combined with each left-hand side object. For instance, the training set of this version may contain images with the query "a RIGHT OF b?", but no images with queries about relations of left-hand object *a* with any other object than *b*. The test set contains images and questions about all combinations, i.e. it evaluates generalization to relations between novel object combinations.

3.1.1 Model. We experiment with three small variations of the FiLM architecture [34]. The original architecture in Perez et al. [34] consists of an unconditional *stem* network, followed by a core of four ResNet [17] blocks with CBN [12], and finally a classifier. The stem network is either a sequence of residual blocks trained from scratch or a fixed pre-trained feature extractor followed by a learnable layer of 3×3 convolutions. The scaling and shifting parameters of the core layers are affine transforms of a question embedding provided by a gated recurrent unit (GRU) [10]. The output of the last residual block is fed to the classifier, which consists of a layer of 512 1×1 convolutions, global



(a) **CLEVR-CoGenT** (b) **SQOOP**

Fig. 1. Examples of the VQA datasets used in our experiments. **Corresponding question-answer pairs**: (a) *Are there any gray things made of the same material as the big cyan cylinder? - No.* (b) *X right_of J? - No*

max-pooling, followed by a fully-connected ReLU [29] layer using (unconditional) BN and a softmax layer, which outputs the probability of each possible answer. We train the following three variants that include CGN[2]:

(1) "all GN": all conditional and regular BN layers are replaced with corresponding conditional or regular GN layers.
(2) "BN stem": all CBN layers are replaced with CGN, regular BN layers in the stem and classifier are left unchanged, except those in the fully-connected hidden layer in the classifier, for which we remove normalization.
(3) "BN stem & classifier": all CBN layers in the core ResNet blocks are replaced with CGN, regular BN in the stem and classifier are left unchanged.

Besides the described changes in the normalization layers, the architecture and hyperparameters are the same as used in Perez et al. [34] for all experiments, except for SQOOP where they are the same as in Bahdanau et al. [6]. The only difference is that we set the constant ϵ of the Adam optimizer [23] to 1e−5 to improve training stability[3]. For SQOOP, the input to the residual network are the raw image pixels. For CLEVR-CoGenT, we instead feed features extracted from layer *conv4* of a ResNet-101 [17], pre-trained on ImageNet [35], following Perez et al. [34].

3.1.2 Results. Tables 1 and 2 show the results of training FiLM with CBN and CGN on the two considered datasets. In the experiments on CLEVR-CoGenT, all three CGN variants of FiLM achieve a slightly higher average accuracy. Note that for the SQOOP dataset we rerun the original CBN experiments by Bahdanau et al. [6] and observe significantly higher accuracy on all versions of the task. In the hardest SQOOP variant with only one right-hand side object per left-hand side object (*1 rhs/lhs*), all three variants of CGN achieve a higher performance than the CBN experiments (both original and ours). For the SQOOP variant with four right-hand side objects per left-hand side object, CGN did not converge in some cases. It is possible that additional regularization is required to guarantee convergence. Note, that learning curves of models successfully trained on SQOOP all seem to follow the same pattern: For a relatively large number of gradient updates there is no significant improvement. Then, at some point, almost instantly the model achieves 100% training accuracy.

3.2 Conditional Image Generation

Here we compare CBN and CGN on the task of generating images conditioned on their class label using the SNGAN [27] architecture.

[2]We always set the number of groups to 4, as the authors of Wu and He [40] 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. [34] confirmed occasional gradient explosions with the original setting of $1e-8$.

Table 1. Classification accuracy on CLEVR-CoGenT *valB*. Mean and standard deviation of three runs with early stopping on *valA* are reported for the models we trained.

Model	Accuracy (%)
CBN (FiLM [34])	75.600
CBN (FiLM, our results)	75.539 ± 0.671
CGN (all GN)	75.758 ± 0.356
CGN (BN stem)	75.703 ± 0.571
CGN (BN stem & classifier)	**75.807 ± 0.511**

Table 2. Test accuracies on several versions of SQOOP. Mean and standard deviation of three runs after early stopping on the validation set are reported for the models we trained. Here, FiLM refers to the model specified in [6], whereas "FiLM, ours" indicates our run of the same.

Model	Accuracies (%)			
	1 rhs/lhs	2 rhs/lhs	4 rhs/lhs	35 rhs/lhs
CBN (FiLM)	65.270 ± 4.610	80.200 ± 4.320	90.420 ± 1.000	99.803 ± 0.219
CBN (FiLM, ours)	72.369 ± 0.529	84.966 ± 4.165	97.043 ± 1.958	**99.841 ± 0.043**
CGN (all GN)	74.020 ± 2.814	**86.689 ± 6.308**	91.404 ± 0.318	99.755 ± 0.025
CGN (BN stem)	73.824 ± 0.334	83.109 ± 0.381	91.601 ± 1.937	99.815 ± 0.122
CGN (BN stem & classifier)	**74.929 ± 3.888**	85.859 ± 5.318	**99.474 ± 0.254**	99.782 ± 0.155

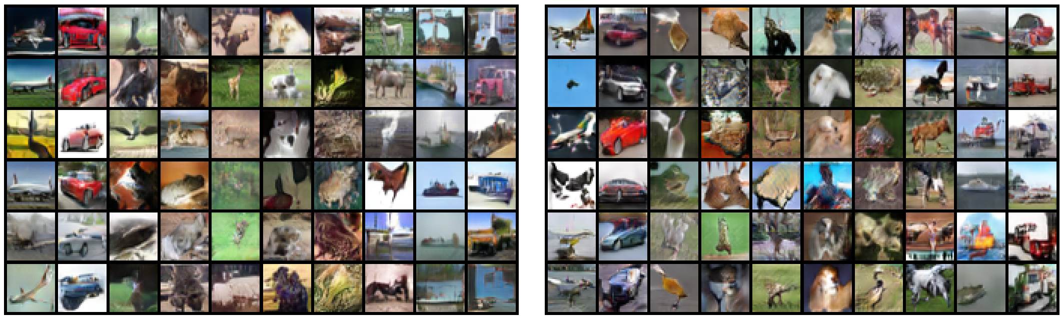
CIFAR-10. CIFAR-10 [24] is a data set containing 60000 32×32 images, 6000 for each of 10 classes. The dataset is split into 50000 training and 10000 test samples.

3.2.1 Model. We use the official implementation of SNGAN [28]. We replace the BN modules in the residual blocks with CBN and CGN for the respective cases, with number of groups set to 4 in case of CGN. We retain the optimization setup of a learning rate of $2e^{-4}$ for both generator and discriminator, five discriminator updates per generator update using Adam optimizer [23]. We train using a single GPU (NVIDIA P100) and a batch size of 64. We also perform experiments where we use smaller batch sizes, to show the effects of CBN and CGN in helping generalization.

3.2.2 Results. Figure 2 shows samples from conditional SNGAN trained using each of the two normalization methods.

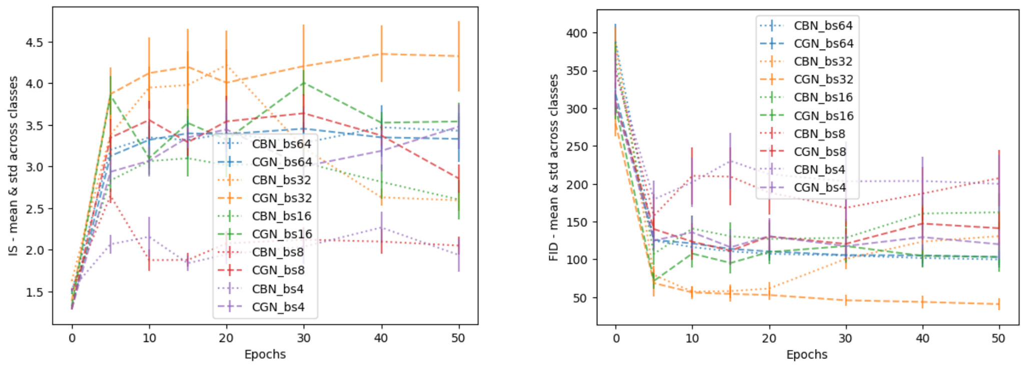
For both normalization methods, in addition to a qualitative check of the generated samples, we calculate two scores that are widely used in the community to evaluate image generation Inception Score (IS) [7, 36] and Fréchet Inception Distance (FID) [18]. Since we are using publicly available PyTorch [33] implementations to compute these scores, the values for real data may differ slightly from scores computed using the original TensorFlow [1] implementation. However, we compare scores using the same implementation of these metrics 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 [37]. Although the suitability of the IS for this purpose has been rightfully put into question [7], 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 InceptionV3 network [37]. In this case, we measure the distance between



(a) CBN (b) CGN

Fig. 2. 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.



(a) IS (b) FID

Fig. 3. (a) Inception Score (IS) [36] (higher is better), (b) Fréchet Inception Distance (FID) [18] (lower is better), each averaged across all classes for one run each of CBN- and CGN-based models, for different batch sizes — 64, 32, 16, 8, 4

the real and generated CIFAR-10 images. This is a better metric than IS, since there is no constraint on the images being natural, and it is able to quantify not only their similarity to the real images, but also diversity in the generated images.

We trained SNGAN models to generate CIFAR-10 images conditioned on the class, for different batch sizes, viz. 64, 32, 16, 8, 4. In each case, we trained one model with CBN and another with CGN to compare them. We calculated IS and FID on these models at different stages of training, as can be seen in Figure 3 (a) and (b).

We see that the models trained with CGN achieve much better values for IS and FID than those with CBN. In the case of batch size 64, CGN performs very closely as CBN, and in all other cases of smaller batch sizes, CGN clearly outperforms CBN. This indicates the heavy dependence of CBN on batch size, which prevents it from generalizing well. Thus, we believe CGN is preferable as a module to use in a deep neural network-based generative models than CBN.

The SNGAN model architecture consists of a series of residual blocks followed by bn-relu-conv layers. Each residual block contains two bn-relu-conv modules, with an optional upsampling layer. Since the architectures of more recent models such as SAGAN [44] and BigGAN [8] are very similar to that of the one we used, it is likely that the conclusions we draw from the SNGAN experiments transfer to them.

4 CONCLUSION

Because the performance of CBN heavily depends on the batch size and on how well training and test statistics match, we investigate the use of CGN as a potential alternative for CBN.

We experimentally show that the effect of this substitution is task-dependent, with performance increases in some VQA tasks that focus on systematic generalization. In conditional image generation, we show that CGN can be trained with significantly smaller batch sizes than CBN, sometimes even with increased performance as measured by the IS and FID metrics. CGN's simpler implementation, its consistent behaviour during training and inference time, as well as its independence from batch sizes, are all good reasons to explore its adoption instead of CBN in tasks that require out-of-distribution generalization. That being said, further analysis is required to be able to confidently suggest one method over the other. For instance, a hyperparameter search for each of the normalization methods would be required to provide a more detailed performance comparison. As shown in our conditional image generation experiments, CGN is more amenable to training with small batch sizes. This suggests investigating applications in domains where efficient large-batch training is non-trivial, such as medical imaging or video processing.

Lastly, since some of the success of BN (and consequently CBN) can be attributed to the regularization effect introduced by noisy batch statistics, it seems worthwhile to explore combinations of CGN with regularization as suggested for GN by Wu and He [40]. This is also motivated by recent successful attempts at replacing (unconditional) BN with careful network initialization [43], which relies on regularization [42] to match generalization performance.

REFERENCES

[1] Martín Abadi et al. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. http://tensorflow.org/

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

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

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

[5] Dzmitry Bahdanau, Felix Hill, Jan Leike, Edward Hughes, Pushmeet Kohli, and Edward Grefenstette. 2018. Learning to follow language instructions with adversarial reward induction. *arXiv preprint arXiv:1806.01946* (2018).

[6] Dzmitry Bahdanau, Shikhar Murty, Michael Noukhovitch, Thien Huu Nguyen, Harm de Vries, and Aaron Courville. 2018. Systematic Generalization: What Is Required and Can It Be Learned? *arXiv preprint arXiv:1811.12889* (2018).

[7] Shane Barratt and Rishi Kant Sharma. 2018. A Note on the Inception Score. *CoRR* abs/1801.01973 (2018).

[8] Andrew Brock, Jeff Donahue, and Karen Simonyan. 2019. Large Scale GAN Training for High Fidelity Natural Image Synthesis. *International Conference on Learning Representations* (2019).

[9] Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733* (2016).

[10] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* (2014).

[11] Navneet Dalal and Bill Triggs. 2005. Histograms of oriented gradients for human detection. In *international Conference on computer vision & Pattern Recognition (CVPR'05)*, Vol. 1. IEEE Computer Society.

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

[13] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. 2017. A Learned Representation For Artistic Style. *International Conference on Learning Representations (ICLR)* (2017).

[14] Angus Galloway, Anna Golubeva, Thomas Tanay, Medhat Moussa, and Graham W Taylor. 2019. Batch Normalization is a Cause of Adversarial Vulnerability. *arXiv preprint arXiv:1905.02161* (2019).

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

[16] Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. 2017. Improved Training of Wasserstein GANs. In *Advances in Neural Information Processing Systems*.

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

[18] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 2017. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *Advances in Neural Information Processing Systems*.

[19] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *International Conference on Machine Learning (ICML)*.

[20] Xiang Jiang, Mohammad Havaei, Farshid Varno, Gabriel Chartrand, Nicolas Chapados, and Stan Matwin. 2018. Learning to learn with conditional class dependencies. (2018).

[21] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2017. CLEVR: A Diagnostic Dataset for Compositional Language and Elementary Visual Reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

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

[23] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)* (2014).

[24] Alex Krizhevsky. 2009. Learning Multiple Layers of Features from Tiny Images. (2009).

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

[26] Mehdi Mirza and Simon Osindero. 2014. Conditional Generative Adversarial Nets. *CoRR* abs/1411.1784 (2014).

[27] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. 2018. Spectral Normalization for Generative Adversarial Networks. *ArXiv* abs/1802.05957 (2018).

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

[29] Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*.

[30] Augustus Odena, Christopher Olah, and Jonathon Shlens. 2017. Conditional Image Synthesis With Auxiliary Classifier GANs. In *International Conference on Machine Learning (ICML)*.

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

[32] Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933* (2016).

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

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

[35] Olga Russakovsky et al. 2015. Imagenet large scale visual recognition challenge. *International journal of computer vision* 115, 3 (2015).

[36] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. 2016. Improved Techniques for Training GANs. In *Advances in Neural Information Processing Systems*.

[37] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

[38] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. 2016. Instance Normalization: The Missing Ingredient for Fast Stylization. *CoRR* abs/1607.08022 (2016).

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*.

[40] Yuxin Wu and Kaiming He. 2018. Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

[41] Sitao Xiang and Hao Li. 2017. On the effects of batch and weight normalization in generative adversarial networks. *arXiv preprint arXiv:1704.03971* (2017).

[42] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. 2018. mixup: Beyond Empirical Risk Minimization. In *International Conference on Learning Representations*. https://openreview.net/forum?id=r1Ddp1-Rb

[43] Hongyi Zhang, Yann N. Dauphin, and Tengyu Ma. 2019. Fixup Initialization: Residual Learning Without Normalization. In *International Conference on Learning Representations*.

[44] Han Zhang, Ian J. Goodfellow, Dimitris N. Metaxas, and Augustus Odena. 2018. Self-Attention Generative Adversarial Networks. *International Conference on Learning Representations (ICLR)* (2018).