Attentive Normalization

Xilai Li, Wei Sun and Tianfu Wu [*]
Department of ECE and the Visual Narrative Initiative, NC State University
{xli47, wsun12, tianfu_wu}@ncsu.edu

Abstract

Batch Normalization (BN) is a vital pillar in the development of deep learning with many recent variations such as Group Normalization (GN) and Switchable Normalization. Channel-wise feature attention methods such as the squeeze-and-excitation (SE) unit have also shown impressive performance improvement. BN and its variants take into account different ways of computing the mean and variance within a min-batch for feature normalization, followed by a learnable channel-wise affine transformation. SE explicitly learns how to adaptively recalibrate channel-wise feature responses. They have been studied separately, however. In this paper, we propose a novel and lightweight integration of feature normalization and feature channel-wise attention. We present Attentive Normalization (AN) as a simple and unified alternative. AN absorbs SE into the affine transformation of BN. AN learns a small number of scale and offset parameters per channel (i.e., different affine transformations). Their weighted sums (i.e., mixture) are used in the final affine transformation. The weights are instance-specific and learned in a way that channel-wise attention is considered, similar in spirit to the squeeze module in the SE unit. AN is complementary and applicable to existing variants of BN. In experiments, we test AN in the ImageNet-1K classification dataset and the MS-COCO object detection and instance segmentation dataset with significantly better performance obtained than the vanilla BN. Our AN also outperforms two state-of-the-art variants of BN, GN and SN. The source code will be released at http://github.com/ivMCL/AttentiveNorm.

1. Introduction

1.1. Motivation and Objective

Batch Normalization (BN) [9] is a milestone feature normalization technique in the development of deep learning, which aims to improve both the training and generalization capacities of deep neural networks (DNNs) by stablizing the distributions of layer inputs. Many variants of BN have been proposed for practical deployment in terms of variations of training and testing settings with impressive results obtained, such as Layer Normalization (LN) [1], Instance Normalization (IN) [18], Group Normalization (GN) [23] and Switchable Normalization (SN) [13]. They share a unified formulation and differ in taking into account different ways of computing the mean and variance within a min-batch for feature normalization. More recently, BN has been deeply analyzed by addressing how it helps optimization [17] and whether it can be replaced, etc. Especially, the latest Fixup initialization method [24] shows that BN can be removed without hurting the performance of DNNs, at least for ResNets [5]. Also, conditional BN [4, 3] has been developed for the generator in generative adversarial networks (GANs) to learn class-specific [14, 2] or style-specific [10] affine transformations. In this paper, we are interested in,

(i) Developing conditional and dynamic BN for discriminative visual recognition tasks (such as image classification in ImageNet [15] and object detection and segmentation in MS-COCO [11]) using convolutional neural networks (CNNs), which has yet been investigated and is complementary to existing variants of BN that focus on the computation of mean and variance to be more effective in different training and testing settings. However, what should be used as the conditions and how to learn them for dynamic and adaptive control of BN in discriminative tasks? (And if doable,)

(ii) Developing lightweight schema (in terms of both parameter increase and training/testing computation cost) that can be used as a drop-in replacement for BN layers in state-of-the-art CNNs such as ResNets [5] in discriminative tasks to improve their performance. Unlike GN [23] which addresses the small batch issue of BN, especially for training large models, we aim to explore how to improve the performance of BN, especially for applications in which BN naturally fits such as training small models for mobile settings. For SN [13], the main drawback is its extra training and inference time overhead, and BN is still included, so

[*]Preprint. On-going work. T. Wu is the corresponding author.

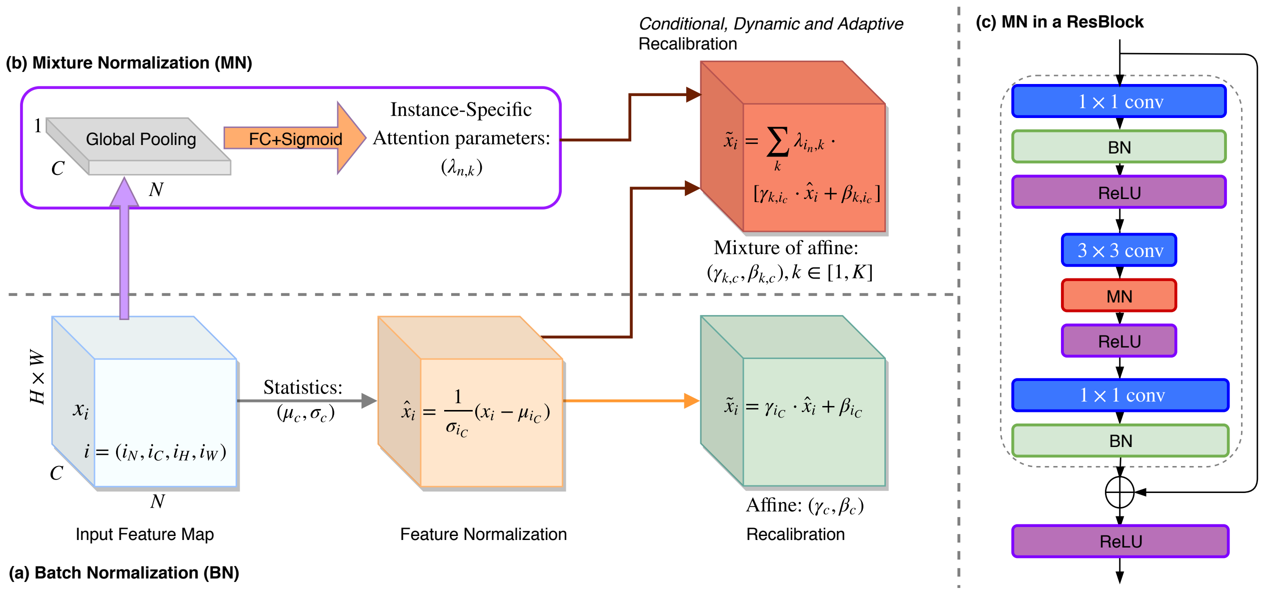
(b) Mixture Normalization (MN)
Global Pooling
FC+Sigmoid
Instance-Specific Attention parameters: $(\lambda_{n,k})$
Conditional, Dynamic and Adaptive Recalibration
$\tilde{x}_i = \sum_k \lambda_{i_n,k} \cdot [\gamma_{k,i_c} \cdot \hat{x}_i + \beta_{k,i_c}]$
Mixture of affine: $(\gamma_{k,c}, \beta_{k,c}), k \in [1, K]$
$H \times W$
x_i
$i = (i_N, i_C, i_H, i_W)$
Statistics: (μ_c, σ_c)
$\hat{x}_i = \frac{1}{\sigma_{i_C}}(x_i - \mu_{i_C})$
$\tilde{x}_i = \gamma_{i_C} \cdot \hat{x}_i + \beta_{i_C}$
Affine: (γ_c, β_c)
Input Feature Map
Feature Normalization
Recalibration
(a) Batch Normalization (BN)
(c) MN in a ResBlock
1×1 conv
BN
ReLU
3×3 conv
MN
ReLU
1×1 conv
BN
ReLU

Figure 1. Illustration of the proposed Attentive Normalization (AN) in (b) using the vanilla Batch Normalization (BN) [9] as backbone (a). AN shares the feature normalization component with BN, and differs in how the affine transformation is done. (c) shows our lightweight deployment of AN in the bottleneck of a ResNet building block (ResBlock) [5] which follows the 3×3 convolution unit to potentially jointly integrate spatial attention in learning the instance-specific attention parameters. AN can also use other variants of BN as backbones. The input feature map is represented using the convention (N, C, H, W) for the batch axis, channel axis, spatial height and width axes respectively. x_i represents a feature response in the input feature map with position index $i = (i_N, i_C, i_H, i_W)$. $\hat{x}_i$ represents the normalized response using the pooled channel-wise mean and variance. $\tilde{x}_i$ is the response after affine transformation with learned scale and offset parameters. See text for details.

improved BN will also benefit it. In addition, if succeeded, the lightweight schema will be helpful for "reclaiming" the importance of BN against the Fixup initialization [24].

Alongside feature normalization, feature attention is an important mechanism for improving task performance. Spatial attention is inherently captured by convolution operations within short-range context, and by non-local extensions [21, 8] for long-range context. Channel-wise attention is less exploited. The squeeze-and-excitation (SE) unit [6] is one of the most popular designs. However, although it is not very heavy and effective for improving performance, the SE units still introduce a significant number of extra parameters (e.g., $\sim$2.5M extra parameters for ResNet50 which originally consists of $\sim$25M parameters, resulting in 10% increase). We are interested in,

iii) Developing conditional and dynamic BN that can exploit feature channel-wise attention as done in the SE unit, but with much less extra cost, thus potentially enjoying comparable performance improvement in a more effective and efficient way. By analogy, we want to maximizing the gain while minimizing the pain. As we will show, the proposed method will only use $\sim$0.13M additional parameters for ResNet50 with the top-1 error rate reduced by almost 1%.

Intuitively, *the channel-wise affine transformation in feature normalization realizes the excitation module in the SE unit, that is to re-scale/recalibrate feature responses.* The difference is that the scale parameters (as well as the offset parameters) in the former are not adaptive and dynamic once learned, while the scale parameters of the latter are adaptively computed via the squeeze module. *So, we can utilize methods similar to the squeeze module to compute channel-wise attention in developing conditional, dynamic and adaptive feature normalization for discriminative tasks.* This simple and intuitive observation motivates us to integrate feature normalization and channel-wise attention for potentially taking advantage of the merits of both.

1.2. Method Overview

Figure 1 illustrates the proposed Attentive Normalization (AN) using BN as the backbone. Unlike BN which only learns one affine transformation for each channel, AN learns a small number K of affine transformation components per channel (e.g., $K = 5$ is a hyperparameter). The scale and offset parameters for the final instance-specific channel-wise affine transformation is computed as weighted sum of the mixture of affine transformation components. The instance-specific weights are learned from the input feature map. For example, we utilize the squeeze module in the SE unit to learn the weights. It consists of a global average pooling layer, a fully-connected layer and the sigmoid activation function. It first utilizes the mean of each filter to represent its "importance" and then learns the interdependencies between the filters from the eye of their means to capture channel-wise attention. We can also learn weights for the scale and offset parameters separately.

When deploying our AN, e.g., into the Bottleneck building block of ResNets [5], to control the extra parameters introduced by our AN, we use a lightweight deployment for it as shown in Figure 1 (c). It follows the 3×3 convolution unit since it has the least number of channels. Potentially, this will jointly integrate local spatial attention in learning the instance-specific attention parameters. We keep the other two feature normalization (BN) units. By mixing AN and BN, we also obtain a new type of Bottleneck operations.

The proposed method is complementary to existing variants of BN, and thus applicable to different feature normalization schema such as LN [1], IN [18], GN [23] and SN [13]. We leave this to future work due to the high demand of computing resources. We hope this work can stimulate more exploration of integrating feature normalization and attention in a more sophisticated yet lightweight way, either practically (as done in this paper) or theoretically.

In experiments, we test our AN in the ImageNet-1K classification dataset [15] and the MS-COCO object detection and instance segmentation dataset [11]. We significantly outperform the vanilla BN [9] with negligible extra parameters. We obtain better performance than GN [23] and SN [13] (our AN is comparable with SN on COCO, but with half training iterations). We also conduct some ablation studies showing the effectiveness of the proposed AN in an empirically interpretable way.

2. Related Work

2.1. Feature Normalization

There are two types of normalization schema, feature normalization (including raw data) [9, 1, 18, 23, 13] and weight normalization [16], which are of indispensable importance in deep learning. The former is first to use the pooled mean and variance (or other moments) to normalize input feature responses, followed by affine transformation with learnable scale and offset parameters to recalibrate the normalized responses. The latter is to normalize weights over the model parameter space, e.g., to decouple the magnitudes of parameter vectors from their directions.

We focus on feature normalization in this paper. Different feature normalization schema differ in how the mean and variance are computed. BN [9] computes the channel-wise mean and variance in the entire min-batch which is driven by improving training efficiency and model generalizability in CNN-based visual recognition tasks. IN [18] focuses on channel-wise and instance-specific statistics which stems from the task of artistic image style transfer. LN [1] computes the instance-specific mean and variance from all channels which is designed to help optimization in recurrent neural networks (RNNs). GN [23] stands in the sweet spot between LN and IN focusing on instance-specific and channel-group-wise statistics for CNN-based visual recognition tasks, especially when only small batches are applicable in practice. SN [13] leaves the design choices of feature normalization schema to the learning system itself by computing weighted sum integration of BN, LN, IN and/or GN via softmax, showing more flexible applicability.

However, the affine transformation component in different feature normalization schema has less variants, and has not been explored well in visual recognition tasks. Its by-design goal is to enable the learning system to be capable of "undoing" the feature normalization, so to maintain model expressivity. Recently, conditional BN [4, 3, 2, 10] has been developed and shown remarkable progress in generative conditional and unconditional image synthesis. Conditional BN learns condition-specific affine transformations in terms of conditions such as class labels and image style latent codes. The paper presents a method of exploring conditional and dynamic BN through the lens of affine transformation for CNN-based visual recognition tasks, and the proposed method is applicable to other feature normalization schema.

2.2. Feature Attention

Similar to feature normalization schema, feature attention mechanisms are also important building blocks in the development of deep learning. Residual Attention Network [20] uses a trunk-and-mask joint spatial and channel attention module in an encoder-decoder style for improve performance of CNNs. To reduce the computational cost, channel and spatial attention are separately applied in [22]. The SE unit [6] further simplifies the attention mechanism by developing a lightweight SE channel-wise attention method. Since we are interested in lightweight extensions of BN and are seeking for ways of controlling the scale and offset parameters in the channel-wise affine transformation, we utilize the best practice in the SE unit in this paper.

This paper takes an integrative approach to connect feature normalization and attention using a lightweight design. The resulting method also exploits the self-attention [19] idea: we first learn the interdependencies between feature channels (i.e., squeeze) through the eye of the pooled mean, and then use the learned weights to re-scale and re-shift the normalized feature responses.

Our Contributions. This paper makes the following main contributions in the field of deep learning.

- It presents a simple method that marries feature normalization (BN) and attention (channel-wise), termed Attentive Normalization (AN). To our best knowledge, AN is the first work to study the attention-based conditional and dynamic BN for CNN-based visual recognition tasks.

- It presents a lightweight deploying method for integrating AN in the widely used Bottleneck operation of ResNets. It leads to a new type of Bottleneck mixing AN and BN.

- It shows better results than GN and SN when using regular batch based training settings in ImageNet-1k and MS-COCO (comparable with SN with half training iterations used). It also shows interpretable justifications of the effectiveness of the proposed scheme of integrating feature normalization and attention.

3. Attentive Normalization (AN)

In this section, we first present a general form of feature normalization following the convention used in GN [23] and review the Squeeze-and-Excitation (SE) unit [6] to be self-contained, and then elaborate on the proposed AN with some underpinning discussed.

3.1. Formulation of Feature Normalization

Without loss of generality, consider visual recognition tasks for 2D images, denote by x a feature map with axes in the convention order of (N, C, H, W) (i.e., batch, channel, height and width). x is represented by 4D tensor. Let $i = (i_N, i_C, i_H, i_C)$ be the address index in the 4D tensor. x_i represents the feature response at the position i. Existing feature normalization schema consist of two components (Figure 1):

- *Computing normalized responses* by,
$$\hat{x}_i = \frac{1}{\sigma_i}(x_i - \mu_i) \tag{1}$$

- *Re-scaling and re-shifting* the normalized responses by affine transformation,
$$\tilde{x}_i = \gamma_i \cdot \hat{x}_i + \beta_i \tag{2}$$

The mean μ_i and standard deviation σ_i are computed by,

$$\begin{aligned} \mu_i =& \frac{1}{M}\sum_{j\in A_i} x_j, \\ \sigma_i =& \sqrt{\frac{1}{M}\sum_{j\in A_i}(x_j - \mu_i)^2 + \epsilon}, \end{aligned} \tag{3}$$

where A_i the support set of feature positions in which the mean and standard deviation are pooled, and $M = |A_i|$ the size of the set. ϵ is a small positive constant to ensure numeric stability.

In BN [9], the mean and standard deviation are channel-wise and pooled from spatial positions and across all sample in a mini-batch, so we have the support set defined by $A_i = \{j | j_C = i_C\}$. The scale and offset parameters are also channel-wise, i.e., $\gamma_i = \gamma_{i_C}$ and $\beta_i = \beta_{i_C}$.

Similarly, we can define the support set A_i and affine transformation parameters for other feature normalization schema, IN [18], LN [1] and GN [23].

Some interesting questions naturally arise, which have not been studied well,

- What are the underlying effects and impacts of the affine transformation in the dynamics of training and w.r.t. model generalization?

- Would the performance of a normalization scheme be improved if we exploit multiple affine transformations for re-scaling and re-shifting normalized feature responses? And, how to realize it in a lightweight way?

3.2. Background on the SE Unit

The SE unit learns channel-wise attention weights for recalibrating the input feature responses. It consists of two components:

- The sequeeze module encodes the interdependencies between feature channels using a latent vector in a low dimensional space with the reduction rate r (e.g., $r = 16$),
$$F_{sq}(x) = v,\ v \in R^{N\times\frac{C}{r}\times 1\times 1} \tag{4}$$
For example, F_{sq} is implemented by a sub-network consisting of a global average pooling layer, a FC layer and ReLU.

- The excitation module computes the channel-wise attention weights by decoding the learned interdependency latent vector v,
$$s = F_{ex}(v),\ s \in R^{N\times C\times 1\times 1} \tag{5}$$
For example, F_{ex} is implemented by a subnetwork consisting of FC layer and sigmoid transformation

Then, we recalibrate feature response to reflect the learned attention, $x'_i = s_{i_n, i_c} \cdot x_i$. We note that both v and s are conditional on the input feature map, thus dynamic and adaptive (instance-specific), and they can be computed in a lightweight way. If we want to introduce a mixture of affine transformations as stated above, it is natural to consider using the full SE unit or just the squeeze module (to be more efficient) to learn the coefficients of the mixture.

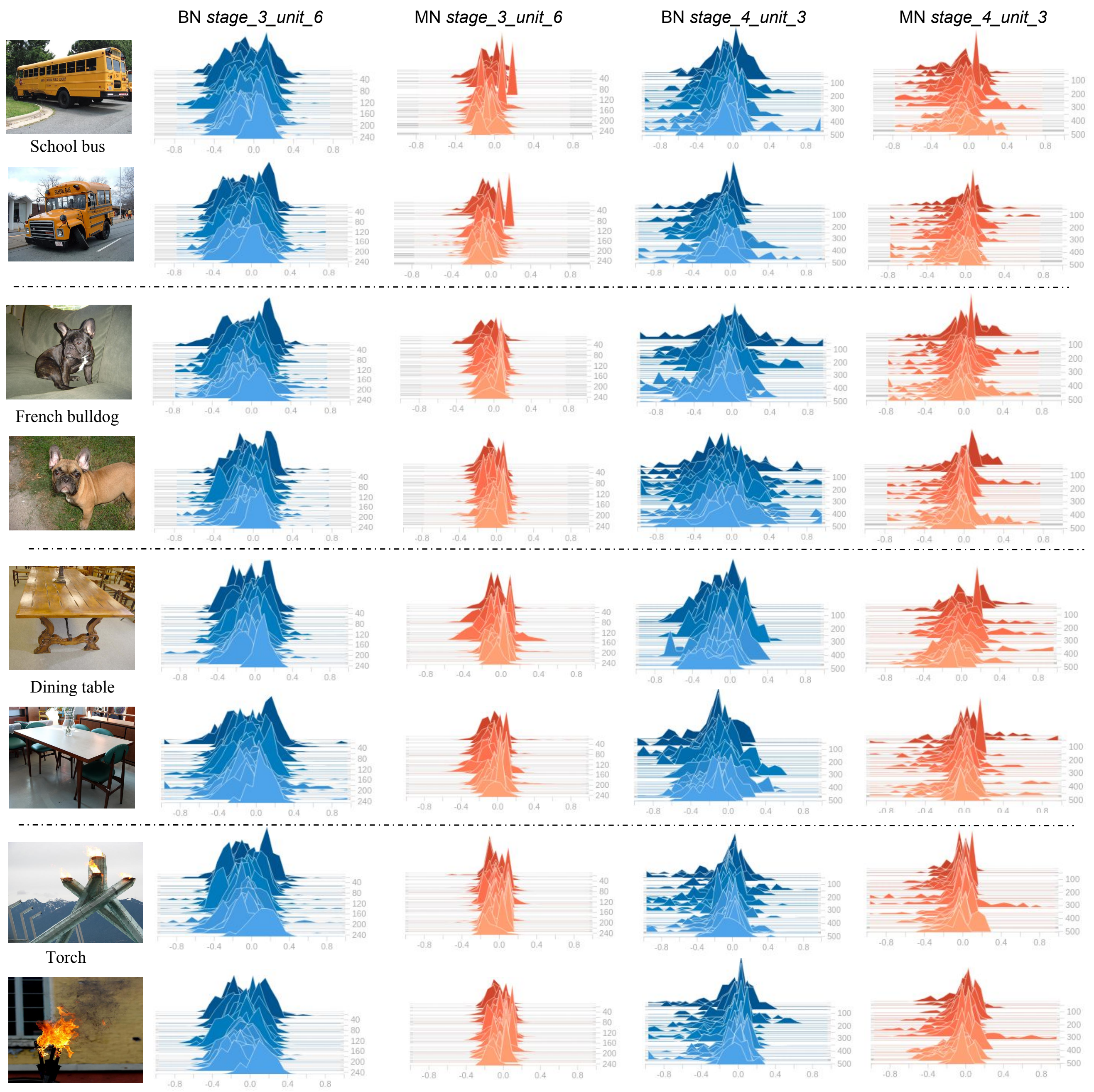

Figure 2. Illustration of the effects of AN and BN on filter responses. We show the filter response histograms (marginal distributions) for different images in different categories. Here we show results of a 4-stage ResNet50. $stage\_i\_unit\_j$ means the histograms are plot for the output feature map of the j-th ResBlock in the i-th stage. From the histograms, we observe that for images from the same class (e.g., school bus), the histograms of our AN show higher similarities with smaller variance. This empirically shows that a channel-wise attention guided mixture of affine transformation helps recalibrate the normalized responses in a more meaningful way.

3.3. Formulation of AN

Consider BN as the backbone, our AN computes normalized feature responses using Eq. 1 (Figure 1). Let K the predefined number of mixture components of affine transformations to be learned in a AN (e.g., $K = 10$). We have the scale parameters $\gamma \in R^{K\times C}$ and the offset parameters $\beta \in R^{K\times C}$. Using a method similar to the squeeze module, AN implements the final affine transformation by,

- Computing the instance-specific mixture coefficients

```
class AttenNorm(nn.BatchNorm2d):
def __init__(self, C, K, eps, momentum, running):
super(AttenNorm, self).__init__(C, eps=eps,
        momentum=momentum, affine=False,
        track_running_stats=running)

self.gamma = nn.Parameter(torch.Tensor(K, C))
self.beta = nn.Parameter(torch.Tensor(K, C))

self.avgpool = nn.AdaptiveAvgPool2d(1)
self.fc = nn.Linear(C, K)
self.sigmoid = nn.Sigmoid()

def forward(self, x):
output = super(AttenNorm, self).forward(x)
size = output.size()

b, c, _, _ = x.size()
y = self.avgpool(x).view(b, c)
y = self.fc(y)
y = self.sigmoid(y)

gamma = y @ self.gamma
beta = y @ self.beta
gamma = weight.unsqueeze(-1).unsqueeze(-1).expand(size)
beta = bias.unsqueeze(-1).unsqueeze(-1).expand(size)

return gamma * output + beta
```

Figure 3. Complete PyTorch code for implementing our AN using BN as the backbone.

by,

$$F_{AN}(x) = \lambda,\ \lambda \in R^{N\times K\times 1\times 1} \tag{6}$$

In our experiments, we implement F_{AN} by a subnetwork consisting of a global average pooling layer, a FC layer and sigmoid transformation. We can also utilize other lightweight subnetworks to learn the weights. For example, we can apply 1×1 convolution from C input channels to K output channels, ReLU and then global average pooling and sigmoid.

- Recalibrating normalized feature responses by,

$$\tilde{x}_i^{AN} = \sum_{k=1}^{K} \lambda_{i_N,k} \cdot [\gamma_{k,i_C} \cdot \hat{x}_i + \beta_{k,i_C}] \tag{7}$$

To illustrate the effects of AN and BN on filter responses, Figure 2 shows empirical comparisons between our AN and the vanilla BN. More analyses as done in [17] will be helpful for gaining detailed insights of how AN helps optimization, which we will do thorough comparisons.

Implementing AN is easy in modern deep learning code framework. For example, Figure 3 shows the complete PyTorch code for AN using BN as the backbone.

Integrating AN in CNNs. Our AN can be used as a drop-in replacement for BN (or other normalization schema) in any existing networks. As Figure 1 shows, we choose to deploy our AN to minimize the overhead. In the Bottleneck operation of ResNets [5], it follows the 3×3 convolution unit since it has the least number of channels.

Method	#Params	FLOPS	top-1	top-5
ResNet-50-BN	25.56M	4.09	23.01	6.68
ResNet-50-GN	25.56M	4.09	23.52	6.85
ResNet-50-SN (8,32)	25.56M	-	22.43	6.35
ResNet-50-SE	28.09 M	-	22.37	6.36
ResNet-50-AN	25.69M	4.09	**22.00**	**6.06**
ResNet-101-BN	44.57M	8.12	20.71	5.43
ResNet-101-AN	44.71M	8.12	**20.06**	**5.12**
DenseNet-161-BN	28.73M	8.50	22.35	6.20
DenseNet-161-AN	30.28M	8.50	**20.13**	**4.94**
MobileNet-v2-BN	3.50M	0.335	28.69	9.33
MobileNet-v2-AN	3.56M	0.335	**26.67**	**8.56**

Table 1. The top-1 and top-5 error rates (%) on the ImageNet-1K validation set using single model and single-crop testing.

4. Experiments

In experiments, we test our AN using ResNet50 [5] in ImageNet-1K [15] and MS-COCO [11]. Since our objective in this paper is to seek lightweight ways of improving BN (or others), and admittedly, due to the high-demand of GPU resources for training and testing on the two datasets, we only test AN using BN as the backbone, and do not test it for different network architectures (such as ResNet101 and DenseNets [7]) with different batch sizes in image classification experiments.

Implementation details. Following the best practice used in BigGAN [2], we initialize the scale and offset parameters in our AN using $\gamma_{k,c} = 1.0 + \mathcal{N}(0,1) \times 0.1$ and $\beta_{k,c} = \mathcal{N}(0,1) \times 0.1$.

4.1. Image Classification in ImageNet

The ILSVRC2012 benchmark consists of about 1.2 million images for training, and $50,000$ for validation, from $1,000$ classes. We adopt the same basic data augmentation scheme (random crop and horizontal flip) for training images as done in [5], and apply a single-crop with size 224×224 at test time. Following the common protocol, we evaluate the top-1 and top-5 classification error rates on the validation set.

We use 4 GPUs (NVIDIA V100) to train all models (ResNet50+BN, ResNet50+GN and ResNet50+AN) under the same settings. Our retrained models of BN and GN are better than those in original papers, so we do not include the original results. For BN and GN, we follow the implementation details of GN [23] to initialize parameters. The batch size is 128 per GPU with FP16 optimization used in training to reduce the training time (although using 8 GPUs and batch size 32 per GPU is the best practice in training ResNets in ImageNet). The initial learning rate is 0.2, and

	Backbone	Head	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{m}	AP^{m}_{50}	AP^{m}_{75}
ResNet-50	BN*	-	38.6	59.8	42.1	34.5	56.4	36.3
	GN	GN	40.3	61.0	44.0	35.7	57.9	37.7
	SN	SN†	41.0	62.3	45.1	36.5	58.9	38.7
	AN (w/ BN)	-	40.5	62.4	44.1	36.5	59.2	38.5
	AN (w/ BN)	AN (w/ GN)	**41.5**	62.2	45.3	**37.1**	59.6	39.5
ResNet-101	BN*	-	40.3	61.5	44.1	36.5	58.1	39.1
	GN	GN	41.8	62.5	45.4	36.8	59.2	39.0
	AN (w/ BN)	-	43.5	64.5	47.4	38.3	60.9	41.0
	AN (w/ BN)	AN (w/ GN)	**43.7**	64.7	48.1	**39.3**	61.6	42.7

Table 2. Detection and segmentation results in COCO, using Mask R-CNN. All models use 2x lr scheduling (180k iterations). BN* means BN is frozen in fine-tuning for object detection. SN† means that only LayerNorm and InstanceNorm are used in the SN based on the latest code and configuration (line 24) . In our implementation of the AN head, AN (w/ GN) means that we use the mixture version of GN in the head.

Model	Backbone	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{m}	AP^{m}_{50}	AP^{m}_{75}
Cascade Mask-RCNN	ResNet-101	43.3			37.6		
	ResNet-101+AN	46.4	65.0	50.4	39.9	62.3	43.2
HTC	ResNet-101	44.9			39.4		
	ResNet-101+AN	46.9	66.4	51.0	40.9	63.6	44.2

Table 3. Detection and segmentation results in COCO, using Cascade Mask R-CNN/Hybrid Task Cascade (HTC).

	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{m}	AP^{m}_{50}	AP^{m}_{75}
MobileNet-v2	34.2	54.6	37.1	30.9	51.1	32.6
MobileNet-v2+AN	36.0	57.0	38.9	32.5	53.8	34.5

Table 4. Detection and segmentation results in COCO, using Mask R-CNN with light-weight backbones.

the cosine learning rate scheduler [12] is used with weight decay 1×10^{-4} and momentum 0.9. We train the models for 120 epochs.

Table 1 shows the comparison results. Our AN obtains the best top-1 and top-5 accuracy results with negligible extra parameters (0.13M) at almost no extra computational cost. Our AN improves BN by almost 1% on top-1, and outperforms SN by 0.4%, which shows the effectiveness of our lightweight integration of feature normalization and attention.

4.2. Object Detection and Segmentation in COCO

We test our AN in the COCO train2017 set and evaluated in the COCO val2017 set (a.k.a minival). We report the standard COCO metrics of Average Precision (AP), AP_{50}, and AP_{75}, for bounding box detection (AP^{bb}) and instance segmentation (AP^{m}).

When fine-tuning the ImageNet pretrained ResNet50+AN on COCO for object detection and segmentation, we freeze all the gamma and beta parameters and the tracked running mean and variance, but allow the FC layers to continue learn except for the FC layer in the first stage. With only AN in the backbone, our AN obtains comparable performance to the model with GN in both backbone and the head classifiers although GN stands right in its sweet pot. This shows that our AN does not suffer too much from the small batch settings in fine-tuning. We conjecture that the FC layers can compensate the small batch issue by learning instance-specific feature channel-wise attention. Compared with the vanilla BN, we significantly improve the performance, which shows the effectiveness of integrating feature normalization and attention in transferring models between different tasks.

5. Conclusion

This paper presents a lightweight integration of feature normalization and attention. It connects the affine transformation in feature normalization schema with the channel-wise attention based recalibration of feature responses. It presents Attentive Normalization (AN) as a simple and unified alternative for feature normalization and attention. It also shows a lightweight deployment of the proposed AN in the Bottleneck operation. To our best knowledge, this is the first work that integrates feature normalization and attention in a lightweight way. The proposed method is validated by testing AN using Batch Normalization (BN) as the backbone in ImageNet and MS-COCO. It obtains better performance than state-of-the-art variants of BN, Group Normalization (GN) and Switchable Normalization (SN). It also shows interpretable visualization justifying the effectiveness of AN. The proposed method is complementary to existing variants of BN and applicable to extending them to corresponding Attentive versions.

References

[1] L. J. Ba, R. Kiros, and G. E. Hinton. Layer normalization. *CoRR*, abs/1607.06450, 2016. 1, 3, 4

[2] A. Brock, J. Donahue, and K. Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018. 1, 3, 6

[3] H. de Vries, F. Strub, J. Mary, H. Larochelle, O. Pietquin, and A. C. Courville. Modulating early visual processing by language. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6597–6607, 2017. 1, 3

[4] V. Dumoulin, I. Belghazi, B. Poole, A. Lamb, M. Arjovsky, O. Mastropietro, and A. C. Courville. Adversarially learned inference. *CoRR*, abs/1606.00704, 2016. 1, 3

[5] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 2, 3, 6

[6] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. *CoRR*, abs/1709.01507, 2017. 2, 3, 4

[7] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 6

[8] Z. Huang, X. Wang, L. Huang, C. Huang, Y. Wei, and W. Liu. Ccnet: Criss-cross attention for semantic segmentation. *CoRR*, abs/1811.11721, 2018. 2

[9] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In D. Blei and F. Bach, editors, *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 448–456. JMLR Workshop and Conference Proceedings, 2015. 1, 2, 3, 4

[10] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. *arXiv preprint arXiv:1812.04948*, 2018. 1, 3

[11] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. 1, 3, 6

[12] I. Loshchilov and F. Hutter. SGDR: stochastic gradient descent with restarts. *CoRR*, abs/1608.03983, 2016. 7

[13] P. Luo, J. Ren, and Z. Peng. Differentiable learning-to-normalize via switchable normalization. *CoRR*, abs/1806.10779, 2018. 1, 3

[14] T. Miyato and M. Koyama. cgans with projection discriminator. *arXiv preprint arXiv:1802.05637*, 2018. 1

[15] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vision (IJCV)*, 115(3):211–252, 2015. 1, 3, 6

[16] T. Salimans and D. P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, page 901, 2016. 3

[17] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry. How does batch normalization help optimization? In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pages 2488–2498, 2018. 1, 6

[18] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *CoRR*, abs/1607.08022, 2016. 1, 3, 4

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6000–6010, 2017. 3

[20] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang. Residual attention network for image classification. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 6450–6458, 2017. 3

[21] X. Wang, R. B. Girshick, A. Gupta, and K. He. Non-local neural networks. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 7794–7803, 2018. 2

[22] S. Woo, J. Park, J. Lee, and I. S. Kweon. CBAM: convolutional block attention module. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part VII*, pages 3–19, 2018. 3

[23] Y. Wu and K. He. Group normalization. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XIII*, pages 3–19, 2018. 1, 3, 4, 6

[24] H. Zhang, Y. N. Dauphin, and T. Ma. Fixup initialization: Residual learning without normalization. *CoRR*, abs/1901.09321, 2019. 1, 2