Attentive Normalization

Xilai Li, Wei Sun, and Tianfu Wu *

Department of Electrical and Computer Engineering, NC State University
{xli47, wsun12, tianfu_wu}@ncsu.edu

Abstract. In state-of-the-art deep neural networks, both feature normalization and feature attention have become ubiquitous. They are usually studied as separate modules, however. In this paper, we propose a light-weight integration between the two schema and present Attentive Normalization (AN). Instead of learning a single affine transformation, AN learns a mixture of affine transformations and utilizes their weighted-sum as the final affine transformation applied to re-calibrate features in an instance-specific way. The weights are learned by leveraging channel-wise feature attention. In experiments, we test the proposed AN using four representative neural architectures in the ImageNet-1000 classification benchmark and the MS-COCO 2017 object detection and instance segmentation benchmark. AN obtains consistent performance improvement for different neural architectures in both benchmarks with absolute increase of top-1 accuracy in ImageNet-1000 between 0.5% and 2.7%, and absolute increase up to 1.8% and 2.2% for bounding box and mask AP in MS-COCO respectively. We observe that the proposed AN provides a strong alternative to the widely used Squeeze-and-Excitation (SE) module. The source codes are publicly available at the ImageNet Classification Repo and the MS-COCO Detection and Segmentation Repo.

1 Introduction

Pioneered by Batch Normalization (BN) [19], feature normalization has become ubiquitous in the development of deep learning. Feature normalization consists of two components: *feature standardization* and *channel-wise affine transformation*. The latter is introduced to provide the capability of undoing the standardization (by design), and can be treated as *feature re-calibration* in general. Many variants of BN have been proposed for practical deployment in terms of variations of training and testing settings with remarkable progress obtained. They can be roughly divided into two categories:

i) Generalizing feature standardization. Different methods are proposed for computing the mean and standard deviation or for modeling/whitening the data distribution in general, within a min-batch. They include Batch Renormalization [18], Decorrelated BN [16], Layer Normalization (LN) [1], Instance Normalization (IN) [42], Instance-level Meta Normalization [20], Group Normalization (GN) [47], Mixture Normalization [21] and Mode Normalization [5]. Switchable

* T. Wu is the corresponding author.

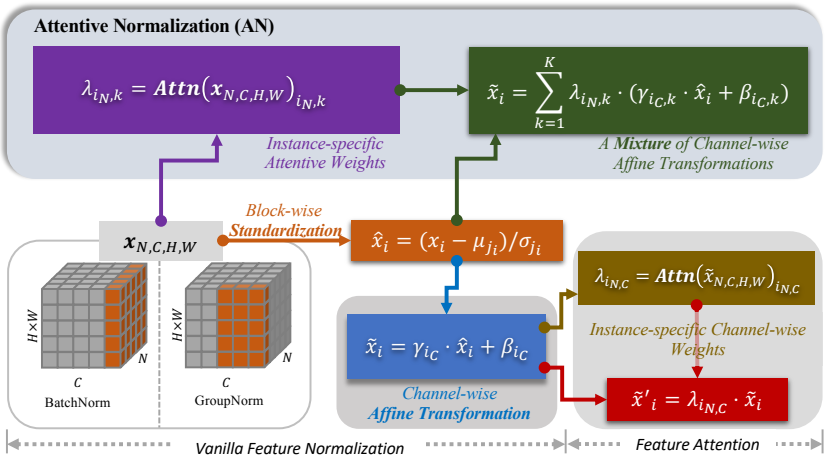

Fig. 1: Illustration of the proposed Attentive Normalization (AN). AN aims to harness the best of a base feature normalization (e.g., BN or GN) and channel-wise feature attention in a single light-weight module. See text for details.

Normalization (SN) [28] and its sparse variant (SSN) [39] learn to switch between different vanilla schema. These methods adopt the vanilla channel-wise affine transformation after standardization, and are often proposed for discriminative learning tasks.

ii) Generalizing feature re-calibration. Instead of treating the affine transformation parameters directly as model parameters, different types of task-induced conditions (*e.g.*, class labels in conditional image synthesis using generative adversarial networks) are leveraged and encoded as latent vectors, which are then used to learn the affine transformation parameters, including different conditional BNs [6,43,33,29,2], style-adaptive IN [22] or layout-adaptive IN [31,40]. These methods have been mainly proposed in generative learning tasks, except for the recently proposed Instance-level Meta Normalization [20] in discriminative learning tasks.

In the meanwhile, *feature attention* has also become an indispensable mechanism for improving task performance in deep learning. For computer vision, spatial attention is inherently captured by convolution operations within short-range context, and by non-local extensions [45,17] for long-range context. Channel-wise attention is relatively less exploited. The squeeze-and-excitation (SE) unit [13] is one of the most popular designs, which learn instance-specific channel-wise attention weights to re-calibrate an input feature map. Unlike the affine transformation parameters in feature normalization, the attention weights for re-calibrating an feature map are often directly learned from the input feature map in the spirit of self-attention, and often instance-specific or pixel-specific.

Although both feature normalization and feature attention have become ubiquitous in state-of-the-art DNNs, they are usually studied as separate mod-

ules. Therefore, in this paper we address the following problem: *How to learn to re-calibrate feature maps in a way of harnessing the best of feature normalization and feature attention in a single light-weight module?* And, we present **Attentive Normalization (AN)**: Fig. 1 illustrates the proposed AN. The basic idea is straightforward. Conceptually, the affine transformation component in feature normalization (Section 3.1) and the re-scaling computation in feature attention play the same role in learning-to-re-calibrate an input feature map, thus providing the foundation for integration (Section 3.2). More specifically, consider a feature normalization backbone such as BN or GN, our proposed AN keeps the block-wise standardization component unchanged. Unlike the vanilla feature normalization in which the affine transformation parameters (γ's and β's) are often frozen in testing, we want the affine transformation parameters to be adaptive and dynamic in both training and testing, controlled directly by the input feature map. The intuition behind doing so is that it will be more flexible in accounting for different statistical discrepancies between training and testing in general, and between different sub-populations caused by underlying inter-/intra-class variations in the data.

To achieve the dynamic and adaptive control of affine transformation parameters, the proposed AN utilizes a simple design (Section 3). It learns a mixture of K affine transformations and exploits feature attention mechanism to learn the instance-specific weights for the K components. The final affine transformation used to re-calibrate an input feature map is the weighted sum of the learned K affine transformations. We propose a general formulation for the proposed AN and study how to learn the weights in an efficient and effective way (Section 3.3).

2 Related Work

Feature Normalization. There are two types of normalization schema, feature normalization (including raw data) [19,18,1,42,47,28,39,21,5] and weight normalization [36,15]. Unlike the former, the latter is to normalize model parameters 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 [19] computes the channel-wise mean and variance in the entire min-batch which is driven by improving training efficiency and model generalizability. BN has been deeply analyzed in terms of how it helps optimization [38]. DecorBN [16] utilizes a whitening operation (ZCA) to go beyond the centering and scaling in the vanilla BN. BatchReNorm [18] introduces extra parameters to control the pooled mean and variance to reduce BN's dependency on the batch size. IN [42] 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 [47] stands in the sweet spot between LN and IN focusing on instance-specific and channel-group-wise statistics, especially when only small batches are applicable in practice. In practice, synchronized BN [32] across multiple GPUs becomes increasingly favorable against GN in some applications. SN [28] leaves the design choices of fea-

ture 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, followed by SSN [39] which learns to make exclusive selection. Instead of computing one mode (mean and variance), MixtureNorm [21] introduces a mixture of Gaussian densities to approximate the data distribution in a mini-batch. ModeNorm [5] utilizes a general form of multiple-mode computation. Unlike those methods, the proposed AN focuses on generalizing the affine transformation component. Related to our work, Instance-level Meta normalization(ILM) [20] first utilizes an encoder-decoder sub-network to learn affine transformation parameters and then add them together to the model's affine transformation parameters. Unlike ILM, the proposed AN utilizes a mixture of affine transformations and leverages feature attention to learn the instance-specific attention weights.

On the other hand, conditional feature normalization schema [6,43,33,2,22,31] [40] have been developed and shown remarkable progress in conditional and unconditional image synthesis. Conditional BN learns condition-specific affine transformations in terms of conditions such as class labels, image style, label maps and geometric layouts. Unlike those methods, the proposed AN learns self-attention data-driven weights for mixture components of affine transformations.

Feature Attention. Similar to feature normalization, feature attention is also an important building block in the development of deep learning. Residual Attention Network [44] uses a trunk-and-mask joint spatial and channel attention module in an encoder-decoder style for improving performance. To reduce the computational cost, channel and spatial attention are separately applied in [46]. The SE module [13] further simplifies the attention mechanism by developing a light-weight channel-wise attention method. The proposed AN leverages the idea of SE in learning attention weights, but formulates the idea in a novel way.

Our Contributions. This paper makes three main contributions: (i) It presents Attentive Normalization which harnesses the best of feature normalization and feature attention (channel-wise). To our knowledge, AN is the first work that studies self-attention based conditional and adaptive feature normalization in visual recognition tasks. (ii) It presents a lightweight integration method for deploying AN in different widely used building blocks of ResNets, DenseNets, MobileNetsV2 and AOGNets. (iii) It obtains consistently better results than the vanilla feature normalization backbones by a large margin across different neural architectures in two large-scale benchmarks, ImageNet-1000 and MS-COCO.

3 The Proposed Attentive Normalization

In this section, we present details of the proposed attentive normalization. Consider a DNN for 2D images, denote by $\mathbf{x}$ a feature map with axes in the convention order of (N, C, H, W) (i.e., batch, channel, height and width). $\mathbf{x}$ is represented by a 4D tensor. Let $i = (i_N, i_C, i_H, i_W)$ be the address index in the 4D tensor. $\mathbf{x}_i$ represents the feature response at a position i.

3.1 Background on Feature Normalization

Existing feature normalization schema often consist of two components (Fig. 1):

i) Block-wise Standardization. Denote by B_j a block (slice) in a given 4-D tensor $\mathbf{x}$. For example, for BN, we have $j = 1, \cdots, C$ and $B_j = \{\mathbf{x}_i | \forall i, i_C = j\}$. We first compute the empirical mean and standard deviation in B_j, denoted by μ_j and σ_j respectively: $\mu_j = \frac{1}{M}\sum_{x \in B_j} x, \quad \sigma_j = \sqrt{\frac{1}{M}\sum_{x \in B_j}(x - \mu_j)^2 + \epsilon}$, where $M = |B_j|$ and ϵ is a small positive constant to ensure $\sigma_j > 0$ for the sake of numeric stability. Then, let j_i be the index of the block that the position i belongs to, and we standardize the feature response by,

$$\hat{\mathbf{x}}_i = \frac{1}{\sigma_{j_i}}(\mathbf{x}_i - \mu_{j_i}) \tag{1}$$

ii) Channel-wise Affine Transformation. Denote by γ_c and β_c the scalar coefficient (re-scaling) and offset (re-shifting) parameter respectively for the c-th channel. The re-calibrated feature response at a position i is then computed by,

$$\tilde{\mathbf{x}}_i = \gamma_{i_C} \cdot \hat{\mathbf{x}}_i + \beta_{i_C}, \tag{2}$$

where γ_c's and β_c's are shared by all the instances in a min-batch across the spatial domain. They are usually frozen in testing and fine-tuning.

3.2 Background on Feature Attention

We focus on channel-wise attention and briefly review the Squeeze-Excitation (SE) module [13]. SE usually takes the feature normalization result (Eqn. 2) as its input (the bottom-right of Fig. 1), and learns channel-wise attention weights:

i) The squeeze module encodes the inter-dependencies between feature channels in a low dimensional latent space with the reduction rate r (e.g., $r = 16$),

$$S(\tilde{\mathbf{x}}; \theta_S) = v, \; v \in \mathbb{R}^{N \times \frac{C}{r} \times 1 \times 1}, \tag{3}$$

which is implemented by a sub-network consisting of a global average pooling layer (AvgPool), a fully-connected (FC) layer and rectified linear unit (ReLU) [23]. θ_S collects all the model parameters.

ii) The excitation module computes the channel-wise attention weights, denoted by λ, by decoding the learned latent representations v,

$$E(v; \theta_E) = \lambda, \; \lambda \in \mathbb{R}^{N \times C \times 1 \times 1}, \tag{4}$$

which is implemented by a sub-network consisting of a FC layer and a sigmoid layer. θ_E collects all model parameters.

Then, the input, $\tilde{\mathbf{x}}$ is re-calibrated by,

$$\tilde{\mathbf{x}}_i^{SE} = \lambda_{i_N, i_C} \cdot \tilde{\mathbf{x}}_i = (\lambda_{i_N, i_C} \cdot \gamma_{i_C}) \cdot \hat{\mathbf{x}}_i + \lambda_{i_N, i_C} \cdot \beta_{i_C}, \tag{5}$$

where the second step is obtained by plugging in Eqn. 2. **It is thus straightforward to see the foundation facilitating the integration between feature normalization and channel-wise feature attention.** However, the SE module often entails a significant number of extra parameters (e.g., ∼2.5M extra parameters for ResNet50 [10] which originally consists of ∼25M parameters, resulting in 10% increase). We aim to design more parsimonious integration that can further improve performance.

3.3 Attentive Normalization

Our goal is to generalize Eqn. 2 in re-calibrating feature responses to enable dynamic and adaptive control in both training and testing. On the other hand, our goal is to simplify Eqn. 5 into a single light-weight module, rather than, for example, the two-module setup using BN+SE. In general, we have,

$$\tilde{\mathbf{x}}_i^{AN} = \mathit{\Gamma}(\mathbf{x};\theta_{\mathit{\Gamma}})_i \cdot \hat{\mathbf{x}}_i + \mathbb{B}(\mathbf{x};\theta_{\mathbb{B}})_i, \tag{6}$$

where both $\mathit{\Gamma}(\mathbf{x};\theta_{\mathit{\Gamma}})$ and $\mathbb{B}(\mathbf{x};\theta_{\mathbb{B}})$ are functions of the entire input feature map (without standardization [1]) with parameters $\theta_{\mathit{\Gamma}}$ and $\theta_{\mathbb{B}}$ respectively. They both compute 4D tensors of the size same as the input feature map and can be parameterized by some attention guided light-weight DNNs. The subscript in $\mathit{\Gamma}(\mathbf{x};\theta_{\mathit{\Gamma}})_i$ and $\mathbb{B}(\mathbf{x};\theta_{\mathbb{B}})_i$ represents the learned re-calibration weights at a position i.

In this paper, we focus on learning instance-specific channel-wise affine transformations. To that end, we have three components as follows.

i) Learning a Mixture of K Channel-wise Affine Transformations. Denote by $\gamma_{k,c}$ and $\beta_{k,c}$ the re-scaling and re-shifting (scalar) parameters respectively for the c-th channel in the k-th mixture component. They are model parameters learned end-to-end via back-propagation.

ii) Learning Attention Weights for the K Mixture Components. Denote by $\lambda_{n,k}$ the instance-specific mixture component weight ($n \in [1, N]$ and $k \in [1, K]$), and by λ the $N \times K$ weight matrix. λ is learned via some attention-guided function from the entire input feature map,

$$\lambda = A(\mathbf{x};\theta_{\lambda}), \tag{7}$$

where θ_{λ} collects all the parameters.

iii) Computing the Final Affine Transformation. With the learned $\gamma_{k,c}$, $\beta_{k,c}$ and λ, the re-calibrated feature response is computed by,

$$\tilde{\mathbf{x}}_i^{AN} = \sum_{k=1}^{K} \lambda_{i_N,k}[\gamma_{k,i_C} \cdot \hat{\mathbf{x}}_i + \beta_{k,i_C}], \tag{8}$$

where $\lambda_{i_N,k}$ is shared by the re-scaling parameter and the re-shifting parameter for simplicity. Since the attention weights λ are adaptive and dynamic in both training and testing, the proposed AN realizes adaptive and dynamic feature re-calibration. Compared to the general form (Eqn. 6), we have,

$$\mathit{\Gamma}(\mathbf{x})_i = \sum_{k=1}^{K} \lambda_{i_N,k} \cdot \gamma_{k,i_C}, \ \mathbb{B}(\mathbf{x})_i = \sum_{k=1}^{K} \lambda_{i_N,k} \cdot \beta_{k,i_C}. \tag{9}$$

Based on the formulation, there are **a few advantages of the proposed AN in training, fine-tuning and testing** a DNN:

[1] We tried the variant of learning $\mathit{\Gamma}()$ and $\mathbb{B}()$ from the standardized features and observed it works worse, so we ignore it in our experiments.

- The channel-wise affine transformation parameters, γ_{k,i_C}'s and β_{k,i_C}'s, are shared across spatial dimensions and by data instances, which can learn population-level knowledge in a more fine-grained manner than a single affine transformation in the vanilla feature normalization.
- $\lambda_{i_N,k}$'s are instance specific and learned from features that are not standardized. Combining them with γ_{k,i_C}'s and β_{k,i_C}'s (Eqn. 8) enables AN paying attention to both the population (what the common and useful information are) and the individuals (what the specific yet critical information are). The latter is particularly useful for testing samples slightly "drifted" from training population, that is to improve generalizability. Their weighted sum encodes more direct and "actionable" information for re-calibrating standardized features (Eqn. 8) without being delayed until back-propagation updates as done in the vanilla feature normalization.
- In fine-tuning, especially between different tasks (*e.g.*, from image classification to object detection), γ_{k,i_C}'s and β_{k,i_C}'s are usually frozen as done in the vanilla feature normalization. They carry information from a source task. But, θ_λ (Eqn. 7) are allowed to be fine-tuned, thus potentially better realizing transfer learning for a target task. This is a desirable property since we can decouple training correlation between tasks. For example, when GN [47] is applied in object detection in MS-COCO, it is fine-tuned from a feature backbone with GN trained in ImageNet, instead of the one with BN that usually has better performance in ImageNet. As we shall show in experiments, the proposed AN facilitates a smoother transition. We can use the proposed AN (with BN) as the normalization backbone in pre-training in ImageNet, and then use AN (with GN) as the normalization backbone for the head classifiers in MS-COCO with significant improvement.

Details of Learning Attention Weights We present a simple method for computing the attention weights $A(\mathbf{x};\theta_\lambda)$ (Eqn. 7). Our goal is to learn a weight coefficient for each component from each individual instance in a mini-batch (i.e, a $N \times K$ matrix). The question of interest is how to characterize the underlying importance of a channel c from its realization across the spatial dimensions (H, W) in an instance, such that we will learn a more informative instance-specific weight coefficient for a channel c in re-calibrating the feature map $\mathbf{x}$.

In realizing Eqn. 7, the proposed method is similar in spirit to the squeeze module in SENets [13] to maintain light-weight implementation. To show the difference, let's first rewrite the vanilla squeeze module (Eqn. 3),

$$v = S(\mathbf{x};\theta_S) = ReLU(fc(AvgPool(\mathbf{x});\theta_S))\,, \tag{10}$$

where the mean of a channel c (via global average pooling, $AvgPool(\cdot)$) is used to characterize its underlying importance. We generalize this assumption by taking into account both mean and standard deviation empirically computed for a channel c, denoted by μ_c and σ_c respectively. More specifically, we compare three different designs using:

i) The mean μ_c only as done in SENets.

ii) The concatenation of the mean and standard deviation, (μ_c, σ_c).
iii) The coefficient of variation or the relative standard deviation (RSD), $\frac{\sigma_c}{\mu_c}$. RSD measures the dispersion of an underlying distribution (i.e., the extent to which the distribution is stretched or squeezed) which intuitively conveys more information in learning attention weights for re-calibration.

RSD is indeed observed to work better in our experiments[2]. Eqn. 7 is then expanded with two choices,

$$\begin{aligned} &\textit{Choice 1: } A_1(\mathbf{x};\theta_\lambda) = Act(fc(RSD(\mathbf{x});\theta_\lambda)), \\ &\textit{Choice 2: } A_2(\mathbf{x};\theta_\lambda) = Act(BN(fc(RSD(\mathbf{x});\theta_{fc});\theta_{BN})), \end{aligned} \tag{11}$$

where $Act(\cdot)$ represents a non-linear activation function for which we compare three designs:

i) The vanilla $ReLU(\cdot)$ as used in the squeeze module of SENets.
ii) The vanilla $sigmoid(\cdot)$ as used in the excitation module of SENets.
iii) The channel-wise $softmax(\cdot)$.
iv) The piece-wise linear hard analog of the sigmoid function, so-called *hsigmoid* function [12], $hsigmoid(a) = \min(\max(a + 3.0, 0), 6.0)/6.0$.

The $hsigmoid(\cdot)$ is observed to work better in our experiments. In the Choice 2 (Eqn. 11), we apply the vanilla BN [19] after the FC layer, which normalizes the learned attention weights across all the instances in a mini-batch with the hope of balancing the instance-specific attention weights better. The Choice 2 improves performance in our experiments in ImageNet.

In AN, we have another hyper-parameter, K. For stage-wise building block based neural architectures such as the four neural architectures tested in our experiments, we use different K's for different stages with smaller values for early stages. For example, for the 4-stage setting, we typically use $K = 10, 10, 20, 20$ for the four stages respectively based on our ablation study. The underlying assumption is that early stages often learn low-to-middle level features which are considered to be shared more between different categories, while later stages learn more category-specific features which may entail larger mixtures.

4 Experiments

In this section, we first show the ablation study verifying the design choices in the proposed AN. Then, we present detailed comparisons and analyses.

Data and Evaluation Metric. We use two benchmarks, the ImageNet-1000 classification benchmark (ILSVRC2012) [35] and the MS-COCO object detection and instance segmentation benchmark [26]. The ImageNet-1000 benchmark consists of about 1.28 million images for training, and 50,000 for validation, from 1,000 classes. We apply a single-crop with size 224×224 in evaluation. Following the common protocol, we report the top-1 and top-5 classification error rates tested using a single model on the validation set. For the MS-COCO benchmark,

[2] In implementation, we use the reverse $\frac{\mu_c}{\sigma_c+\epsilon}$ for numeric stability, which is equivalent to the original formulation when combing with the fc layer.

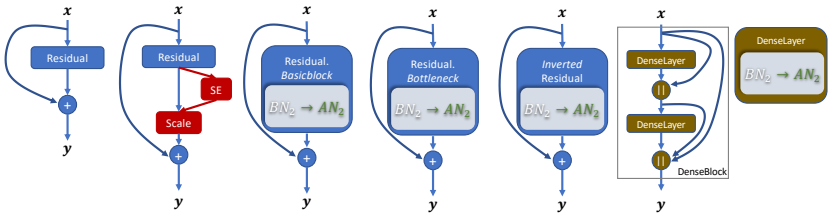

Fig. 2: Illustration of integrating the proposed AN in different building blocks. The first two show the vanilla residual block and the SE-residual block. The remaining four are: the Basicblock and Bottleneck design of a residual block, the inverted residual block (used in MobileNetV2), and the DenseBlock. For the residual block and its variants, the proposed AN is used to replace the vanilla BN(s) followed the last 3×3 convolution in different blocks. This potentially enables jointly integrating local spatial attention (conveyed by the 3×3 convolution) in learning the instance-specific attention weights, which is also observed helpful in [30] and is shown beneficial for the SE module itself in our experiments (Table 3). For the dense block, we replace the second vanilla BN (after the 1×1 convolution applied to the concatenated features) with our AN.

there are 80 categories of objects. We use `train2017` in training and evaluate the trained models using `val2107`. We report the standard COCO metrics of Average Precision (AP) at different intersection-over-union (IoU) thresholds, *e.g.*, AP_{50} and AP_{75}, for bounding box detection (AP^{bb}_{IoU}) and instance segmentation (AP^{m}_{IoU}), and the mean AP over IoU=$0.5 : 0.05 : 0.75$, AP^{bb} and AP^{m} for bounding box detection and instance segmentation respectively.

Neural Architectures and Vanilla Feature Normalization Backbones. We use four representative neural architectures: (i) *ResNets* [10] (ResNet50 and ResNet101), which are the most widely used architectures in practice, (ii) *DenseNets* [14], which are popular alternatives to ResNets, (iii) *MobileNetV2* [37]. MobileNets are popular architectures under mobile settings and MobileNetV2 uses inverted residuals and linear Bottlenecks, and (iv) *AOGNets* [24], which are grammar-guided networks and represent an interesting direction of network architecture engineering with better performance than ResNets and DenseNets. So, the improvement by our AN will be both broadly useful for existing ResNets, DenseNets and MobileNets based deployment in practice and potentially insightful for on-going and future development of more advanced and more powerful DNNs in the community.

In classification, we use BN [19] as the feature normalization backbone for our proposed AN, denoted by **AN (w/ BN)**. We compare with the vanilla BN, GN [47] and SN [28]. In object detection and instance segmentation, we use the Mask-RCNN framework [8] and its cascade variant [3] in the MMDetection code platform [4]. We fine-tune feature backbones pretrained on the ImageNet-1000 dataset. We also test the proposed AN using GN as the feature normalization backbone, denoted by **AN (w/ GN)** in the head classifier of Mask-RCNN.

Where to Apply AN? Fig. 2 illustrates the integration of our proposed AN in different building blocks. At the first thought, it is straightforward to replace all vanilla feature normalization modules (*e.g.*, BN) in a DNN. It may not be necessary to do so, similar in spirit to the SE-residual block which re-calibrates the residual part once in a building block. As we shall see, our ablation study supports the design choice shown in Fig. 2.

Initialization of our AN. The initialization of $\gamma_{k,c}$'s and $\beta_{k,c}$'s (Eqn. 8) is based on, $\gamma_{k,c} = 1.0 + \mathcal{N}(0,1) \times 0.1$ and $\beta_{k,c} = \mathcal{N}(0,1) \times 0.1$, where $\mathcal{N}(0,1)$ represents the standard Gaussian distribution. This type of initialization is also adopted for conditional BN used in the BigGAN [2].

4.1 Ablation Study

We compare different design choices in our proposed AN using ResNet50 in ImageNet-1000. Table 1 summarizes the results. There are four categories of design choices: The first three are related to the realization of learning the attention weights (Eqn. 7): three types of inputs, two architectural choices and four activation function choices. The last one refers to the number K of components in the mixture of affine transformation which is used for each of the four stages in ResNet50 and we empirically select three options for simplicity. All the models are trained using the same settings (the vanilla setup in Section 4.2).

Design Choices in AN (w/ BN)	#Params	FLOPS	top-1	top-5
mean + $A_2(\cdot)$ + hsigmoid + $K = \binom{10}{10}_{20}^{20}$	25.76M	4.09G	21.85	5.92
(mean,std) + $A_2(\cdot)$ + hsigmoid + $K = \binom{10}{10}_{20}^{20}$	25.82M	4.09G	21.73	5.85
RSD + $\mathbf{A}_1(\cdot)$ + hsigmoid + $K = \binom{10}{10}_{20}^{20}$	25.76M	4.09G	21.76	6.05
RSD + $A_2(\cdot)$ + **softmax** + $K = \binom{10}{10}_{20}^{20}$	25.76M	4.09G	21.72	5.90
RSD + $A_2(\cdot)$ + **relu** + $K = \binom{10}{10}_{20}^{20}$	25.96M	4.09G	21.89	6.04
RSD + $A_2(\cdot)$ + **sigmoid** + $K = \binom{10}{10}_{20}^{20}$	25.76M	4.09G	21.96	5.91
RSD + $A_2(\cdot)$ + hsigmoid + $\mathbf{K} = \binom{5}{10}_{20}^{10}$	25.76M	4.09G	21.92	5.93
RSD + $A_2(\cdot)$ + hsigmoid + $\mathbf{K} = \binom{20}{20}_{40}^{40}$	25.96M	4.09G	21.62	5.63
RSD + $A_2(\cdot)$ + hsigmoid + $K = \binom{10}{10}_{20}^{20}$	25.76M	4.09G	**21.59**	**5.58**
* RSD + $A_2(\cdot)$ + hsigmoid + $K = \binom{10}{10}_{20}^{20}$	26.96M	4.10G	22.15	6.24

Table 1: Ablation study on different design choices in AN with BN as feature normalization backbone using ResNet50+Bottleneck in ImageNet-1000. * means AN is applied to all the BNs of the network.

The best combination is RSD + $A_2(\cdot)$ + hsigmoid + $K = \binom{10}{10}_{20}^{20}$. During our development, we first observed the best combination based on our intuitive reasoning and small experiments (a few epochs) in the process, and then design this ablation study to verify the design choices. Based on the observed best combination, we further verify that *replacing all vanilla BNs is not helpful* (the last row in Table 1). One explanation is that we may not need to re-calibrate the features using our AN (as well as other channel-wise feature attention methods) for both before and after a 1×1 convolution, since channel-wise re-calibration can be tackled by the 1×1 convolution kernel and the vanilla feature normalization

themselves in training. The ablation study is in support of the intuitions and design choices discussed in Section 3.3.

4.2 Image Classification in ImageNet-1000

Common Training Settings. We use 8 GPUs (NVIDIA V100) to train models using the same settings for apple-to-apple comparisons. The method proposed in [9] is used to initialize all convolutions for all models. The batch size is 128 per GPU. with FP16 optimization used in training to reduce the training time. The mean and standard deviation for block-wise standardization are computed *within* each GPU. The initial learning rate is 0.4, and the cosine learning rate scheduler [27] is used with 5 warm-up epochs and weight decay 1×10^{-4} and momentum 0.9. For AN, the best practice observed in our ablation study (Table 1) is used. AN is not used in the stem layer in all the models. In addition to the common settings, we have two different setups in experimental comparisons:

i) The Vanilla Setup. We adopt the basic data augmentation scheme (random crop and horizontal flip) in training as done in [10]. We train the models for 120 epochs. All ResNets [10] use the vanilla stem layer with 7×7 convolution. The MobileNetsV2 uses 3×3 convolution in the stem layer. The AOGNets use two consecutive 3×3 convolution in the stem layer. All the γ and β parameters of the feature normalization backbones are initialized to 1 and 0 respectively.

The Vanilla Setup				
Method	#Params	FLOPS	top-1	top-5
ResNet34+BN	21.80M	3.68G	$25.58_{\downarrow(1.15)}$	$8.19_{\downarrow(0.76)}$
ResNet34+AN	21.92M	3.68G	**24.43**	**7.43**
ResNet50-BN	25.56M	4.09G	$23.01_{\downarrow(1.42)}$	$6.68_{\downarrow(0.80)}$
†ResNet50-GN [47]	25.56M	4.09G	$23.52_{\downarrow(1.93)}$	$6.85_{\downarrow(0.97)}$
†ResNet50-SN [28]	25.56M	-	$22.43_{\downarrow(0.83)}$	$6.35_{\downarrow(0.47)}$
†ResNet50-SE [13]	28.09M	4.12G	$22.37_{\downarrow(0.78)}$	$6.36_{\downarrow(0.48)}$
ResNet50-SE	28.09M	4.12G	$22.35_{\downarrow(0.76)}$	$6.09_{\downarrow(0.21)}$
ResNet50-AN	25.76M	4.09G	**21.59**	**5.88**
ResNet101-BN	44.57M	8.12G	$21.33_{\downarrow(0.72)}$	$5.85_{\downarrow(0.44)}$
ResNet101-AN	45.00M	8.12G	**20.61**	**5.41**
DenseNet121-BN	7.98M	2.86G	$25.35_{\downarrow(2.73)}$	$7.83_{\downarrow(1.41)}$
DenseNet121-AN	8.34M	2.86G	**22.62**	**6.42**
MobileNetV2-BN	3.50M	0.34G	$28.69_{\downarrow(2.02)}$	$9.33_{\downarrow(0.77)}$
MobileNetV2-AN	3.56M	0.34G	**26.67**	**8.56**
AOGNet12M-BN	12.26M	2.19G	$22.22_{\downarrow(0.94)}$	$6.06_{\downarrow(0.30)}$
AOGNet12M-AN	12.37M	2.19G	**21.28**	**5.76**
AOGNet40M-BN	40.15M	7.51G	$19.84_{\downarrow(0.51)}$	$4.94_{\downarrow(0.22)}$
AOGNet40M-AN	40.39M	7.51G	**19.33**	**4.72**
The State-of-the-Art Setup				
Method	#Params	FLOPS	top-1	top-5
ResNet50-BN	25.56M	4.09G	$21.08_{\downarrow(1.16)}$	$5.56_{\downarrow(0.52)}$
ResNet50-AN	25.76M	4.09G	**19.92**	**5.04**
ResNet101-BN	44.57M	8.12G	$19.71_{\downarrow(0.86)}$	$4.89_{\downarrow(0.26)}$
ResNet101-AN	45.00M	8.12G	**18.85**	**4.63**
AOGNet12M-BN	12.26M	2.19G	$21.63_{\downarrow(1.06)}$	$5.60_{\downarrow(0.22)}$
AOGNet12M-AN	12.37M	2.19G	**20.57**	**5.38**
AOGNet40M-BN	40.15M	7.51G	$18.70_{\downarrow(0.57)}$	$4.47_{\downarrow(0.21)}$
AOGNet40M-AN	40.39M	7.51G	**18.13**	**4.26**

Table 2: Comparisons between BN and our AN (w/ BN) in terms of the top-1 and top-5 error rates (%) in the ImageNet-1000 validation set using *the vanilla setup* and *the state-of-the-art setup*. † means the model is not trained by us. All other models are trained from scratch under the same settings.

ii) The State-of-the-Art Setup. There are different aspects in the vanilla setup which have better variants developed with better performance shown [11]. *We want to address whether the improvement by our proposed AN are truly fundamental or will disappear with more advanced tips and tricks added in training ConvNets.* First, on top of the basic data augmentation, we also use label smoothing [41] (with rate 0.1) and the mixup (with rate 0.2) [48]. We increase the total number of epochs to 200. We use the same stem layer with two consecutive 3×3

convolution for all models. For ResNets, we add the zero γ initialization trick, which uses 0 to initialize the last normalization layer to make the initial state of a residual block to be identity.

Results Summary. Table 2 shows the comparison results for the two setups respectively. **Our proposed AN consistently obtains the best top-1 and top-5 accuracy results with more than 0.5% absolute top-1 accuracy increase (up to 2.7%) in all models without bells and whistles.** *The improvement is often obtained with negligible extra parameters* (e.g., 0.06M parameter increase in MobileNetV2 for 2.02% absolute top-1 accuracy increase, and 0.2M parameter increase in ResNet50 with 1.42% absolute top-1 accuracy increase) *at almost no extra computational cost* (up to the precision used in measuring FLOPs). With ResNet50, our AN also outperforms GN [47] and SN [28] by 1.93% and 0.83% in top-1 accuracy respectively. For GN, it is known that it works (slightly) worse than BN under the normal (big) mini-batch setting [47]. For SN, our result shows that it is more beneficial to improve the re-calibration component than to learn-to-switch between different feature normalization schema. We observe that the proposed AN is more effective for small ConvNets in terms of performance gain. Intuitively, this makes sense. Small ConvNets usually learn less expressive features. With the mixture of affine transformations and the instance-specific channel-wise feature re-calibration, the proposed AN offers the flexibility of clustering intra-class data better while separating inter-class data better in training.

Method	#Params	FLOPS	top-1	top-5
ResNet50-SE (BN_3)	28.09M	4.12G	$22.35_{\downarrow(0.76)}$	$6.09_{\downarrow(0.21)}$
ResNet50-SE (BN_2)	26.19M	4.12G	$22.10_{\downarrow(0.55)}$	$6.02_{\downarrow(0.14)}$
ResNet50-SE (All)	29.33M	4.13G	$22.13_{\downarrow(0.52)}$	$5.96_{\downarrow(0.08)}$
ResNet50-AN (w/BN_3)	26.35M	4.11G	$21.78_{\downarrow(0.19)}$	$5.98_{\downarrow(0.1)}$
ResNet50-AN (w/BN_2)	**25.76M**	**4.09G**	**21.59**	**5.88**
ResNet50-AN (All)	25.92M	4.10G	$21.85_{\downarrow(0.26)}$	$6.06_{\downarrow(0.18)}$

Table 3: Comparisons between SE and our AN (w/ BN) in terms of the top-1 and top-5 error rates (%) in the ImageNet-1000 validation set using *the vanilla setup*. By "(All)", it means SE or AN is used for all the three BNs in a bottleneck block.

Comparisons with the SE module. Our proposed AN provides a strong alternative to the widely used SE module. Table 3 shows the comparisons. We observe that applying SE after the second BN in the bottleneck in ResNet50 is also beneficial with better performance and smaller number of extra parameters.

4.3 Object Detection and Segmentation in COCO

In object detection and segmentation, high-resolution input images are beneficial and often entailed for detecting medium to small objects, but limit the batch-size in training (often 1 or 2 images per GPU). GN [47] and SN [28] have shown significant progress in handling the applicability discrepancies of feature normalization schema from ImageNet to MS-COCO. We test our AN in MS-COCO following the standard protocol, as done in GN [47]. We build on the MMDetection code platform [4]. We observe further performance improvement.

We first summarize the details of implementation. Following the terminologies used in MMDetection [4], there are four modular components in the R-CNN detection framework [7,34,8]: *i) Feature Backbones*. We use the pre-trained net-

Architecture	Backbone	Head	#Params	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{m}	AP^{m}_{50}	AP^{m}_{75}
MobileNetV2	$\mathbb{BN}$	-	22.72M	$34.2_{\downarrow(1.8)}$	$54.6_{\downarrow(2.4)}$	$37.1_{\downarrow(1.8)}$	$30.9_{\downarrow(1.6)}$	$51.1_{\downarrow(2.7)}$	$32.6_{\downarrow(1.9)}$
	AN (w/ BN)	-	22.78M	**36.0**	**57.0**	**38.9**	**32.5**	**53.8**	**34.5**
ResNet50	$\mathbb{BN}$	-	45.71M	$39.2_{\downarrow(1.6)}$	$60.0_{\downarrow(2.1)}$	$43.1_{\downarrow(1.4)}$	$35.2_{\downarrow(1.2)}$	$56.7_{\downarrow(2.2)}$	$37.6_{\downarrow(1.1)}$
	$\mathbb{BN} + SE(BN_3)$	-	48.23M	$40.1_{\downarrow(0.7)}$	$61.2_{\downarrow(0.9)}$	$43.8_{\downarrow(0.7)}$	$35.9_{\downarrow(0.5)}$	$57.9_{\downarrow(1.0)}$	$38.1_{\downarrow(0.6)}$
	$\mathbb{BN} + SE(BN_2)$	-	46.34M	$40.1_{\downarrow(0.7)}$	$61.2_{\downarrow(0.9)}$	$43.8_{\downarrow(0.7)}$	$35.9_{\downarrow(0.5)}$	$57.9_{\downarrow(1.0)}$	$38.4_{\downarrow(0.3)}$
	AN (w/ BN)	-	45.91M	**40.8**	**62.1**	**44.5**	**36.4**	**58.9**	**38.7**
	†GN	GN [47]	45.72M	$40.3_{\downarrow(1.3)}$	$61.0_{\downarrow(1.0)}$	$44.0_{\downarrow(1.7)}$	$35.7_{\downarrow(1.7)}$	$57.9_{\downarrow(1.6)}$	$37.7_{\downarrow(2.2)}$
	†SN	SN [28]	-	$41.0_{\downarrow(0.6)}$	$\mathbf{62.3}_{\downarrow(-0.3)}$	$45.1_{\downarrow(0.6)}$	$36.5_{\downarrow(0.9)}$	$58.9_{\downarrow(0.6)}$	$38.7_{\downarrow(1.2)}$
	AN (w/ BN)	AN (w/ GN)	45.96M	**41.6**	62.0	**45.7**	**37.4**	**59.5**	**39.9**
ResNet101	$\mathbb{BN}$	-	64.70M	$41.4_{\downarrow(1.7)}$	$62.0_{\downarrow(2.1)}$	$45.5_{\downarrow(1.8)}$	$36.8_{\downarrow(1.4)}$	$59.0_{\downarrow(2.0)}$	$39.1_{\downarrow(1.6)}$
	AN (w/ BN)	-	65.15M	**43.1**	**64.1**	**47.3**	**38.2**	**61.0**	**40.7**
	†GN	GN [47]	64.71M	$41.8_{\downarrow(1.4)}$	$62.5_{\downarrow(1.5)}$	$45.4_{\downarrow(1.9)}$	$36.8_{\downarrow(2.0)}$	$59.2_{\downarrow(2.1)}$	$39.0_{\downarrow(2.6)}$
	AN (w/ BN)	AN (w/ GN)	65.20M	**43.2**	**64.0**	**47.3**	**38.8**	**61.3**	**41.6**
AOGNet12M	$\mathbb{BN}$	-	33.09M	$40.7_{\downarrow(1.3)}$	$61.4_{\downarrow(1.7)}$	$44.6_{\downarrow(1.5)}$	$36.4_{\downarrow(1.4)}$	$58.4_{\downarrow(1.7)}$	$38.8_{\downarrow(1.6)}$
	AN (w/ BN)	-	33.21M	$42.0_{\downarrow(1.0)}$	$63.1_{\downarrow(1.1)}$	$46.1_{\downarrow(0.7)}$	$37.8_{\downarrow(0.9)}$	$60.1_{\downarrow(1.0)}$	$40.4_{\downarrow(1.3)}$
	AN (w/ BN)	AN (w/ GN)	33.26M	**43.0**	**64.2**	**46.8**	**38.7**	**61.1**	**41.7**
AOGNet40M	$\mathbb{BN}$	-	60.73M	$43.4_{\downarrow(0.7)}$	$64.2_{\downarrow(0.9)}$	$47.5_{\downarrow(0.7)}$	$38.5_{\downarrow(0.5)}$	$61.0_{\downarrow(1.0)}$	$41.4_{\downarrow(0.4)}$
	AN (w/ BN)	-	60.97M	$44.1_{\downarrow(0.8)}$	$65.1_{\downarrow(1.1)}$	$48.2_{\downarrow(0.9)}$	$39.0_{\downarrow(1.2)}$	$62.0_{\downarrow(1.2)}$	$41.8_{\downarrow(1.5)}$
	AN (w/ BN)	AN (w/ GN)	61.02M	**44.9**	**66.2**	**49.1**	**40.2**	**63.2**	**43.3**

Table 4: Detection and segmentation results in MS-COCO `val2017` [26]. All models use 2x lr scheduling (180k iterations). $\mathbb{BN}$ means BN is frozen in fine-tuning for object detection. † means that models are not trained by us. All other models are trained from scratch under the same settings. The numbers show sequential improvement in the two AOGNet models indicating the importance of adding our AN in the backbone and the head respectively.

works in Table 2 (with the vanilla setup) for fair comparisons in detection, since we compare with some models which are not trained by us from scratch and use the feature backbones pre-trained in a way similar to our vanilla setup and with on par top-1 accuracy. In fine-tuning a network with AN (w/ BN) pre-trained in ImageNet such as ResNet50+AN (w/ BN) in Table 2, we freeze the stem layer and the first stage as commonly done in practice. For the remaining stages, we freeze the standardization component only (the learned mixture of affine transformations and the learned running mean and standard deviation), but allow the attention weight sub-network to be fine-tuned. *ii) Neck Backbones*: We test the feature pyramid network (FPN) [25] which is widely used in practice. *iii) Head Classifiers*. We test two setups: *(a) The vanilla setup* as done in GN [47] and SN [28]. In this setup, we further have two settings: with vs without feature normalization in the bounding box head classifier. The former is denoted by "-" in Table 4, and the latter is denoted by the corresponding type of feature normalization scheme in Table 4 (*e.g.*, GN, SN and AN (w/ GN)). We experiment on using AN (w/ GN) in the bounding box head classifier and keeping GN in the mask head unchanged for simplicity. Adding AN (w/ GN) in the mask head classifier may further help improve the performance. When adding AN (w/ GN) in the bounding box head, we adopt the same design choices except for "Choice 1, $A_1(\cdot)$" (Eqn. 11) used in learning attention weights. *(b) The state-of-the-art setup* which is based on the cascade generalization of head classifiers [3] and does not include feature normalization scheme, also denoted by "-" in Table 5. *iv) RoI Operations*. We test the RoIAlign operation [8].

Architecture	Backbone	Head	#Params	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{m}	AP^{m}_{50}	AP^{m}_{75}
ResNet101	$\mathbb{BN}$	-	96.32M	$44.4_{\downarrow(1.4)}$	$62.5_{\downarrow(1.8)}$	$48.4_{\downarrow(1.4)}$	$38.2_{\downarrow(1.4)}$	$59.7_{\downarrow(2.0)}$	$41.3_{\downarrow(1.4)}$
	AN (w/ BN)	-	96.77M	**45.8**	**64.3**	**49.8**	**39.6**	**61.7**	**42.7**
AOGNet40M	$\mathbb{BN}$	-	92.35M	$45.6_{\downarrow(0.9)}$	$63.9_{\downarrow(1.1)}$	$49.7_{\downarrow(1.1)}$	$39.3_{\downarrow(0.7)}$	$61.2_{\downarrow(1.1)}$	$42.7_{\downarrow(0.4)}$
	AN (w/ BN)	-	92.58M	**46.5**	**65.0**	**50.8**	**40.0**	**62.3**	**43.1**

Table 5: Results in MS-COCO using the cascade variant [3] of Mask R-CNN.

Result Summary. The results are summarized in Table 4 and Table 5. Compared with the vanilla BN that are frozen in fine-tuning, our AN (w/ BN) improves performance by a large margin in terms of both bounding box AP and mask AP (*1.8%* & *1.6%* for MobileNetV2, *1.6%* & *1.2%* for ResNet50, *1.7%* & *1.4%* for ResNet101, *1.3%* & *1.4%* for AOGNet12M and *0.7%* & *0.5%* for AOGNet40M). It shows the advantages of the self-attention based dynamic and adaptive control of the mixture of affine transformations (although they themselves are frozen) in fine-tuning.

With the AN further integrated in the bounding box head classifier of Mask-RCNN and trained from scratch, we also obtain better performance than GN and SN. Compared with the vanilla GN [47], our AN (w/ GN) improves bounding box and mask AP by 1.3% and 1.7% for ResNet50, and 1.4% and 2.2% for ResNet101. Compared with SN [28] which outperforms the vanilla GN in ResNet50, our AN (w/ GN) is also better by 0.6% bounding box AP and 0.9% mask AP increase respectively. Slightly less improvements are observed with AOGNets. Similar in spirit to the ImageNet experiments, we want to verify whether the advantages of our AN will disappear if we use state-of-the-art designs for head classifiers of R-CNN such as the widely used cascade R-CNN [3]. Table 5 shows that similar improvements are obtained with ResNet101 and AOGNet40M.

5 Conclusion

This paper presents Attentive Normalization (AN) that aims to harness the best of feature normalization and feature attention in a single lightweight module. AN learns a mixture of affine transformations and uses the weighted sum via a self-attention module for re-calibrating standardized features in a dynamic and adaptive way. AN provides a strong alternative to the Squeeze-and-Excitation (SE) module. In experiments, AN is tested with BN and GN as the feature normalization backbones. AN is tested in both ImageNet-1000 and MS-COCO using four representative networks (ResNets, DenseNets, MobileNetsV2 and AOGNets). It consistently obtains better performance, often by a large margin, than the vanilla feature normalization schema and some state-of-the-art variants.

Acknowledgement

This work is supported in part by NSF IIS-1909644, ARO Grant W911NF1810295, NSF IIS-1822477 and NSF IUSE-2013451. The views presented in this paper are those of the authors and should not be interpreted as representing any funding agencies.

References

1. Ba, L.J., Kiros, R., Hinton, G.E.: Layer normalization. CoRR **abs/1607.06450** (2016), http://arxiv.org/abs/1607.06450 1, 3
2. Brock, A., Donahue, J., Simonyan, K.: Large scale gan training for high fidelity natural image synthesis. arXiv preprint arXiv:1809.11096 (2018) 2, 4, 10
3. Cai, Z., Vasconcelos, N.: Cascade R-CNN: delving into high quality object detection. In: 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018. pp. 6154–6162 (2018). https://doi.org/10.1109/CVPR.2018.00644, http://openaccess.thecvf.com/content_cvpr_2018/html/Cai_Cascade_R-CNN_Delving_CVPR_2018_paper.html 9, 13, 14
4. Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Xu, J., Zhang, Z., Cheng, D., Zhu, C., Cheng, T., Zhao, Q., Li, B., Lu, X., Zhu, R., Wu, Y., Dai, J., Wang, J., Shi, J., Ouyang, W., Loy, C.C., Lin, D.: MMDetection: Open mmlab detection toolbox and benchmark. arXiv preprint arXiv:1906.07155 (2019) 9, 12
5. Deecke, L., Murray, I., Bilen, H.: Mode normalization. In: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019 (2019), https://openreview.net/forum?id=HyN-M2Rctm 1, 3, 4
6. Dumoulin, V., Belghazi, I., Poole, B., Lamb, A., Arjovsky, M., Mastropietro, O., Courville, A.C.: Adversarially learned inference. CoRR **abs/1606.00704** (2016), http://arxiv.org/abs/1606.00704 2, 4
7. Girshick, R.: Fast R-CNN. In: Proceedings of the International Conference on Computer Vision (ICCV) (2015) 12
8. He, K., Gkioxari, G., Dollár, P., Girshick, R.B.: Mask R-CNN. In: IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017. pp. 2980–2988 (2017). https://doi.org/10.1109/ICCV.2017.322, https://doi.org/10.1109/ICCV.2017.322 9, 12, 13
9. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: 2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015. pp. 1026–1034 (2015). https://doi.org/10.1109/ICCV.2015.123, https://doi.org/10.1109/ICCV.2015.123 11
10. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016) 5, 9, 11
11. He, T., Zhang, Z., Zhang, H., Zhang, Z., Xie, J., Li, M.: Bag of tricks for image classification with convolutional neural networks. CoRR **abs/1812.01187** (2018), http://arxiv.org/abs/1812.01187 11
12. Howard, A., Sandler, M., Chu, G., Chen, L., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., Le, Q.V., Adam, H.: Searching for mobilenetv3. CoRR **abs/1905.02244** (2019), http://arxiv.org/abs/1905.02244 8
13. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. CoRR **abs/1709.01507** (2017), http://arxiv.org/abs/1709.01507 2, 4, 5, 7, 11
14. Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2017) 9
15. Huang, L., Liu, X., Lang, B., Yu, A.W., Wang, Y., Li, B.: Orthogonal weight normalization: Solution to optimization over multiple dependent stiefel manifolds

in deep neural networks. In: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018. pp. 3271–3278 (2018), https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17072 3
16. Huang, L., Yang, D., Lang, B., Deng, J.: Decorrelated batch normalization. In: 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018. pp. 791–800 (2018) 1, 3
17. Huang, Z., Wang, X., Huang, L., Huang, C., Wei, Y., Liu, W.: Ccnet: Criss-cross attention for semantic segmentation. CoRR **abs/1811.11721** (2018), http://arxiv.org/abs/1811.11721 2
18. Ioffe, S.: Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. In: Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA. pp. 1945–1953 (2017) 1, 3
19. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Blei, D., Bach, F. (eds.) Proceedings of the 32nd International Conference on Machine Learning (ICML-15). pp. 448–456. JMLR Workshop and Conference Proceedings (2015), http://jmlr.org/proceedings/papers/v37/ioffe15.pdf 1, 3, 8, 9
20. Jia, S., Chen, D., Chen, H.: Instance-level meta normalization. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019. pp. 4865–4873 (2019), http://openaccess.thecvf.com/content_CVPR_2019/html/Jia_Instance-Level_Meta_Normalization_CVPR_2019_paper.html 1, 2, 4
21. Kalayeh, M.M., Shah, M.: Training faster by separating modes of variation in batch-normalized models. IEEE Transactions on Pattern Analysis and Machine Intelligence pp. 1–1 (2019). https://doi.org/10.1109/TPAMI.2019.2895781 1, 3, 4
22. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. arXiv preprint arXiv:1812.04948 (2018) 2, 4
23. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Neural Information Processing Systems (NIPS). pp. 1106–1114 (2012) 5
24. Li, X., Song, X., Wu, T.: Aognets: Compositional grammatical architectures for deep learning. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019. pp. 6220–6230 (2019) 9
25. Lin, T., Dollár, P., Girshick, R.B., He, K., Hariharan, B., Belongie, S.J.: Feature pyramid networks for object detection. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017. pp. 936–944 (2017). https://doi.org/10.1109/CVPR.2017.106, https://doi.org/10.1109/CVPR.2017.106 13
26. Lin, T., Maire, M., Belongie, S.J., Bourdev, L.D., Girshick, R.B., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: common objects in context. CoRR **abs/1405.0312** (2014), http://arxiv.org/abs/1405.0312 8, 13
27. Loshchilov, I., Hutter, F.: SGDR: stochastic gradient descent with restarts. CoRR **abs/1608.03983** (2016), http://arxiv.org/abs/1608.03983 11
28. Luo, P., Ren, J., Peng, Z.: Differentiable learning-to-normalize via switchable normalization. CoRR **abs/1806.10779** (2018), http://arxiv.org/abs/1806.10779 2, 3, 9, 11, 12, 13, 14

29. Miyato, T., Koyama, M.: cgans with projection discriminator. arXiv preprint arXiv:1802.05637 (2018) 2
30. Pan, X., Zhan, X., Shi, J., Tang, X., Luo, P.: Switchable whitening for deep representation learning. In: 2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019. pp. 1863–1871. IEEE (2019). https://doi.org/10.1109/ICCV.2019.00195, https://doi.org/10.1109/ICCV.2019.00195 9
31. Park, T., Liu, M., Wang, T., Zhu, J.: Semantic image synthesis with spatially-adaptive normalization. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019. pp. 2337–2346 (2019) 2, 4
32. Peng, C., Xiao, T., Li, Z., Jiang, Y., Zhang, X., Jia, K., Yu, G., Sun, J.: Megdet: A large mini-batch object detector. In: 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018. pp. 6181–6189 (2018) 3
33. Perez, E., de Vries, H., Strub, F., Dumoulin, V., Courville, A.C.: Learning visual reasoning without strong priors. CoRR **abs/1707.03017** (2017), http://arxiv.org/abs/1707.03017 2, 4
34. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards real-time object detection with region proposal networks. In: Neural Information Processing Systems (NIPS) (2015) 12
35. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. Int. J. Comput. Vision (IJCV) **115**(3), 211–252 (2015). https://doi.org/10.1007/s11263-015-0816-y 8
36. Salimans, T., Kingma, D.P.: 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. p. 901 (2016) 3
37. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4510–4520 (2018) 9
38. Santurkar, S., Tsipras, D., Ilyas, A., Madry, A.: 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. pp. 2488–2498 (2018), http://papers.nips.cc/paper/7515-how-does-batch-normalization-help-optimization 3
39. Shao, W., Meng, T., Li, J., Zhang, R., Li, Y., Wang, X., Luo, P.: Ssn: Learning sparse switchable normalization via sparsestmax. CoRR **abs/1903.03793** (2019), http://arxiv.org/abs/1903.03793 2, 3, 4
40. Sun, W., Wu, T.: Image synthesis from reconfigurable layout and style. In: International Conference on Computer Vision, ICCV (2019) 2, 4
41. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. CoRR **abs/1512.00567** (2015), http://arxiv.org/abs/1512.00567 11
42. Ulyanov, D., Vedaldi, A., Lempitsky, V.S.: Instance normalization: The missing ingredient for fast stylization. CoRR **abs/1607.08022** (2016), http://arxiv.org/abs/1607.08022 1, 3
43. de Vries, H., Strub, F., Mary, J., Larochelle, H., Pietquin, O., Courville, A.C.: 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. pp. 6597–6607 (2017), http://papers.nips.cc/paper/7237-modulating-early-visual-processing-by-language 2, 4

44. Wang, F., Jiang, M., Qian, C., Yang, S., Li, C., Zhang, H., Wang, X., Tang, X.: 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. pp. 6450–6458 (2017). https://doi.org/10.1109/CVPR.2017.683, https://doi.org/10.1109/CVPR.2017.683 4
45. Wang, X., Girshick, R.B., Gupta, A., He, K.: 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. pp. 7794–7803 (2018). https://doi.org/10.1109/CVPR.2018.00813, http://openaccess.thecvf.com/content_cvpr_2018/html/Wang_Non-Local_Neural_Networks_CVPR_2018_paper.html 2
46. Woo, S., Park, J., Lee, J., Kweon, I.S.: CBAM: convolutional block attention module. In: Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part VII. pp. 3–19 (2018). https://doi.org/10.1007/978-3-030-01234-2_1, https://doi.org/10.1007/978-3-030-01234-2_1 4
47. Wu, Y., He, K.: Group normalization. In: Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XIII. pp. 3–19 (2018). https://doi.org/10.1007/978-3-030-01261-8_1, https://doi.org/10.1007/978-3-030-01261-8_1 1, 3, 7, 9, 11, 12, 13, 14
48. Zhang, H., Cissé, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. In: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings (2018), https://openreview.net/forum?id=r1Ddp1-Rb 11