On Feature Normalization and Data Augmentation

Boyi Li[*12] Felix Wu[*3] Ser-Nam Lim[4] Serge Belongie[12] Kilian Q. Weinberger[13]
[1]Cornell University [2]Cornell Tech [3]ASAPP [4]Facebook AI
{bl728,sjb344,kilian}@cornell.edu fwu@asapp.com sernamlim@fb.com

Abstract

The moments (a.k.a., mean and standard deviation) of latent features are often removed as noise when training image recognition models, to increase stability and reduce training time. However, in the field of image generation, the moments play a much more central role. Studies have shown that the moments extracted from instance normalization and positional normalization can roughly capture style and shape information of an image. Instead of being discarded, these moments are instrumental to the generation process. In this paper we propose Moment Exchange, an implicit data augmentation method that encourages the model to utilize the moment information also for recognition models. Specifically, we replace the moments of the learned features of one training image by those of another, and also interpolate the target labels—forcing the model to extract training signal from the moments in addition to the normalized features. As our approach is fast, operates entirely in feature space, and mixes different signals than prior methods, one can effectively combine it with existing augmentation approaches. We demonstrate its efficacy across several recognition benchmark data sets where it improves the generalization capability of highly competitive baseline networks with remarkable consistency.

1. Introduction

Image recognition and image generation are two corner stones of computer vision. While both are burgeoning fields, specialized techniques from both sub-areas can sometimes form a dichotomy. Examples are mixup [69] and squeeze-and-excitation [19] from the former, and adaptive instance normalization [22] from the latter, although exceptions exist. Historically, the field of deep learning was widely popularized in discriminative image classification with AlexNet [32], and image generation through GANs [12] and VAEs [30].

One particular aspect of this dichotomy is that in image recognition, popularized by batch normalization [24], the first and second moments (a.k.a., mean and standard deviation) in image recognition are computed across instances in a mini-batch and typically removed as noise [24, 60]. Studies have shown that this smoothes the optimization landscape [46] and enables larger learning rates [2], which leads to faster convergence in practice. In contrast, for techniques like instance normalization [55] and positional normalization [34], moments play a central role in the image generation process. For example, exchanging moments of latent features across samples has become a popular way to control for the style or shape of generated images [22, 23, 27, 34, 41]. Here, moments are viewed as features, not noise, with research showing that they encode the style of an image [22, 27], as well as the underlying structure [34]. To substantiate this point, we depict the first and second moments of the features extracted in the first layer of a ResNet [16] in Fig. 1 , using the technique described in [34]. The class label can still be inferred visually from both moments, which is a testament to the signal that remains in these statistics. To further substantiate our observation, we also show in Fig. 2 that simply using moments (from the first ResNet layer) for image classification already yields non-trivial performance (red bar) as compared to random guessing (gray bar). Similarly, removing the moments from positional normalization has a detrimental effect (blue bar vs. green bar).

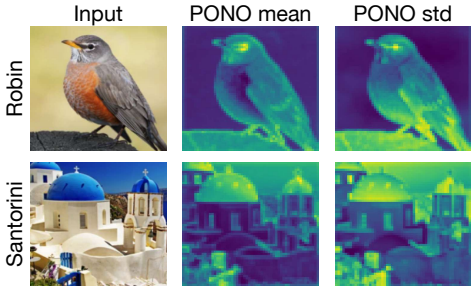

Figure 1. PONO mean and std captures structural information.

As there is evidently important signal in the moments and in the normalized features, we would like to introduce a

∗: Equal contribution.

Example images are from Shutterstock.

way to *regulate* how much attention the deep net should pay to each source. One approach to direct neural networks to a particular signal source is to introduce dedicated feature augmentation. For example, it has been shown that ConvNets trained on ImageNet [6] are biased towards textures instead of shapes [10]. To overcome this, Geirhos et al. [10] introduce a style transfer model to create a set of images with unreal textures. For example, they generate cats with elephant skins and bears with Coca-Cola bottle texture. An image classifier is trained to recognize the shape (cats or bears) instead of the textures (elephants or bottles).

In this paper we propose a novel data augmentation scheme that, to our knowledge, is the first method to systematically regulate how much attention a network pays to the signal in the feature moments. Concretely, we extract the mean and variance (across channels) after the first layer, but instead of simply removing them, we swap them between images. See Fig. 3 for a schematic illustration, where we extract and remove the feature moments of a cat image, and inject the moments of a plane. Knowing that the resulting features now contain information about both images, we make the network predict an interpolation of the two labels. In the process, we force the network to pay attention to two aspects of the data: the normalized feature (from the cat) and the moments (from the plane). By basing its prediction on two different signals we increase the robustness of the classification, as during testing both would point towards the same label. We call our method Moment Exchange or MoEx for short.

Through exchanging the moments, we swap the shape (or style) information of two images, which can be viewed as an implicit version of the aforementioned method proposed by Geirhos et al. [10]. However, MoEx does not require a pre-trained style transfer model to create a dataset explicitly. In fact, MoEx is very effective for training with mini-batches and can be implemented in a few lines of code: During training we compute the feature mean and variance for each instance at a given layer (acros channels), permute them across the mini-batch, and re-inject them into the feature representation of other instances (while interpolating the labels).

MoEx operates purely in feature space and can therefore easily be applied jointly with existing data augmentation methods that operate in the input space, such as cropping, flipping, rotating, but even label-perturbing approaches like Mixup [69] or Cutmix [67]. Importantly, because MoEx only alters the first and second moments of the pixel distributions, it has an orthogonal effect to existing data augmentation methods and its improvements can be "stacked" on top of their established gains in generalization. We conduct extensive experiments on eleven different tasks/-datasets using more than ten varieties of models. The results show that MoEx consistently leads to significant improvements across models and tasks, and it is particularly well suited to be combined with existing augmentation approaches. Further, our experiments show that MoEx is not limited to computer vision, but is also readily applicable and highly effective in applications within speech recognition and natural language processing—suggesting that MoEx reveals a fundamental insight about deep nets that crosses areas and data types. Our implementation is available at https://github.com/Boyiliee/MoEx.

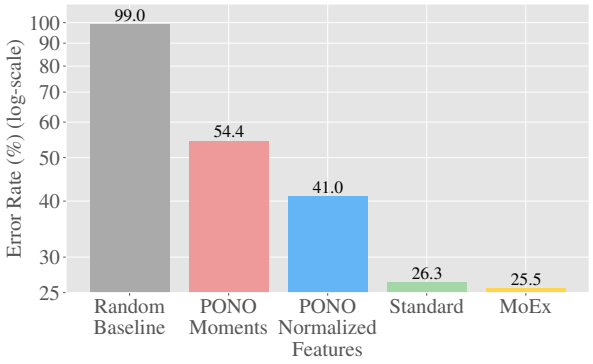

Figure 2. Error rates of ResNet-110 using different features on CIFAR-100. The numbers are averaged over three random runs. We compute the moments after the first convolutional layer, and either adds a PONO layer right after the first Conv-BN-ReLU block or computes the first two moments and concatenate them as a two-channel feature map. A ResNet which can only see the moments can still make nontrivial predictions (Red is much better than gray). Additionally, using only the normalized feature (i.e., removing the PONO moments) hurts the performance (Blue is worse than green), which also shows that these moments contains important information. Finally, MoEx improves the performance by encouraging the model to use both sources of signal during training.

2. Background and Related Work

Feature normalization has always been a prominent part of neural network training [33, 35]. Initially, when networks had predominately one or two hidden layers, the practice of z-scoring the features was limited to the input itself. As networks became deeper, Ioffe and Szegedy [24] extended the practice to the intermediate layers with the celebrated BatchNorm algorithm. As long as the mean and variance are computed across the entire input, or a randomly picked mini-batch (as it is the case for BatchNorm), the extracted moments reveal biases in the data set with no predictive information — removing them causes no harm but can substantially improve optimization and generalization [2, 33, 45].

In contrast, recently proposed normalization methods [1, 34, 55, 60] treat the features of each training instance as a distribution and normalize them for each sample individually. We refer to the extracted mean and variance as *intra-*

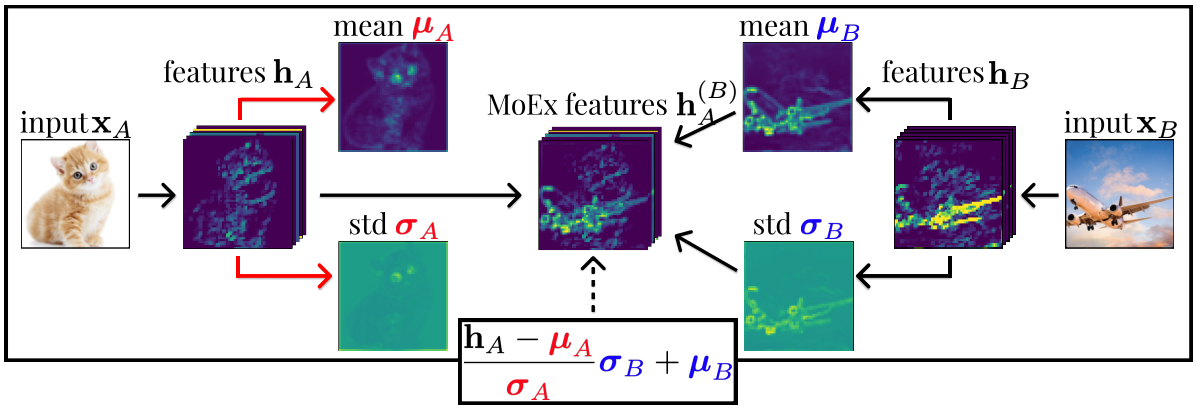

Figure 3. MoEx with PONO normalization. The features $\mathbf{h}_A$ of the cat image are normalized and then infused with moments $\boldsymbol{\mu}_B, \boldsymbol{\sigma}_B$ from the plane image. See Appendix for more examples.

instance moments. We argue that intra-instance moments are attributes of a data instance that describe the distribution of its features and should not be discarded. Recent works [22, 34] have shown that such attributes can be useful in several generative models. Realizing that these moments capture interesting information about data instances, we propose to use them for data augmentation.

Data augmentation has a similarly long and rich history in machine learning. Initial approaches discovered the concept of *label-preserving* transformations [47, 49] to mimic larger training data sets to suppress overfitting effects and improve generalization. For instance, Simard et al. [50] randomly translate or rotate images assuming that the labels of the images would not change under such small perturbations. Many subsequent papers proposed alternative flavors of this augmentation approach based on similar insights [4, 5, 7, 27, 28, 36, 51, 63, 70]. Beyond vision tasks, back-translation [3, 8, 48, 66] and word dropout [25] are commonly used to augment text data. Besides augmenting inputs, Ghiasi et al. [11], van der Maaten et al. [56], Wang et al. [58] adjust either the features or loss function as implicit data augmentation methods. In addition to label-preserving transformations, there is an increasing trend to use *label-perturbing* data augmentation methods. Zhang et al. [69] arguably pioneered the field with Mixup, which interpolates two training inputs in feature and label space simultaneously. Cutmix [67], instead, is designed especially for image inputs. It randomly crops a rectangular region of an image and pastes it into another image, mixing the labels proportional to the number of pixels contributed by each input image to the final composition.

3. Moment Exchange

In this section we introduce Moment Exchange (MoEx), which blends feature normalization with data augmentation. Similar to Mixup and Cutmix, it fuses features and labels across two training samples, however it is unique in its asymmetry, as it mixes two very different components: The normalized features of one instance are combined with the feature moments of another. This asymmetric composition in feature space allows us to capture and smooth out different directions of the decision boundary, not previously covered by existing augmentation approaches. We also show that MoEx can be implemented very efficiently in a few lines of code, and should be regarded as an effective default companion to existing data augmentation methods.

Setup. Deep neural networks are composed of layers of transformations including convolution, pooling, transformers [57], fully connected layers, and non-linear activation layers. Consider a batch of input instances $\mathbf{x}$, these transformations are applied sequentially to generate a series of hidden features $\mathbf{h}^1, ..., \mathbf{h}^L$ before passing the final feature $\mathbf{h}^L$ to a linear classifier. For each instance, any feature presentation $\mathbf{h}^\ell$ is a 3D tensor indexed by channel (C), height (H), and width (W).

Normalization. We assume the network is using an invertible intra-instance normalization. We denote this function by F, which takes the features $\mathbf{h}_i^\ell$ of the i-th input $\mathbf{x}_i$ at layer ℓ and produces three outputs: the normalized features $\hat{\mathbf{h}}_i$, the first moment $\boldsymbol{\mu}_i$, and the second moment $\boldsymbol{\sigma}_i$:

$$(\hat{\mathbf{h}}_i^\ell, \boldsymbol{\mu}_i^\ell, \boldsymbol{\sigma}_i^\ell) = F(\mathbf{h}_i^\ell), \;\; \mathbf{h}_i^\ell = F^{-1}(\hat{\mathbf{h}}_i^\ell, \boldsymbol{\mu}_i^\ell, \boldsymbol{\sigma}_i^\ell).$$

The inverse function F^{-1} reverses the normalization process. As an example, PONO [34] computes the first and second moments across channels from the feature representation at a given layer

$$\boldsymbol{\mu}_{b,h,w}^\ell = \frac{1}{C}\sum_c \mathbf{h}_{b,c,h,w}^\ell,$$

$$\boldsymbol{\sigma}_{b,h,w}^\ell = \sqrt{\frac{1}{C}\sum_c \left(\mathbf{h}_{b,c,h,w}^\ell - \boldsymbol{\mu}_{b,h,w}^\ell\right)^2 + \epsilon}.$$

The normalized features have zero-mean and standard deviation 1 along the channel dimension. Note that after using MoEx with an *intra-instance* normalization to exchange

features, we can still apply an *inter-instance* normalization (like BatchNorm) on these exchanged or mixed features, with their well-known beneficial impact on convergence. As the norms compute statistics across different dimensions their interference is insignificant.

Moment Exchange. The procedure described in the following functions identically for each layer it is applied to and we therefore drop the ℓ superscript for notational simplicity. Further, for now, we only consider two randomly chosen samples $\mathbf{x}_A$ and $\mathbf{x}_B$ (see Fig. 3 for a schematic illustration). The intra-instance normalization decomposes the features of input $\mathbf{x}_A$ at layer ℓ into three parts, $\hat{\mathbf{h}}_A, \boldsymbol{\mu}_A, \boldsymbol{\sigma}_A$. Traditionally, batch-normalization [24] discards the two moments and only proceeds with the normalized features $\hat{\mathbf{h}}_A$. If the moments are computed across instances (e.g. over the mini-batch) this makes sense, as they capture biases that are independent of the label. However, in our case we focus on intra-instance normalization (See Fig. 1), and therefore both moments are computed only from $\mathbf{x}_A$ and are thus likely to contain label-relevant signal. This is clearly visible in the *cat* and *plane* examples in Fig. 3. All four moments ($\boldsymbol{\mu}_A$,$\boldsymbol{\sigma}_A$,$\boldsymbol{\mu}_B$,$\boldsymbol{\sigma}_B$), capture the underlying structure of the samples, revealing their respective class labels.

We consider the normalized features and the moments as distinct views of the same instance. It generally helps robustness if a machine learning algorithm leverages multiple sources of signal, as it becomes more resilient in case one of them is under-expressed in a test example. For instance, the first moment conveys primarily structural information and only little color information, which, in the case of cat images can help overcome overfitting towards fur color biases in the training data set.

In order to encourage the network to utilize the moments, we use the two images and combine them by injecting the moments of image $\mathbf{x}_B$ into the feature representation of image $\mathbf{x}_A$: $\mathbf{h}_A^{(B)} = F^{-1}(\hat{\mathbf{h}}_A, \boldsymbol{\mu}_B, \boldsymbol{\sigma}_B)$. In the case of PONO, the transformation becomes $\mathbf{h}_A^{(B)} = \boldsymbol{\sigma}_B \frac{\mathbf{h}_A - \boldsymbol{\mu}_A}{\boldsymbol{\sigma}_A} + \boldsymbol{\mu}_B$. We now proceed with these features $\mathbf{h}_A^{(B)}$, which contain the moments of image B (plane) hidden inside the features of image A (cat). In order to encourage the neural network to pay attention to the injected features of B we modify the loss function to predict the class label y_A and also y_B, up to some mixing constant $\lambda \in [0, 1]$. The loss becomes a straight-forward combination

$$\lambda \cdot \ell(\mathbf{h}_A^{(B)}, y_A) + (1 - \lambda) \cdot \ell(\mathbf{h}_A^{(B)}, y_B).$$

Implementation. In practice one needs to apply MoEx only on a single layer in the neural network, as the fused signal is propagated until the end. With PONO as the normalization method, we observe that the first layer ($\ell = 1$) usually leads to the best result. In contrast, we find that MoEx is more suited for later layers when using IN [55], GN [60], or LN [1] for moment extraction. Please see Sec. 5 for a detailed ablation study. The inherent randomness of mini-batches allows us to implement MoEx very efficiently. For each input instance in the mini-batch $\mathbf{x}_i$ we compute the normalized features and moments $\hat{\mathbf{h}}_i, \boldsymbol{\mu}_i, \boldsymbol{\sigma}_i$. Subsequently we sample a random permutation π and apply MoEx with a random pair within the mini-batch $\mathbf{h}_i^{(\pi(i))} \leftarrow F^{-1}(\hat{\mathbf{h}}_i, \boldsymbol{\mu}_{\pi(i)}, \boldsymbol{\sigma}_{\pi(i)})$. See Algorithm 1 in the Appendix for an example implementation in PyTorch [42]. Note that all computations are extremely fast and only introduce negligible overhead during training.

Model	#param.	CIFAR10	CIFAR100
ResNet-110 (3-stage)	1.7M	6.82±0.23	26.28±0.10
+MoEx	1.7M	**6.03±0.24**	**25.47±0.09**
DenseNet-BC-100 (k=12)	0.8M	4.67±0.10	22.61±0.17
+MoEx	0.8M	**4.58±0.03**	**21.38±0.18**
ResNeXt-29 (8×64d)	34.4M	4.00±0.04	18.54±0.27
+MoEx	34.4M	**3.64±0.07**	**17.08±0.12**
WRN-28-10	36.5M	3.85±0.06	18.67±0.07
+MoEx	36.5M	**3.31±0.03**	**17.69±0.10**
DenseNet-BC-190 (k=40)	25.6M	3.31±0.04	17.10±0.02
+MoEx	25.6M	**2.87±0.03**	**16.09±0.14**
PyramidNet-200 ($\tilde{\alpha} = 240$)	26.8M	3.65±0.10	16.51±0.05
+MoEx	26.8M	**3.44±0.03**	**15.50±0.27**

Table 1. Classification results (Err (%)) on CIFAR-10, CIFAR-100 in comparison with various competitive baseline models. See text for details.

Hyper-parameters. To control the intensity of our data augmentation, we perform MoEx during training with some probability p. In this way, the model can still see the original features with probability $1 - p$. In practice we found that $p = 0.5$ works well on most datasets except that we set $p = 1$ for ImageNet where we need stronger data augmentation. The interpolation weight λ is another hyper-parameter to be tuned. Empirically, we find that 0.9 works well across data sets. The reason can be that the moments contain less information than the normalized features. Please see Appendix for a detailed ablation study.

Properties. MoEx is performed entirely at the feature level inside the neural network and can be readily combined with other augmentation methods that operate on the raw input (pixels or words). For instance, Cutmix [67] typically works best when applied on the input pixels directly. We find that the improvements of MoEx are complimentary to such prior work and recommend to use MoEx in combination with established data augmentation methods.

4. Experiments

We evaluate the efficacy of MoEx thoroughly across several tasks and data modalities.

4.1. Image Classification on CIFAR

CIFAR-10 and CIFAR-100 [31] are benchmark datasets containing 50K training and 10K test colored images at

PyramidNet-200 ($\tilde{\alpha} = 240$) (# params: 26.8 M)	Top-1 / Top-5 Error (%)
Baseline	16.45 / 3.69
Manifold Mixup [69]	16.14 / 4.07
StochDepth [21]	15.86 / 3.33
DropBlock [11]	15.73 / 3.26
Mixup [69]	15.63 / 3.99
ShakeDrop [65]	15.08 / **2.72**
MoEx	**15.02** / 2.96
Cutout [7]	16.53 / 3.65
Cutout + MoEx	**15.11** / **3.23**
CutMix [67]	14.47 / 2.97
CutMix + MoEx	**13.95** / **2.95**
CutMix + ShakeDrop [65]	13.81 / 2.29
CutMix + ShakeDrop + MoEx	**13.47** / **2.15**

Table 2. Combining MoEx with other regularization methods on CIFAR-100 following the setting of [67]. The best numbers in each group are **bold**.

32x32 resolution. We evaluate our method using various model architectures [15, 16, 20, 64, 68] on CIFAR-10 and CIFAR-100. We follow the conventional setting with random translation as the default data augmentation and apply MoEx to the features after the first layer. Furthermore, to justify the compatibility of MoEx with other regularization methods, we follow the official setup of [67] and apply MoEx jointly with several regularization methods to PyramidNet-200 [15] on CIFAR-100.

Table 1 displays the classification results on CIFAR-10 and CIFAR-100 with and without MoEx. We report mean and standard error over three runs [14]. MoEx consistently enhances the performance of all the baseline models. Table 2 demonstrates the CIFAR-100 classification results on the basis of PyramidNet-200. Compared to other augmentation methods, PyramidNet trained with MoEx obtains the lowest error rates in all-but one settings. However, significant additional improvements are achieved when MoEx is combined with existing methods — setting a new state-of-the-art for this particular benchmark task when combined with the two best performing alternatives, CutMix and ShakeDrop.

4.2. Image Classification on ImageNet

We evaluate on ImageNet [6] (ILSVRC 2012 version), which consists of 1.3M training images and 50K validation images of various resolutions. For faster convergence, we use NVIDIA's mixed-precision training code base with batch size 1024, default learning rate $0.1 \times \text{batch\_size}/256$, cosine annealing learning rate scheduler [40] with linear warmup [13] for the first 5 epochs. As the model might require more training updates to converge with data augmentation, we apply MoEx to ResNet-50, ResNeXt-50 (32×4d), DenseNet-265 and train them for 90 and 300 epochs. For a fair comparison, we also report Cutmix [67] under the same setting. Since the test server of ImageNet is no longer available to the public, we follow the common practice [11, 20, 64, 67, 69] reporting the validation scores of the baselines as well as our method.

https://github.com/bearpaw/pytorch-classification
https://github.com/clovaai/CutMix-PyTorch
https://github.com/NVIDIA/apex/tree/master/examples/imagenet

Model	# of epochs	Test Error (%)	
		Baseline	+MoEx
ResNet-50	90	23.6	**23.1**
ResNeXt-50 (32×4d)	90	22.2	**21.4**
DenseNet-265	90	21.9	**21.6**
ResNet-50	300	23.1	**21.9**
ResNeXt-50 (32×4d)	300	22.5	**22.0**
DenseNet-265	300	21.5	**20.9**

Table 3. Classification results (Test Err (%)) on ImageNet in comparison with various models. Note: The ResNeXt-50 (32×4d) models trained for 300 epochs overfit. They have higher training accuracy but lower test accuracy than the 90-epoch ones.

ResNet50 (# params: 25.6M)	# of epochs	Top-1 / Top-5 Error (%)
ISDA [58]	90	23.3 / 6.8
Shape-ResNet [10]	105	23.3 / 6.7
Mixup [69]	200	22.1 / 6.1
AutoAugment [4]	270	22.4 / 6.2
Fast AutoAugment [37]	270	22.4 / 6.3
DropBlock [11]	270	21.9 / 6.0
Cutout [7]	300	22.9 / 6.7
Manifold Mixup [69]	300	22.5 / 6.2
Stochastic Depth [21]	300	22.5 / 6.3
CutMix [67]	300	21.4 / 5.9
Baseline	300	23.1 / 6.6
MoEx	300	21.9 / 6.1
CutMix	300	21.3 / **5.7**
CutMix + MoEx	300	**20.9** / **5.7**

Table 4. Comparison of regularization and augmentation methods on ImageNet. Stochastic Depth and Cutout results are from [67].

Table 3 shows the test error rates on the ImageNet data set. MoEx is able to improve the classification performance throughout, regardless of model architecture. Similar to the previous CIFAR experiments, we observe in Table 4 that MoEx is highly competitive when compared to existing regularization methods and truly shines when it is combined with them. When applied jointly with CutMix (the strongest alternative), we obtain our lowest Top-1 and Top-5 error of 20.9/5.7 respectively. Beyond, we apply MoEx to EfficientNet-B0 [54] and follow the official train-

ing scheme, we find MoEx is able to help reduce the error rate of baseline from 22.9 to 22.3.

4.3. Fintuneing Imagenet pretrained models on Pascal VOC for Object Detection

To demonstrate that MoEx encourages models to learn better image representations, we apply models pre-trained on ImageNet with MoEx to downstream tasks including object detection on Pascal VOC 2007 dataset. We use the Faster R-CNN [44] with C4 or FPN [38] backbones implemented in Detectron2 [61] and following their default training configurations. We consider three ImageNet pretrained models: the ResNet-50 provided by He et al. [16], our ResNet-50 baseline trained for 300 epochs, our ResNet-50 trained with CutMix [67], and our ResNet-50 trained with MoEx. A Faster R-CNN is initialized with these pretrained weights and finetuned on Pascal VOC 2007 + 2012 training data, tested on Pascal VOC 2007 test set, and evaluated with the PASCAL VOC style metric: average precision at IoU 50% which we call AP_{VOC} (or AP50 in detectron2). We also report MS COCO [39] style average precision metric AP_{COCO} which is recently considered as a better choice. Notably, MoEx is not applied during finetuning.

Table 5 shows the average precision of different initializations. We discover that MoEx provides a better initialization than the baseline ResNet-50 and is competitive against CutMix[67] for the downstream cases and leads slightly better performance regardless of backbone architectures.

Backbone	Initialization	AP_{VOC}	AP_{COCO}
C4	ResNet-50 (default)	80.3	51.8
	ResNet-50 (300 epochs)	81.2	53.5
	ResNet-50 + CutMix	**82.1**	54.3
	ResNet-50 + MoEx	81.6	**54.6**
FPN	ResNet-50 (default)	81.8	53.8
	ResNet-50 (300 epochs)	82.0	54.2
	ResNet-50 + CutMix	82.1	**54.3**
	ResNet-50 + MoEx	**82.3**	**54.3**

Table 5. Object detection on PASCAL VOC 2007 test set using Faster R-CNN whose backbone is initialized with different pretrained weights. We use either the original C4 or feature pyramid network [38] backbone.

4.4. 3D model classification on ModelNet

We conduct experiments on Princeton ModelNet10 and ModelNet40 datasets [62] for 3D model classification. This task aims to classify 3D models encoded as 3D point clouds into 10 or 40 categories. As a proof of concept, we use PointNet++ (SSG) [43] implemented efficiently in PyTorch Geometric [9] as the baseline. It does not use surface normal as additional inputs. We apply MoEx to the features after the first set abstraction layer in PointNet++. Following their default setting, all models are trained with ADAM [29] at batch size 32 for 200 epochs. The learning rate is set to 0.001. We tune the hyper-parameters of MoEx on ModelNet-10 and apply the same hyper-parameters to ModelNet-40. We choose $p = 0.5$, $\lambda = 0.9$, and InstanceNorm for this task, which leads to slightly better results.

Table 6 summarizes the results out of three runs, showing mean error rates with standard errors. MoEx reduces the classification errors from 6.0% to 5.3% and 9.2% to 8.8% on ModelNet10 and ModelNet40, respectively.

Model	ModelNet10	ModelNet40
PointNet++	6.02±0.10	9.16±0.16
+ MoEx	**5.25±0.18**	**8.78±0.28**

Table 6. Classification errors (%) on ModelNet10 and ModelNet40. The mean and standard error out of 3 runs are reported.

5. Ablation Study

5.1. MoEx Design Choices

In the previous section we have established that MoEx yields significant improvements across many tasks and model architectures. In this section we shed light onto which design choices crucially contribute to these improvements. Table 7 shows results on CIFAR-100 with a ResNet-110 architecture, averaged over 3 runs. The column titled MoEx indicates if we performed moment exchange or not.

Label smoothing. First, we investigate if the positive effect of MoEx can be attributed to label smoothing [53]. In label smoothing, one changes the loss of a sample $\mathbf{x}$ with label y to $\lambda\ell(\mathbf{x}, y) + \frac{1}{C-1}\sum_{y' \neq y}(1-\lambda)\ell(\mathbf{x}, y')$, where C denotes the total number of classes. Essentially the neural network is not trained to predict one class with 100% certainty, but instead only up to a confidence λ.

Further, we evaluate **Label Interpolation only.** Here, we evaluate MoEx with label interpolation - but without any feature augmentation, essentially investigating the effect of label interpolation alone. Both variations yield some improvements over the baseline, but are significantly worse than MoEx.

Interpolated targets. The last three rows of Table 7 demonstrate the necessity of utilizing the moments for prediction. We investigate two variants: $\lambda = 1$, which corresponds to no label interpolation; MoEx with label smoothing (essentially assigning a small loss to all labels except y_A). The last row corresponds to our proposed method, MoEx ($\lambda = 0.9$).

Two general observations can be made: 1) interpolating the labels is crucial for MoEx to be beneficial — the

https://github.com/rusty1s/pytorch_geometric

We do hyper-parameter search from $p \in \{0.5, 1\}$, $\lambda \in \{0.5, 0.9\}$ and whether to use PONO or InstanceNorm.

name	MoEx	Test Error
Baseline	✗	26.3±0.10
Label smoothing [53]	✗	26.0±0.06
Label Interpolation only	✗	26.0±0.12
MoEx (λ = 1, not interpolating the labels)	✓	26.3±0.02
MoEx with label smoothing	✓	25.8±0.09
MoEx (λ = 0.9, label interpolation, proposed)	✓	**25.5±0.09**

Table 7. Ablation study on different design choices.

approach leads to absolutely no improvement when we set $\lambda = 1$. 2) it is also important to perform moment exchange, without it MoEx reduces to a version of label smoothing, which yields significantly smaller benefits.

Choices of normalization methods. We study how MoEx performs when using moments from LayerNorm (LN) [1], InstanceNorm (IN) [55], PONO [34], GroupNorm (GN) [60], and local response normalization (LRN) [32] perform. For LRN, we use a recent variant [26] which uses the unnormalized 2nd moment at each position. We conduct experiments on CIFAR-100 with ResNet110. For each normalization, we do a hyper-parameter sweep to find the best setup. Table 8 shows classification results of MoEx with various feature normalization methods on CIFAR-100 averaged over 3 runs (with corresponding standard errors). We observe that MoEx generally works with all normalization approaches, however PONO has a slight but significant edge, which we attribute to the fact that it catches the structural information of the feature most effectively. Table 9 shows that different normalization methods work the best at different layers. With PONO or GN, we apply MoEx in the first layer (right before the first stage), whereas the LN moments work best when exchanged before the third stage of a 3-stage ResNet-110; IN is better to be applied right before the second stage. We hypothesize the reason is that PONO moments captures local information while LN and IN compute global features which are better encoded at later stages of a ResNet. For image classification, using PONO seems generally best. While MoEx with other normalization methods in different stages could also obtain competitive results such as LN before Stage 3 in Table 8. We assume it is because in the early layers it is important to exchange the whole mode (and PONO has a significant advantage), whereas for the last stage the scale already contains a lot of information (LN performs best here), which is worth being studied in other architecture such as Transformer [57], etc. Beyond, for some other tasks we observe that using moments from IN can be more favorable (See Subsec. 4.4).

We select the best result from experiments with $\lambda \in \{0.6, 0.7, 0.8, 0.9\}$ and $p \in \{0.25, 0.5, 0.75, 1.0\}$. We choose the best layer among the 1st layer, 1st stage, 2nd stage, and 3rd stage. For each setting, we obtain the mean and standard error out of 3 runs with different random seeds.

Moments to exchange	Test Error
No MoEx	26.3±0.10
All features in a layer, i.e. LN	25.6±0.02
Feature in each channel, i.e. IN	25.7±0.13
Features in Group of channels, i.e. GN (g=4)	25.7±0.09
Features at each position, i.e. PONO	**25.5±0.09**
1st moment at each position	25.9±0.06
2nd moment at each position	26.0±0.13
Unnormalized 2nd moment at each position, i.e. LRN	26.3±0.05

Table 8. MoEx with different normalization methods on CIFAR-100.

Model	Before Stage 1	Before Stage 2	Before Stage 3
LN	25.9 ± 0.08	25.9 ± 0.07	**25.6 ± 0.02**
IN	26.0 ± 0.13	**25.7 ± 0.13**	26.2 ± 0.13
GN	**25.7 ± 0.09**	26.1 ± 0.09	25.8 ± 0.13
PONO	**25.5 ± 0.09**	26.1 ± 0.03	26.0 ± 0.09

Table 9. MoEx with different normalization methods applied to different layers in a 3-stage ResNet-110 on CIFAR-100. We bold the best layer of each normalization method.

5.2. MoEx Hyper-parameters

In MoEx, λ and $1 - \lambda$ serve as the target interpolation weights of labels y_A, y_B, respectively. To explore the relationship between λ and model performance, we train a ResNet-50 on ImageNet with $\lambda \in \{0.3, 0.5, 0.7, 0.9\}$ with on PONO. The results are summarized in Table 10. We observe that generally higher λ leads to lower error, probably because more information is captured in the normalized features than in the moments. After all, moments only capture general statistics, whereas the features have many channels and can capture texture information in great detail. We also investigate various values of the exchange probability p (for fixed $\lambda = 0.9$), but on the ImageNet data $p = 1$ (i.e. apply MoEx on every image) tends to perform best.

Model	λ	p	Top-1 / Top-5 Error(%)
ResNet-50	1	0	23.1 / 6.6
	0.9	0.25	22.6 / 6.6
	0.9	0.5	22.4 / 6.4
	0.9	0.75	22.3 / 6.3
	0.3	1	22.9 / 6.9
	0.5	1	22.2 / 6.4
	0.7	1	**21.9** / 6.2
	0.9	1	**21.9 / 6.1**
	0.95	1	22.5 / 6.3
	0.99	1	22.6 / 6.5

Table 10. Ablation study on ImageNet with different λ and p (exchange probability) trained for 300 epochs.

5.3. Model Analysis

To estimate the robustness of the models trained with MoEx, we follow the procedure proposed by [18] and evaluate our model on their ImageNet-A data set, which contains 7500 natural images (not originally part of Ima-

geNet) that are misclassified by a publicly released ResNet-50 in torchvision. We compare our models with various publicly released pretrained models including Cutout [69], Mixup [69], CutMix [67], Shape-ResNet [10], and recently proposed AugMix [17]. We report all 5 metrics implemented in the official evaluation code: model accuracy (Acc), root mean square calibration rrror (RMS), mean absolute distance calibration error (MAD), the area under the response rate accuracy curve (AURRA) and soft F1 [18, 52]. Table 11 summarizes all results. In general MoEx performs fairly well across the board. The combination of MoEx and Cutmix leads to the best performance on most of the metrics.

Name	Acc↑	RMS↓	MAD↓	AURRA↑	Soft F1↑
ResNet-50 (torchvision)	0	62.6	55.8	0	60.0
Shape-ResNet	2.3	57.8	50.7	1.8	62.1
AugMix	3.8	51.1	43.7	3.3	66.8
Fast AutoAugment	4.7	54.7	47.8	4.5	62.3
Cutout	4.4	55.7	48.7	3.8	61.7
Mixup	6.6	51.8	44.4	7.0	63.7
Cutmix	7.3	45.0	36.5	7.2	69.3
ResNet-50 (300 epochs)	4.2	54.0	46.8	3.9	63.7
MoEx	5.5	43.2	**34.2**	5.7	**72.9**
Cutmix + MoEx	**7.9**	**42.6**	34.3	**8.5**	70.5

Table 11. The performance of ResNet-50 variants on ImageNet-A. The up-arrow represents the higher the better, and vice versa.

5.4. Beyond Computer Vision

We also found that MoEx can be beneficial in other areas such as natural language processing and speech recognition. We use the Speech Command dataset [59] which contains 65000 utterances (one second long) from thousands of people. The goal is to classify them in to 30 command words such as "Go", "Stop", etc. There are 56196, 7477, and 6835 examples for training, validation, and test. We use an open source implementation to encode each audio into a mel-spectrogram of size 1x32x32 and feeds it to 2D ConvNets as an one-channel input. We follow the default setup in the codebase training models with initial learning rate 0.01 with ADAM [29] for 70 epochs. The learning rate is reduce on plateau. We use the validation set for hyper-parameter selection and tune MoEx $p \in \{0.25, 0.5, 0.75, 1\}$ and $\lambda \in \{0.5, 0.9\}$. We test the proposed MoEx on three baselines models: DenseNet-BC-100, VGG-11-BN, and WRN-28-10.

Table 12 displays the validation and test errors. We observe that training models with MoEx improve over the baselines significantly in all but one case. The only exception is DenseNet-BC-100, which has only 2% of the parameters of the wide ResNet, confirming the findings of Zhang et al. [69] that on this data set data augmentation has little effect on tiny models.

Model	# Param	Val Err	Test Err
DenseNet-BC-100	0.8M	3.16	**3.23**
+MoEx	0.8M	**2.97**	3.31
VGG-11-BN	28.2M	3.05	3.38
+MoEx	28.2M	**2.76**	**3.00**
WRN-28-10	36.5M	2.42	2.21
+MoEx	36.5M	**2.22**	**1.98**

Table 12. Speech classification on Speech Command. Similar to the observation of [69], regularization methods work better for models with large capacity on this dataset.

In addition, please see Appendix for additional natural language processing results. In contrast to prior augmentation methods, which combine two images in pixel or feature space through linear or non-linear interpolation, MoEx extracts and injects (higher order) statistics about the features.Moments are a natural first choice, but other statistics are possible (e.g. principal components).

6. Conclusion and Future Work

In this paper we propose MoEx, a novel data augmentation algorithm for deep recognition models. Instead of disregarding the moments extracted by the (intra-instance) normalization layer, it forces the neural network to pay special attention towards them. We show empirically that this approach is consistently able to improve classification accuracy and robustness across many data sets, model architectures, and prediction tasks. As an augmentation method in feature space, MoEx is complementary to existing state-of-the-art approaches and can be readily combined with them. Because of its ease of use and extremely simple implementation we hope that MoEx will be useful to many practitioners in computer vision, and beyond — in fact, anybody who trains discriminative deep networks with mini-batches.

Acknowledgments

This research is supported in part by the grants from Facebook, DARPA, the National Science Foundation (III-1618134, III-1526012, IIS1149882, IIS-1724282, and TRIPODS-1740822), the Office of Naval Research DOD (N00014-17-1-2175), Bill and Melinda Gates Foundation. We are thankful for generous support by Zillow and SAP America Inc. Facebook has no collaboration with the other sponsors of this project. In particular, we appreciate the valuable discussion with Gao Huang.

https://download.pytorch.org/models/resnet50-19c8e357.pth

https://github.com/hendrycks/natural-adv-examples

We attribute the Speech Command dataset to the Tensorflow team and AIY project: https://ai.googleblog.com/2017/08/launching-speech-commands-dataset.html

https://github.com/tugstugi/pytorch-speech-commands

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. arXiv preprint arXiv:1607.06450, 2016. 2, 4, 7

[2] Nils Bjorck, Carla P Gomes, Bart Selman, and Kilian Q Weinberger. Understanding batch normalization. In NeurIPS, 2018. 1, 2

[3] Isaac Caswell, Ciprian Chelba, and David Grangier. Tagged back-translation. In WMT, 2019. 3

[4] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In CVPR, 2019. 3, 5

[5] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical data augmentation with no separate search. arXiv preprint arXiv:1909.13719, 2019. 3

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In CVPR, 2009. 2, 5

[7] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. arXiv preprint arXiv:1708.04552, 2017. 3, 5

[8] Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. Understanding back-translation at scale. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 489–500, 2018. 3

[9] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In ICLR Workshop on Representation Learning on Graphs and Manifolds, 2019. 6

[10] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. arXiv preprint arXiv:1811.12231, 2018. 2, 5, 8

[11] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Dropblock: A regularization method for convolutional networks. In NeurIPS, 2018. 3, 5

[12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Advances in neural information processing systems, pages 2672–2680, 2014. 1

[13] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. arXiv preprint arXiv:1706.02677, 2017. 5

[14] John Gurland and Ram C Tripathi. A simple approximation for unbiased estimation of the standard deviation. The American Statistician, 25(4):30–32, 1971. 5

[15] Dongyoon Han, Jiwhan Kim, and Junmo Kim. Deep pyramidal residual networks. In CVPR, 2017. 5

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385, 2015. 1, 5, 6

[17] Dan Hendrycks, Norman Mu, Ekin D. Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. AugMix: A simple data processing method to improve robustness and uncertainty. ICLR, 2020. 8

[18] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. arXiv preprint arXiv:1907.07174, 2019. 7, 8

[19] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 7132–7141, 2018. 1

[20] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In CVPR, 2017. 5

[21] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In ECCV, 2016. 5

[22] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In ICCV, 2017. 1, 3

[23] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In ECCV, 2018. 1

[24] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167, 2015. 1, 2, 4

[25] Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. Deep unordered composition rivals syntactic methods for text classification. In ACL-IJNLP, 2015. 3

[26] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In ICLR, 2018. 7

[27] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In CVPR, 2019. 1, 3

[28] Kenji Kawaguchi, Yoshua Bengio, Vikas Verma, and Leslie Pack Kaelbling. Towards understanding generalization via analytical learning theory. arXiv preprint arXiv:1802.07426, 2018. 3

[29] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014. 6, 8

[30] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114, 2013. 1

[31] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 4

[32] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In NeurIPS, pages 1097–1105, 2012. 1, 7

[33] Y. LeCun, L. Bottou, G. Orr, and K. Muller. Efficient backprop. In G. Orr and Muller K., editors, Neural Networks: Tricks of the trade. Springer, 1998. 2

[34] Boyi Li, Felix Wu, Kilian Q Weinberger, and Serge Belongie. Positional normalization. In NeurIPS, 2019. 1, 2, 3, 7

[35] Guoying Li and Jian Zhang. Sphering and its properties. Sankhyā: The Indian Journal of Statistics, Series A, 1998. 2

[36] Yingwei Li, Qihang Yu, Mingxing Tan, Jieru Mei, Peng Tang, Wei Shen, Alan Yuille, and Cihang Xie. Shape-texture debiased neural network training. arXiv preprint arXiv:2010.05981, 2020. 3

[37] Sungbin Lim, Ildoo Kim, Taesup Kim, Chiheon Kim, and Sungwoong Kim. Fast autoaugment. In NeurIPS, 2019. 5

[38] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He,

Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In CVPR, 2017. 6

[39] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In ECCV, 2014. 6

[40] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. arXiv preprint arXiv:1608.03983, 2016. 5

[41] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2337–2346, 2019. 1

[42] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 4

[43] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In NeurIPS, 2017. 6

[44] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In NeurIPS, 2015. 6

[45] Stephane Ross, Paul Mineiro, and John Langford. Normalized online learning. In UAI, 2013. 2

[46] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In NeurIPS, 2018. 1

[47] Bernhard Schölkopf, Chris Burges, and Vladimir Vapnik. Incorporating invariances in support vector learning machines. In ICANN. Springer, 1996. 3

[48] Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving neural machine translation models with monolingual data. arXiv preprint arXiv:1511.06709, 2015. 3

[49] Patrice Simard, Yann LeCun, and John S Denker. Efficient pattern recognition using a new transformation distance. In NeurIPS, 1993. 3

[50] Patrice Y Simard, Dave Steinkraus, and John C Platt. Best practices for convolutional neural networks applied to visual document analysis. In ICDAR, 2003. 3

[51] Krishna Kumar Singh and Yong Jae Lee. Hide-and-seek: Forcing a network to be meticulous for weakly-supervised object and action localization. In ICCV, 2017. 3

[52] Marina Sokolova, Nathalie Japkowicz, and Stan Szpakowicz. Beyond accuracy, f-score and roc: a family of discriminant measures for performance evaluation. In Australasian joint conference on artificial intelligence. Springer, 2006. 8

[53] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In CVPR, 2016. 6, 7

[54] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In ICML, pages 6105–6114, 2019. 5

[55] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. arXiv preprint arXiv:1607.08022, 2016. 1, 2, 4, 7

[56] Laurens van der Maaten, Minmin Chen, Stephen Tyree, and Kilian Weinberger. Learning with marginalized corrupted features. In ICML, 2013. 3

[57] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In NeurIPS, 2017. 3, 7

[58] Yulin Wang, Xuran Pan, Shiji Song, Hong Zhang, Gao Huang, and Cheng Wu. Implicit semantic data augmentation for deep networks. In NeurIPS, 2019. 3, 5

[59] Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition. arXiv preprint arXiv:1804.03209, 2018. 8

[60] Yuxin Wu and Kaiming He. Group normalization. In ECCV, 2018. 1, 2, 4, 7

[61] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 6

[62] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In CVPR, 2015. 6

[63] Cihang Xie, Mingxing Tan, Boqing Gong, Jiang Wang, Alan Yuille, and Quoc V Le. Adversarial examples improve image recognition. arXiv preprint arXiv:1911.09665, 2019. 3

[64] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In CVPR, pages 1492–1500, 2017. 5

[65] Yoshihiro Yamada, Masakazu Iwamura, Takuya Akiba, and Koichi Kise. Shakedrop regularization for deep residual learning. arXiv preprint arXiv:1802.02375, 2018. 5

[66] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. Qanet: Combining local convolution with global self-attention for reading comprehension. arXiv preprint arXiv:1804.09541, 2018. 3

[67] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In ICCV, 2019. 2, 3, 4, 5, 6, 8

[68] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. arXiv preprint arXiv:1605.07146, 2016. 5

[69] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In ICLR, 2018. 1, 2, 3, 5, 8

[70] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In AAAI, 2020. 3