A Maximum entropy transformations

To guarantee as much diversity as possible in our model of common corruptions, we follow the principle of maximum entropy to define our distributions of transformations [8]. Note that using a set of augmentations that guarantees maximum entropy comes naturally when trying to optimize the sample complexity derived from certain information theoretic generalization bounds, both in the clean [42] and corrupted setting [28]. Specifically, the principle of maximum entropy postulates favoring those distributions that are as unbiased as possible given the set of constraints that defines a family of distributions. In our case, these constraints are given in the form of an expected strength, i.e., σ^2, desired smoothness, i.e., K, and/or some boundary conditions, e.g.,, the displacement field must be zero at the borders of an image.

Let us make this formal. In particular, let $\mathcal{I}$ denote the space of all images $\boldsymbol{x} : \mathbb{R}^2 \to \mathbb{R}^3$, and let $f : \mathcal{I} \to \mathcal{I}$ denote a random image transformation distributed according to the law μ. Further, let us define a set of constraints $\mathcal{C} \subseteq \mathcal{F}$, which restrict the domain of applicability of f, i.e., $f \in \mathcal{C}$, and where $\mathcal{F}$ denotes the space of functions $\mathcal{I} \to \mathcal{I}$. The principle of maximum entropy postulates using the distribution μ which has maximum entropy given the constraints:

$$\begin{aligned} \underset{\mu}{\text{maximize}} \quad & H(\mu) = \int_{\mathcal{F}} \mathrm{d}\mu(f) \log(\mu(f)) \\ \text{subject to} \quad & f \in \mathcal{C} \quad \forall f \sim \mu, \end{aligned} \tag{8}$$

where $H(\mu)$ represents the entropy of the distribution μ [8]. In its general form, solving Eq. (8) for any set of constraints $\mathcal{C}$ is intractable. However, leveraging results from statistical physics, we will see that for our domains of interest, Eq. (8) has a simple solution. In what follows we derive those distributions for each of our family of transformations.

A.1 Spectral domain

As we introduced in Sec. 2, we propose to parameterize our family of spectral transformations using an FIR filter of size $K_\omega \times K_\omega$. That is, we are interested in finding a maximum entropy distribution over the space of spectral transformations with a finite spatial support.

Nevertheless, on top of this smoothness constraint we are also interested in controlling the strength of the transformations. We define the strength of a distribution of random spectral transformations applied to an image $\boldsymbol{x}$, as the expected L_2 norm of the difference between the clean and transformed images, i.e.,

$$\mathbb{E}_\omega \|\boldsymbol{x} - \omega(\boldsymbol{x})\|_2^2 = \mathbb{E}_{\boldsymbol{\omega}'} \|\boldsymbol{\omega}' * \boldsymbol{x}\|_2^2, \tag{9}$$

which using Young's convolution inequality is bounded as

$$\mathbb{E}_{\boldsymbol{\omega}'} \|\boldsymbol{\omega}' * \boldsymbol{x}\|_2^2 \leq \|\boldsymbol{x}\|_1^2 \, \mathbb{E}_{\boldsymbol{\omega}'} \|\boldsymbol{\omega}'\|_2^2. \tag{10}$$

Indeed, we can see that the strength of a distribution of random smooth spectral transformations is governed by the expected norm of its filter. In the discrete domain, this can be simply computed as

$$\mathbb{E}_{\boldsymbol{\omega}'}\|\boldsymbol{\omega}'\|_2^2 = \sum_{i=1}^{K_\omega}\sum_{j=1}^{K_\omega}\mathbb{E}_{\boldsymbol{\omega}'}\boldsymbol{\omega}'^2_{i,j}. \tag{11}$$

Considering this, we should then look for a maximum entropy distribution whose samples satisfy

$$\mathcal{C} = \left\{\boldsymbol{\omega}' \in \mathbb{R}^{K_\omega \times K_\omega} \wedge \mathbb{E}_{\boldsymbol{\omega}'}\|\boldsymbol{\omega}'\|_2^2 = K_\omega^2\sigma_\omega^2 \,|\, \omega \sim \mu_\omega\right\}. \tag{12}$$

Now, note that this set is defined by an equality constraint involving a sum of K_ω^2 quadratic random variables. In this sense, we know that the Equipartition Theorem [1] applies and can be used to identify the distribution of maximum entropy. That is, the solution of Eq. (8) in the case that $\mathcal{C}$ is given by Eq. (12), is equal to the distribution of FIR filters whose coefficients are iid with law $\mathcal{N}(0, \sigma_\omega^2)$.

A.2 Spatial domain

The distribution of diffeomorphisms of maximum entropy with a fixed norm was derived by Petrini *et al.* in [32]. The derivation is similar to the spectral domain, but with the additional constraint that the diffeomorphisms produce a null displacement at the borders of the image.

A.3 Color domain

We can follow a very similar route to derive the distribution of maximum entropy among all color transformations, where, specifically, we constraint the transformations to yield $\gamma(0) = 0$ and $\gamma(1) = 1$ on every channel independently. Doing so, the derivation of the maximum entropy distribution can follow the same steps as in [32].

B PRIME implementation details

In this section, we provide additional details regarding the implementation of PRIME described in Sec. 3. Since the parameters of the transformations are empirically selected, we first provide more visual examples for different values of smoothness K and strength σ. Then, we give the exact values of the parameters we use in our experiments supported by additional visual examples and we also describe the parameters we use for the mixing procedure.

B.1 Additional transformed examples

We provide additional visual examples for each of the primitives of PRIME illustrating the effect of the following two factors: (i) smoothness controlled by parameter K, and (ii) strength of the transformation σ on the resulting transformed images created by the primitives. Figs. 6, 7 and 8 demonstrate the resulting spectrum of images created by applying spectral, spatial and color transformations while varying the parameters K and σ. Notice how increasing the strength σ of each transformation drifts the augmented image farther away from its clean counterpart, yet produces plausible images when appropriately controlled.

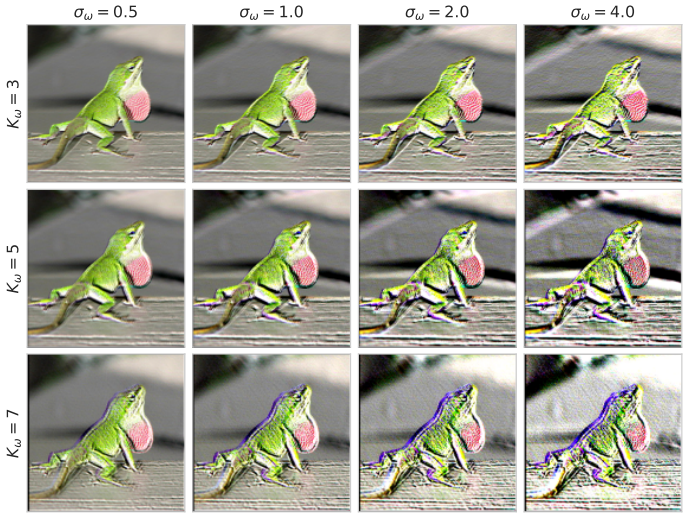


Fig. 6. Example images (IN) generated with spectral transformations from our common corruptions model. In each row, we enlarge the transformation strength σ_ω from left to right. From top to bottom, we increase the spectral resolution of the filter K_ω.

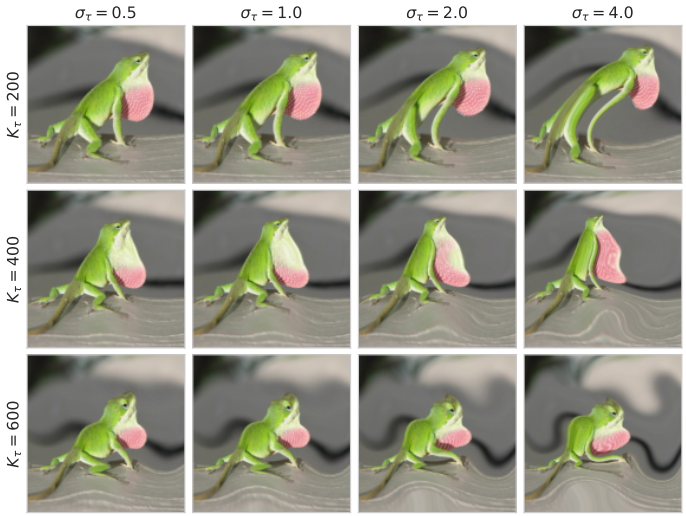


Fig. 7. Example images (IN) generated with spatial transformations from our common corruptions model. In each row, we enlarge the transformation strength σ_τ from left to right. From top to bottom, we increase the cut frequency K_τ.

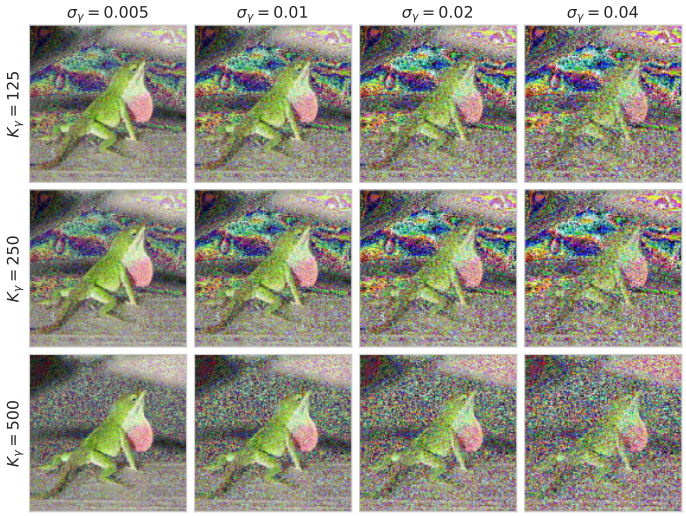


Fig. 8. Example images (IN) generated with color transformations from our common corruptions model. In each row, we enlarge the transformation strength σ_γ from left to right. From top to bottom, we increase K_γ.

B.2 Transformation parameters

We now provide the parameters of each transform that we selected and used in our experiments. In general, the values might vary for inputs of different dimensionality and resolution (i.e., CIFAR-10/100 vs ImageNet images).

Spectral transform Regarding the spectral transform of Eq. (3) we found out that, for the FIR filter $\boldsymbol{\omega}'$, a size of $K_\omega = 3$ results into semantically preserving images for CIFAR-10/100 and ImageNet. For the latter, one can stretch the filter size to 5×5 or even 7×7, but then slight changes on the strength, σ_ω, might destroy the image semantics. Eventually, given $K_\omega = 3$, we observed that $\sigma_\omega = 4$ is good enough for CIFAR-10/100 and ImageNet.

Spatial transform Concerning the spatial transform of Eq. (5), for the cut-off parameter K_τ we followed the value regimes proposed by Petrini *et al.* [32] and set $K_\tau = 100$ for CIFAR-10/100; $K_\tau = 500$ for ImageNet. Furthermore, for a given K_τ, Petrini *et al.* also compute the appropriate bounds for the transformation strength, $\sigma^2_{\tau_{\min}} \leq \sigma^2_\tau \leq \sigma^2_{\tau_{\max}}$, such that the resulting diffeomorphism remains bijective and the pixel displacement does not destroy the image. In fact, in their original implementation[6], Petrini *et al.* directly sample $\sigma_\tau \sim U(\sigma_{\tau_{\min}}, \sigma_{\tau_{\max}})$ instead of explicitly setting the strength. In our implementation, we also follow the same approach.

Color transform Regarding the color transform of Eq. (6) we found out that for CIFAR-10/100 a cut-off value of $K_\gamma = 10$ and a strength of $\sigma_\gamma = 0.01$ result into semantically preserving images for CIFAR-10/100; while for ImageNet, the corresponding values are $K_\gamma = 500$ and $\sigma_\gamma = 0.05$. As for the bandwidth (consecutive frequencies) Δ we observed that a value of $\Delta = 20$ was memory sufficient for ImageNet, but for CIFAR-10/100, due to its lower dimensionality, we can afford all the frequencies to be used, e.g., $\Delta = K_\gamma$.

Finally, as mentioned in Sec. 3, we randomly sample the strength of the transformations σ from a uniform distribution of given minimum and maximum values. Regarding the maximum, we always set it to be the one we selected through visual inspection, while the minimum is set to 0. Fig. 9 displays additional augmented images created by applying each of the primitive transformations in our model using the aforementioned set of parameters on ImageNet. Our choice of parameters produces diverse image augmentations, while retaining the semantic content of the images.

[6] The official implementation of Petrini *et al.* diffeomorphisms can be found at https://github.com/pcsl-epfl/diffeomorphism.

Fig. 9. Example images (IN) generated with the transformations of our common corruptions model. Despite the perceptibility of the introduced distortion, the image semantics are preserved.

B.3 Parameters for mixing procedure

Regarding the mixing parameters of our experiments, we fix the total number of generated transformed images (width) to be $n = 3$. As for the composition of the transformations (depth), we follow a stochastic approach such that, on every iteration $i \in \{1, \ldots, n\}$, only $\hat{m} \in [1, m]$ compositions are performed, with $m = 3$. In fact, in Algorithm 1 we do not explicitly select randomly a new $\hat{m}$ for every i but we provide the identity operator Id instead. This guarantees that, in some cases, no transformation is performed.

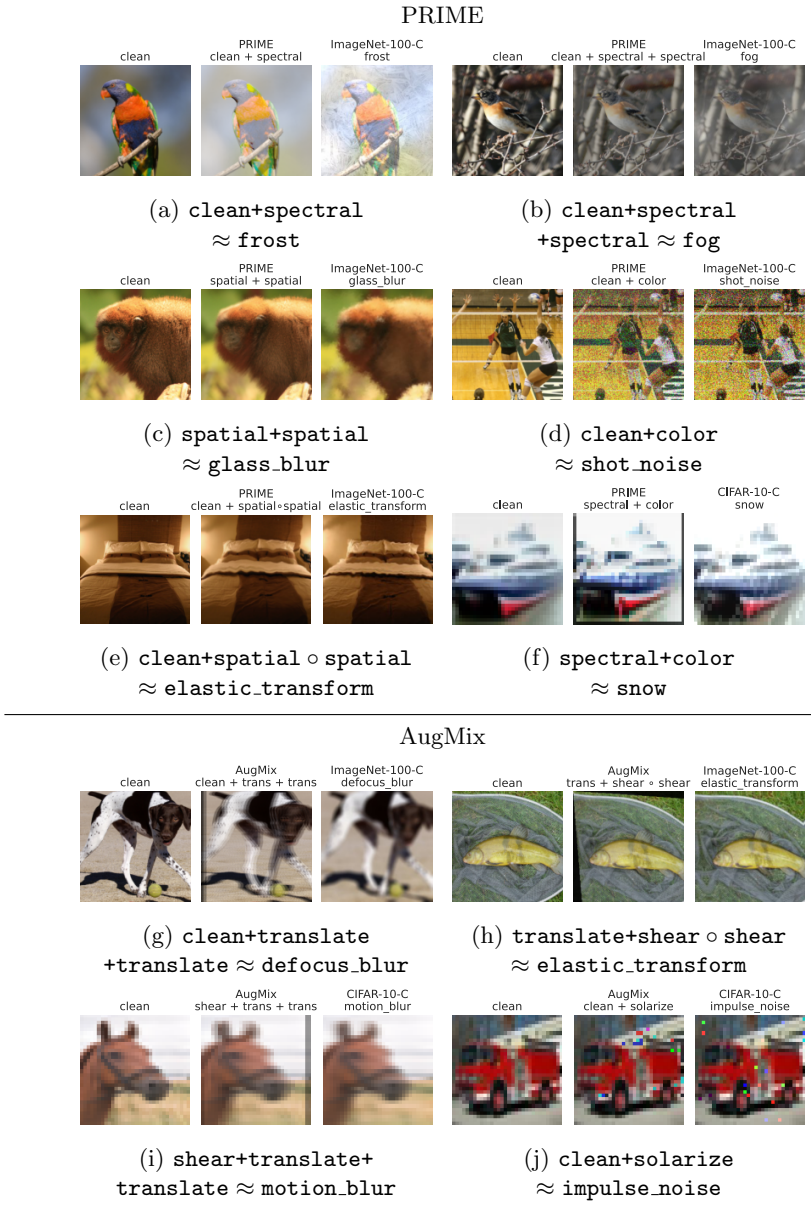
C Detailed experimental setup

We now provide all the experimental details for the performance evaluation of Sec. 4. All models are implemented in PyTorch [48] and are trained for 100 epochs using a cyclic learning rate schedule [49] with cosine annealing and a maximum learning rate of 0.2 unless stated otherwise. For IN, we fine-tune a regularly pretrained network (provided in PyTorch) with a maximum learning rate of 0.01 following Hendrycks *et al.* [20]. We use SGD optimizer with momentum factor 0.9 and Nesterov momentum [47]. On C-10 & C-100, we set the batch size to 128 and use a weight decay of 0.0005. On IN-100 and IN, the batch size is 256 and weight decay is 0.0001. We employ ResNet-18 [19] on C-10, C-100 and IN-100; and use ResNet-50 for IN. The augmentation hyperparameters for AugMix and DeepAugment are the same as in their original implementations.

D Additional mixing examples

Continuing Sec. 5.2, we present additional examples in Fig. 10 to demonstrate the significance of mixing in PRIME. We observe that the mixing procedure is capable of constructing augmented images that look perceptually similar to common corruptions. To illustrate this, we provide several examples in Fig. 10 for PRIME (upper half) and AugMix (lower half) on CIFAR-10 and ImageNet-100. As shown in Figs. 10a and 10b, mixing spectral transformations with the clean images tends to create weather-like artefacts resembling frost and fog respectively. Carefully combining clean and spatially transformed images produces blurs (Fig. 10c) and even elastic transform (Fig. 10e). Moreover, blending color augmentation with clean image produces shot noise as evident in Fig. 10d; Whereas spectral+color transformed image looks similar to snow corruption (Fig. 10f). All these observations explain the good performance of PRIME on the respective corruptions.

Apart from the mixing in PRIME, the mixing in AugMix also plays a crucial role in its performance. In fact, a combination of translate and shear operations with the clean image create blur-like modifications that resemble defocus blur (Fig. 10g) and motion blur (Fig. 10i). This answers why AugMix excels at blur corruptions and is even better than DeepAugment against blurs (cf. Tab. 6). In addition, on CIFAR-10, notice that mixing solarize and clean produces impulse noise-like modifications (Fig. 10j), which justifies the improvements on noise attained by AugMix (refer Tab. 5).



(a) `clean+spectral` ≈ `frost`

(b) `clean+spectral` `+spectral` ≈ `fog`

(c) `spatial+spatial` ≈ `glass_blur`

(d) `clean+color` ≈ `shot_noise`

(e) `clean+spatial` ∘ `spatial` ≈ `elastic_transform`

(f) `spectral+color` ≈ `snow`

(g) `clean+translate` `+translate` ≈ `defocus_blur`

(h) `translate+shear` ∘ `shear` ≈ `elastic_transform`

(i) `shear+translate+` `translate` ≈ `motion_blur`

(j) `clean+solarize` ≈ `impulse_noise`

Fig. 10. The mixing procedure creates distorted images that look visually similar to the test-time corruptions. In each example (CIFAR-10/ImageNet-100), we show the clean image, the PRIME/AugMix augmented image and the corresponding common corruption that resembles the image produced by mixing. We also report the mixing combination used for recreating the corruption. ∘ stands for composition and + represents convex combination (mixing). (Top 3 rows): PRIME, and (Last 2 rows): AugMix.

E SimCLR nearest neighbours

Regarding the minimum distances in the SimCLRv2 embedding space of Tab. 3, we also provide in Fig. 11 some visual examples of the nearest neighbours of each method. In general, we observe that indeed smaller distance in the embedding space typically corresponds to closer visual similarity in the input space, with PRIME generating images that resemble more the corresponding common corruptions, compared to AugMix. Nevertheless, we also notice that for "Blurs" AugMix generates images that are more visually similar to the corruptions than PRIME, an observation that is on par with the lower performance of PRIME (without JSD) on blur corruptions (cf. Tab. 6) compared to AugMix.

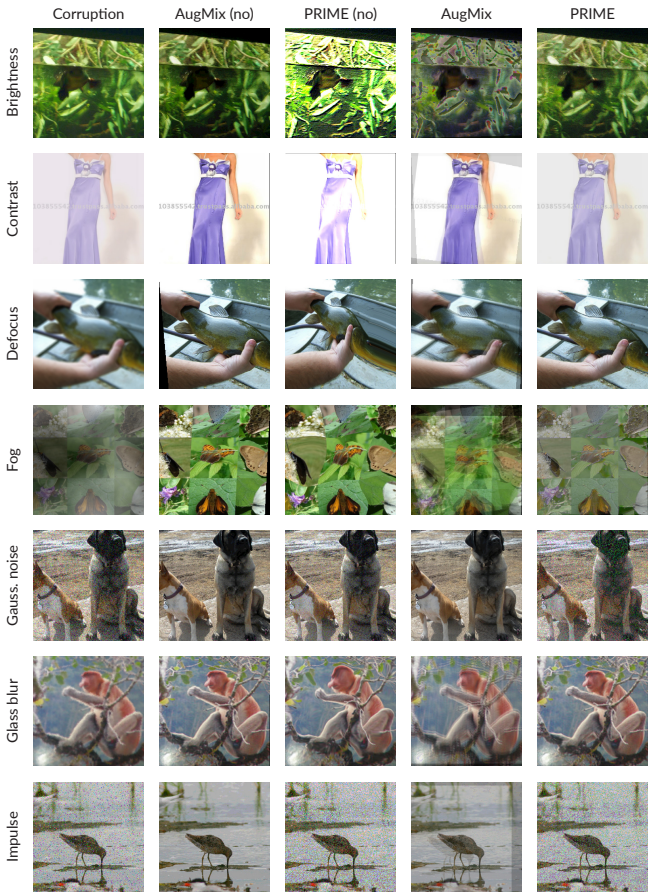


Fig. 11. Examples of nearest neighbours in SimCLRv2 embedding space. Columns: (first): the common corruption; (second): AugMix transformations (no mixing); (third): PRIME transformations (no mixing); (fourth): AugMix; (fifth): PRIME.

Table 4. Percentiles of the minimum cosine distances in the ResNet-50 SimCLRv2 embedding space between 100 augmented samples from 1000 ImageNet images, and their corresponding common corruptions.

Method	Min. cosine distance ($\times 10^{-3}$) (↓)				
	5%	10%	25%	50%	75%
None (clean)	0.33	0.64	1.97	6.43	17.44
AugMix (w/o mix)	0.17	0.31	1.04	3.55	10.71
PRIME (w/o mix)	0.04	0.07	0.24	1.87	7.11
AugMix	0.11	0.21	0.69	2.61	8.37
PRIME	0.08	0.12	0.32	1.61	5.76

F Cosine distance statistics

Recall that in Tab. 3 we provide the average and the median of the minimum cosine distances computed in the SimCLRv2 embedding space. We now provide in Tab. 4 the values for different percentiles of these distances. We observe that the behaviour is consistent across different percentiles: PRIME (with or without mixing) is always producing feature representations that are more similar to the common corruptions, compared to any version of AugMix. Note also that for smaller percentiles $(5\%, 10\%, 25\%)$ it seems that PRIME without mixing reaches even lower values than PRIME. However, the difference with respect to PRIME can be considered as insignificant since it is in the order of 10^{-5} (note that all values in the table are in the order of 10^{-3}); while a larger population of images (> 1000) would potentially smooth out this difference.

G Embedding space visualization

To qualitatively compare how diverse are the augmentations of PRIME with respect to other methods, we can follow the procedure in [41]. We randomly select 3 images from ImageNet, each one belonging to a different class. For each image, we generate 100 transformed instances using AugMix and PRIME, while with DeepAugment we can only use the original images and the 2 transformed instances that are pre-generated with the EDSR and CAE image-to-image networks that DeepAugment uses. Then, we pass the transformed instances of each method through a ResNet-50 pre-trained on ImageNet and extract the features of its embedding space. On the features extracted for each method, we perform PCA and then visualize the projection of the features onto the first two principal components. We visualize the projected augmented space in Fig. 12, which demonstrates that PRIME generates more diverse (larger variance) features than AugMix and DeepAugment.

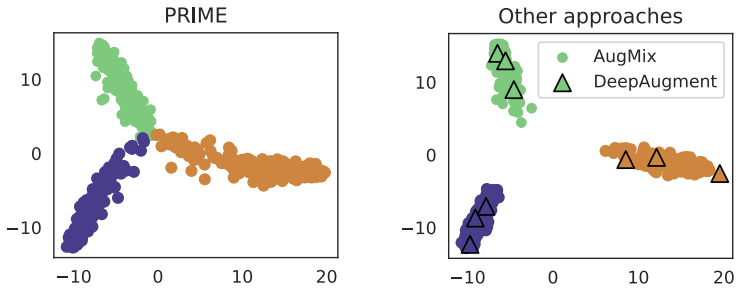


Fig. 12. Projections of augmentations generated by different methods on the embedding space of a ResNet-50.

H Performance per corruption

Beyond the average corruption accuracy that we report in Tab. 1, we also provide here the performance of each method on the individual corruptions. The results on CIFAR-10/100 and ImageNet/ImageNet-100 are shown on Tab. 5 and Tab. 6 respectively. Compared to AugMix on CIFAR-10/100, the improvements from PRIME are mostly observed against Gaussian noise (+7.6%/12.3%), shot noise (+3.3%/7.0%), glass blur (+6.4%/11.0%) and JPEG compression (+1.3%/2.6%). These results show that PRIME can really push the performance against certain corruptions in CIFAR-10/100-C despite the fact that AugMix is already good on these datasets. However, AugMix turns out to be slightly better than PRIME against impulse noise, defocus blur and motion blur modifications; all of which have been shown to be resembled by AugMix created images (see Fig. 10). With ImageNet-100, PRIME enhances the diversity of augmented images, and leads to general improvements against all corruptions except certain blurs. On ImageNet, we observe that, in comparison to DeepAugment, the supremacy of PRIME is reflected on almost every corruption type, except some blurs and pixelate corruptions where DeepAugment is slightly better. When PRIME is used in conjunction with DeepAugment, compared to AugMix combined with DeepAugment, our method seems to lack behind only on blurs, while on the rest of the corruptions achieves higher robustness.

Table 5. Per-corruption accuracy of different methods on C-10/100 (ResNet-18).

Dataset	Method	Clean	CC	Noise			Blur				Weather				Digital			
				Gauss.	Shot	Impulse	Defoc.	Glass	Motion	Zoom	Snow	Frost	Fog	Bright.	Contr.	Elastic	Pixel.	JPEG
C-10	Standard	95.0	74.0	45.1	58.7	54.9	83.2	53.3	76.9	79.1	83.1	79.3	89.0	93.6	76.3	83.9	75.1	77.9
	AugMix	95.2	88.6	79.3	84.8	85.8	94.1	78.9	92.4	93.4	89.7	89.0	91.9	94.3	90.5	90.5	87.6	87.5
	PRIME	94.2	89.8	86.9	88.1	88.6	92.6	85.3	90.8	92.2	89.3	90.5	89.8	93.7	92.4	90.1	88.1	88.8
C-100	Standard	76.7	51.9	25.3	33.7	26.6	60.8	47.1	55.5	57.6	60.8	56.2	62.5	72.2	53.2	63.4	50.1	52.7
	AugMix	78.2	64.9	46.7	55.1	60.6	76.2	47.3	72.6	74.3	67.4	64.4	69.9	75.5	67.4	69.6	64.9	61.8
	PRIME	78.4	68.2	59.0	62.1	68.1	74.0	58.3	70.5	72.3	68.9	68.5	69.8	76.8	74.4	70.1	65.5	64.4

Table 6. Per-corruption accuracy of different methods on IN-100 (ResNet-18) and IN (ResNet-50). † indicates that JSD consistency loss is not used. *Models taken from [9].

Dataset	Method	Clean	CC	Noise			Blur				Weather				Digital			
				Gauss.	Shot	Impulse	Defoc.	Glass	Motion	Zoom	Snow	Frost	Fog	Bright.	Contr.	Elastic	Pixel.	JPEG
IN-100	Standard	88.0	49.7	30.9	29.0	22.0	45.6	44.6	50.4	53.9	43.8	46.2	50.5	78.6	42.9	68.8	68.0	70.6
	AugMix	88.7	60.7	45.2	45.8	43.4	58.7	53.3	69.5	71.0	49.1	52.7	60.2	80.7	59.6	73.3	73.6	74.7
	DA	86.3	67.7	76.3	75.6	75.7	64.2	61.7	61.3	62.7	54.4	62.8	55.7	81.6	49.7	69.9	83.3	80.6
	PRIME	85.9	71.6	80.6	80.0	80.1	57.2	66.3	66.2	68.2	61.5	68.2	57.2	81.2	68.3	73.7	82.9	81.9
	DA+AugMix	86.5	73.1	75.2	75.8	74.9	74.1	68.5	76.0	72.1	59.9	66.8	61.4	82.1	72.4	73.1	83.8	81.1
	DA+PRIME	84.9	74.9	81.1	80.9	81.2	70.5	74.2	72.0	71.5	66.3	73.6	56.6	81.9	72.8	74.8	83.4	82.3
IN	Standard*	76.1	39.2	29.3	27.0	23.8	38.8	26.8	38.7	36.2	32.5	38.1	45.4	68.0	39.0	45.3	44.8	53.4
	AugMix*	77.5	48.3	40.6	41.1	37.7	47.7	34.9	53.5	49.0	39.9	43.8	47.1	69.5	51.1	52.0	57.0	60.3
	DA*	76.7	52.6	56.6	54.9	56.3	51.7	40.1	48.7	39.5	44.2	50.3	52.1	71.1	48.3	50.9	65.5	59.3
	PRIME†	77.0	55.0	61.9	60.6	60.9	47.6	39.0	48.4	46.0	47.4	50.8	54.1	71.7	58.2	56.3	59.5	62.2
	DA+AugMix	75.8	58.1	59.4	59.6	59.1	59.0	46.8	61.1	51.5	49.4	53.3	55.9	70.8	58.7	54.3	68.8	63.3
	DA+PRIME†	75.5	59.9	67.4	67.2	66.8	56.2	47.5	54.3	47.3	52.8	56.4	56.3	71.7	62.3	57.3	70.3	65.1

I Performance per severity level

We also want to investigate the robustness of each method on different severity levels of the corruptions. The results for CIFAR-10/100 and ImageNet/ImageNet-100 are presented in Tab. 7 and Tab. 8 respectively. With CIFAR-10/100, PRIME predominantly helps against corruptions with maximal severity and yields +3.9% and +7.1% gains on CIFAR-10 and CIFAR-100 respectively. Besides on ImageNet-100, PRIME again excels at corruptions with moderate to higher severity. This observations also holds when PRIME is employed in concert with DeepAugment. With ImageNet too this trend continues, and we observe that, compared to DeepAugment, PRIME improves significantly on corruptions of larger severity (+3.4% and +5.5% on severity levels 4 and 5 respectively). Also, this behaviour is consistent even when PRIME is combined with DeepAugment and is compared to DeepAugment+AugMix, where we see that again on levels 4 and 5 there is a significant improvement of +2.1% and +3.7% respectively.

Table 7. Average accuracy for each corruption severity level of different methods on C-10 and C-100 (ResNet-18).

Dataset	Method	Clean	CC	Avg.	Severity				
					1	2	3	4	5
C-10	Standard	95.0	74.0	87.4	81.7	75.7	68.3	56.7	
	AugMix	95.2	88.6	93.1	91.8	89.9	86.7	81.7	
	PRIME	94.2	89.8	92.8	91.6	90.4	88.6	85.6	
C-100	Standard	76.7	51.9	66.7	59.4	52.8	45.0	35.4	
	AugMix	78.2	64.9	73.3	70.0	66.6	61.3	53.4	
	PRIME	78.4	68.2	74.0	71.6	69.2	65.6	60.5	

Table 8. Average accuracy for each corruption severity level of different methods on IN-100 (ResNet-18) and IN (ResNet-50). † indicates that JSD consistency loss is not used. *Models taken from `RobustBench` [9].

Dataset	Method	Clean	CC Avg.	Severity 1	2	3	4	5
IN-100	Standard	88.0	49.7	73.5	61.0	49.8	37.2	27.0
	AugMix	88.7	60.7	80.4	71.8	63.8	50.3	37.2
	DA	86.3	67.7	81.2	75.4	69.9	61.2	50.8
	PRIME	85.9	71.6	81.7	77.5	73.4	66.9	58.4
	DA+AugMix	86.5	73.1	82.7	78.0	75.5	69.6	59.9
	DA+PRIME	84.9	74.9	82.0	78.7	76.4	71.8	65.5
IN	Standard*	76.1	39.2	60.6	49.8	39.8	27.7	18.0
	AugMix*	77.5	48.3	66.7	58.3	51.1	39.1	26.5
	DA*	76.7	52.6	69.0	61.7	55.4	44.9	32.1
	PRIME†	77.0	55.0	68.9	63.1	56.9	48.3	37.6
	DA+AugMix	75.8	58.1	70.3	64.5	60.5	53.0	42.2
	DA+PRIME†	75.5	59.9	70.8	66.3	61.6	55.1	45.9

J Performance on other corruptions

Finally, to examine the universality of PRIME, we evaluate the performance of our ImageNet-100 trained models against two other corrupted datasets: (i) ImageNet-100-$\overline{\text{C}}$ (IN-100-$\overline{\text{C}}$) [29], and (ii) stylized ImageNet-100 (SIN-100) [17]. While IN-100-$\overline{\text{C}}$ is composed of corruptions that are perceptually dissimilar to those in IN-100-C, stylized IN-100 only retains global shape information and discard local texture cues from IN-100 test images, via style transfer. Thus, it would be interesting test the performance of PRIME against these datasets since it would serve as a indicator for general corruption robustness of PRIME. More information about the corruption types contained in IN-100-$\overline{\text{C}}$ is available in the original paper [29].

Tab. 9 enumerates the classification accuracy of different standalone approaches against IN-100-$\overline{\text{C}}$ on average, individual corruptions in IN-100-$\overline{\text{C}}$ and SIN-100. We can see that PRIME surpasses AugMix and DeepAugment by 4% and 1.2% respectively on IN-100-$\overline{\text{C}}$. PRIME particularly helps against certain distortions such as blue noise sample (BSmpl), inverse sparkles and plasma noise. PRIME also works well against style-transferred images in SIN-100 and improves

Table 9. Classification accuracy of different methods on IN-100-C, IN-100-$\overline{\text{C}}$ and Stylized IN-100 (SIN-100) with ResNet-18.

Method	Clean	IN-100-C Avg.	IN-100-$\overline{\text{C}}$ Avg.	IN-100-$\overline{\text{C}}$ BSmpl	Brown	Caustic	Ckbd	CSine	ISpark	Perlin	Plasma	SFreq	Spark	SIN-100
Standard	88.0	49.7	55.1	47.6	71.3	70.1	66.4	29.5	45.7	72.1	34.6	34.9	78.4	18.8
AugMix	88.7	60.7	61.0	63.0	73.2	75.3	69.4	39.9	44.9	77.4	42.8	44.7	79.8	28.0
DA	86.3	67.7	63.8	77.1	76.6	72.6	60.9	42.9	44.3	78.0	43.4	64.5	77.8	29.9
PRIME	85.9	71.6	65.0	74.9	74.3	73.2	59.2	53.4	47.5	76.8	48.6	66.9	75.5	33.1
+1.5x epochs	86.1	72.5	65.9	77.1	75.6	74.1	59.4	54.0	46.3	77.6	50.4	67.7	76.4	34.1

Table 10. Classification accuracy of different methods on IN-C, IN-$\overline{\text{C}}$, ImageNet-R (IN-R) and Stylized IN (SIN) with ResNet-50. † indicates that JSD consistency loss is not used. *Models taken from `RobustBench` [9].

Method	Clean	IN-C	IN-$\overline{\text{C}}$	IN-$\overline{\text{C}}$										IN-R	SIN
		Avg.	Avg.	BSmpl	Brown	Caustic	Ckbd	CSine	ISpark	Perlin	Plasma	SFreq	Spark		
Standard*	76.1	39.2	40.0	36.2	57.8	54.1	46.1	14.4	20.9	61.6	24.3	19.0	65.2	36.2	7.4
AugMix*	77.5	48.3	46.5	59.5	56.5	59.1	51.7	25.6	21.6	65.3	23.1	36.2	66.4	41.0	11.2
DA*	76.7	52.6	48.3	60.1	61.1	57.7	46.8	25.4	24.4	68.4	26.5	45.6	66.8	42.2	14.2
PRIME†	77.0	55.0	49.6	59.5	61.4	60.1	48.1	26.9	28.3	66.5	36.4	41.9	66.5	42.2	14.0

accuracy by 5.1% over AugMix and 3.2% over DeepAugment. Besides, the diversity of our method means that we can actually get a better performance by increasing the number of training epochs. With 1.5x training epochs, we observe about 1% accuracy refinement on each benchmark.

We also perform a similar analysis with ImageNet trained models and evaluate their robustness on three other distribution shift benchmarks: (i) IN-$\overline{\text{C}}$ [29], (ii) SIN [17] as described previously and (iii) ImageNet-R (IN-R) [20]. ImageNet-R contains naturally occurring artistic renditions (e.g., paintings, embroidery, etc.) of objects from the ImageNet dataset. The classification accuracy achieved by different methods on these datasets is listed in Tab. 10. On IN-$\overline{\text{C}}$, PRIME outperforms AugMix and DeepAugment by 3.1% and 1.3% respectively. Besides, PRIME also obtains competitive results on IN-R and SIN datasets. Altogether, our empirical results indicate that the performance gains obtained by PRIME indeed translate to other corrupted datasets.

K Unsupervised domain adaptation

Recently, robustness to common corruptions has also been of significant interest in the field of unsupervised domain adaptation [2,37]. The main difference is that, in domain adaptation, one exploits the limited access to test-time corrupted samples to adjust certain network parameters. Hence, it would be interesting to investigate the utility of PRIME under the setting of domain adaption.

To that end, we combine our method with the adaption trick of [37]. Specifically, we adjust the batch normalization (BN) statistics of our models using a few corrupted samples. Suppose $z_s \in \{\mu_s, \sigma_s\}$ are the BN mean and variance estimated from the training data, and $z_t \in \{\mu_t, \sigma_t\}$ are the corresponding statistics computed from n unlabelled, corrupted test samples, then we re-estimate the BN statistics as follows:

$$\hat{z} = \frac{N}{N+n} z_s + \frac{n}{N+n} z_t. \tag{13}$$

We consider three adaptation scenarios: single sample ($n = 1, N = 16$), partial ($n = 8, N = 16$) and full ($n = 400, N = 0$) adaptation. Here, we do not perform parameter tuning for N. As shown in Tab. 11, simply correcting BN statistics using as little as 8 corrupted samples pushes the corruption accuracy of PRIME

from 71.6% to 75.3%. In general, PRIME yields cumulative gains in combination with adaptation and has the best IN-100-C accuracy.

Table 11. Performance of different methods in concert with domain adaptation on IN-100. Partial adaptation uses 8 samples; full adaptation uses 400 corrupted samples. Network used: ResNet-18.

	IN-100-C acc. (↑)				IN-100 (↑)
Method	w/o	single	partial	full	single
Standard	49.7	53.8	62.0	63.9	88.1
AugMix	60.7	65.5	71.3	73.0	**88.3**
DA	67.7	70.2	72.7	74.6	86.3
PRIME	**71.6**	**73.5**	**75.3**	**76.6**	85.7

L Comparison with other methods

In addition to the two well-established baselines from AugMix and DeepAugment in Tab. 1, we also compare the performance of PRIME with more baselines in Tab. 12. In particular, we report the results of methods that use ResNet-18 on C-10/100 and ResNet-50 on IN. For keeping the comparisons fair, we exclude few of the very recent works that rely on ensembling or use transformations that overlap with Common Corruptions or use non-standard architectures. On C-10, AugMax achieves the highest robustness, with PRIME being only 0.5% behind, while on C-100, PRIME is by 2.5% better than AugMax. Note, though, that (i) AugMax is trained for 2× more epochs and, (ii) is roughly 10× slower than PRIME, since it requires 10 additional backward passes. On IN, PRIME is by far the best performing method, while its complexity is much lower than the very heavy AdA method.

Table 12. Performance comparison of PRIME with additional baselines. †Use extra data.

Dataset	Method	Clean Acc (↑)	Common Acc (↑)	Corruption mCE (↓)
C-10	Standard	95.0	74.0	26.0
	CARDS [13]]	95.6	76.1	23.9
	RLAT [24]	93.1	84.1	15.9
	AugMax-DuBIN [41]	**95.7**	**90.3**	**9.7**
	PRIME	94.2	89.8	10.2
C-100	Standard	76.7	51.9	48.1
	AugMax-DuBIN	**78.6**	65.7	34.3
	PRIME	78.4	**68.2**	**31.8**
IN	Standard	76.1	38.1	76.7
	Stylized-IN [17]	74.8	45.5	69.3
	AdA† [5]	73.2	–	75.0
	PRIME	**77.0**	**55.0**	**57.5**
	AdA+DA+AugMix†	69.2	–	62.9
	PRIME+DA†	**75.5**	**59.9**	**51.3**

Supplementary References

47. Nesterov, Y.E.: A method for solving the convex programming problem with convergence rate $O(1/k^2)$. Dokl. Akad. Nauk SSSR (1983)
48. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems (2019)
49. Smith, L.N., Topin, N.: Super-convergence: Very fast training of residual networks using large learning rates. arXiv preprint arXiv:1708.07120 (2018)