Improving *robustness to corruptions* with *multiplicative weight perturbations*

Trung Trinh

Markus Heinonen

Luigi Acerbi

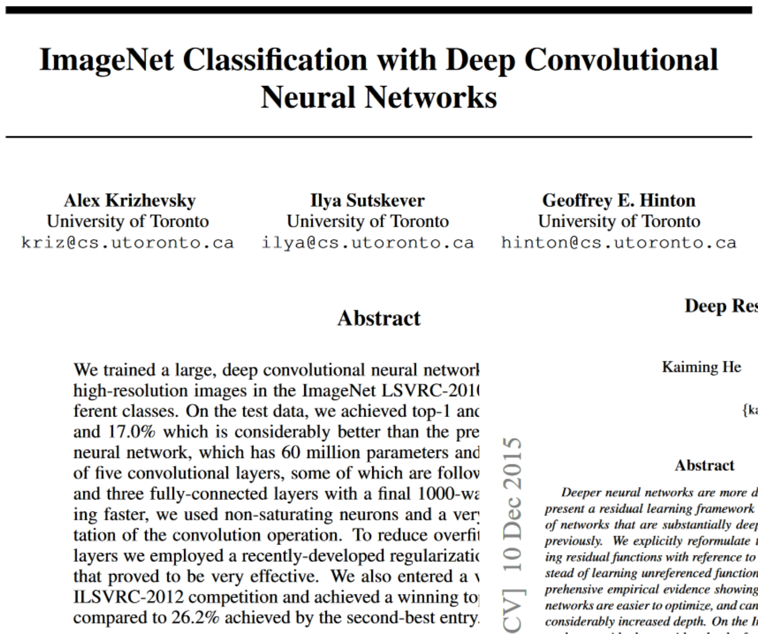
Samuel Kaski

Aalto University School of Science and Technology

UNIVERSITY OF HELSINKI

MANCHESTER 1824 The University of Manchester

Deep neural networks (DNNs) produce SOTA performance in computer vision



ImageNet Classification with Deep Convolutional Neural Networks

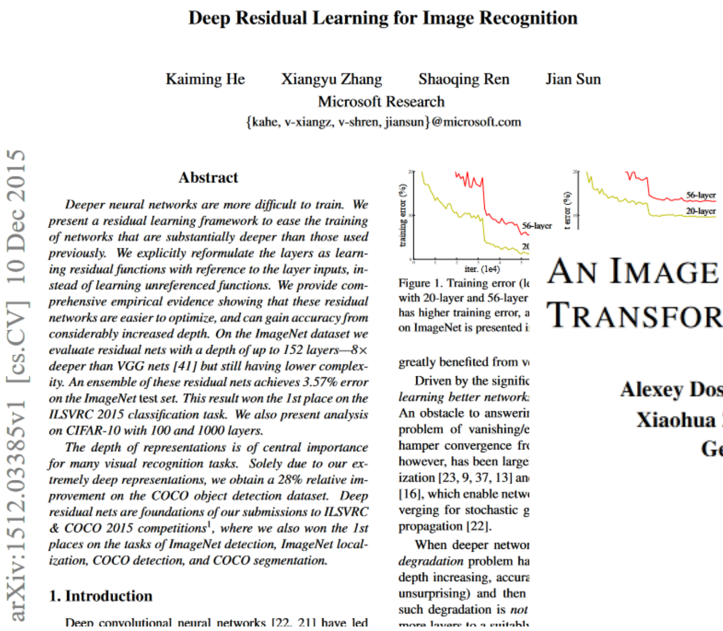
Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

Abstract

We trained a large, deep convolutional neural networ
high-resolution images in the ImageNet LSVRC-201
ferent classes. On the test data, we achieved top-1 an
and 17.0% which is considerably better than the pre
neural network, which has 60 million parameters and
of five convolutional layers, some of which are follo
and three fully-connected layers with a final 1000-wa
ing faster, we used non-saturating neurons and a ver
tation of the convolution operation. To reduce overfi
layers we employed a recently-developed regularizati
that proved to be very effective. We also entered a
ILSVRC-2012 competition and achieved a winning to
compared to 26.2% achieved by the second-best entry.



Deep Residual Learning for Image Recognition

Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun
Microsoft Research
{kahe, v-xiangz, v-shren, jiansun}@microsoft.com

Abstract

Deeper neural networks are more difficult to train. We present a residual learning framework to ease the training of networks that are substantially deeper than those used previously. We explicitly reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. We provide comprehensive empirical evidence showing that these residual networks are easier to optimize, and can gain accuracy from considerably increased depth. On the ImageNet dataset we evaluate residual nets with a depth of up to 152 layers—8× deeper than VGG nets [41] but still having lower complexity. An ensemble of these residual nets achieves 3.57% error on the ImageNet test *set. This result won the 1st place on the ILSVRC 2015 classification task. We also present analysis on CIFAR-10 with 100 and 1000 layers.*

The depth of representations is of central importance for many visual recognition tasks. Solely due to our extremely deep representations, we obtain a 28% relative improvement on the COCO object detection dataset. Deep residual nets are foundations of our submissions to ILSVRC & COCO 2015 competitions[1], *where we also won the 1st places on the tasks of ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation.*

1. Introduction

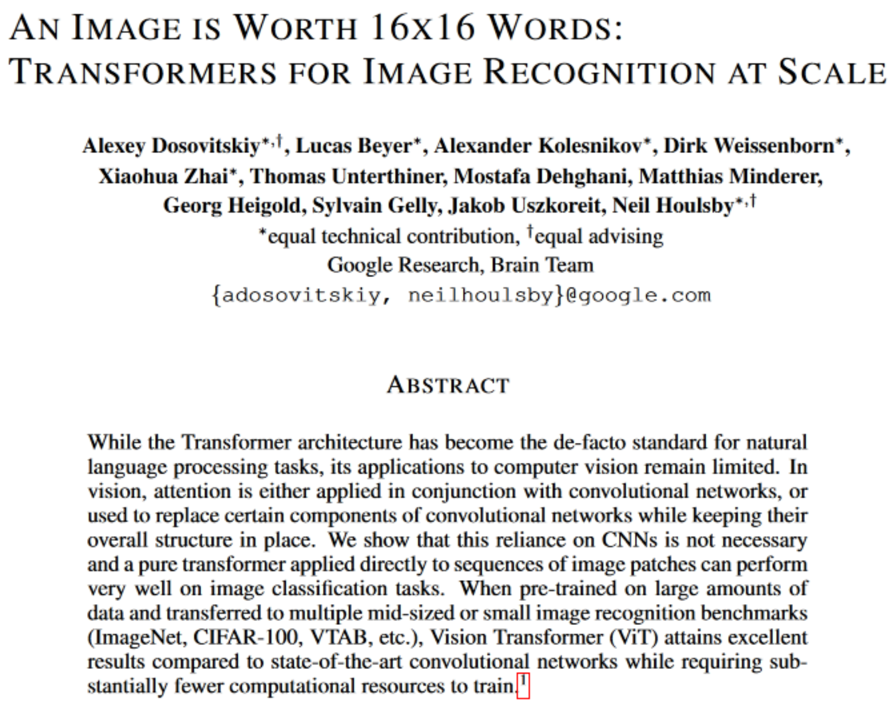
Deep convolutional neural networks [22, 21] have led

Figure 1. Training error (l
with 20-layer and 56-layer
has higher training error, a
on ImageNet is presented i

greatly benefited from v

Driven by the signific
learning better network
An obstacle to answeri
problem of vanishing/e
hamper convergence fr
however, has been large
ization [23, 9, 37, 13] an
[16], which enable netw
verging for stochastic g
propagation [22].

When deeper netwo
degradation problem ha
depth increasing, accur
unsurprising) and then
such degradation is *not*



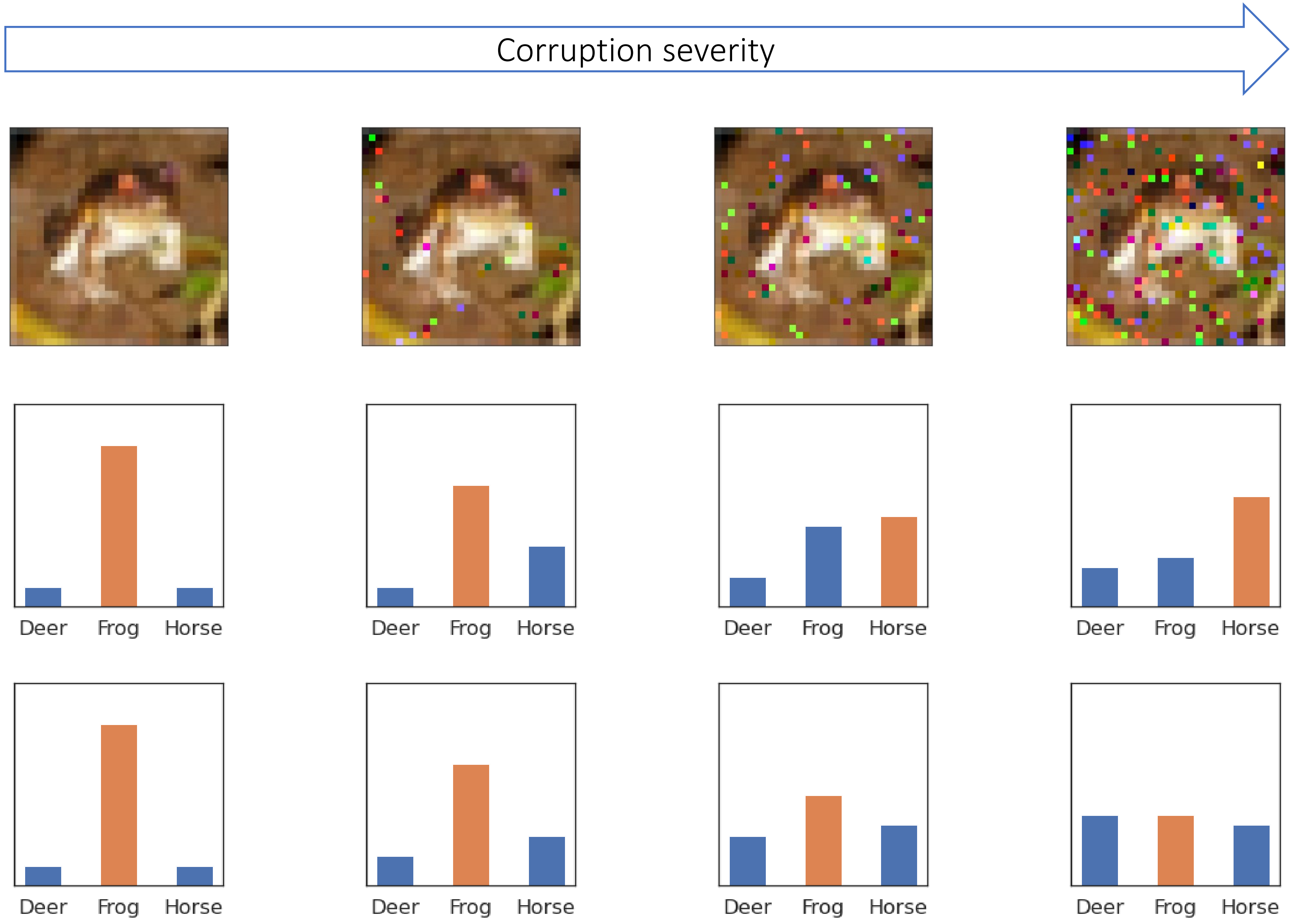
AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE

Alexey Dosovitskiy[*,†], **Lucas Beyer**[*], **Alexander Kolesnikov**[*], **Dirk Weissenborn**[*], **Xiaohua Zhai**[*], **Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby**[*,†]
[*]equal technical contribution, [†]equal advising
Google Research, Brain Team
{adosovitskiy, neilhoulsby}@google.com

ABSTRACT

While the Transformer architecture has become the de-facto standard for natural language processing tasks, its applications to computer vision remain limited. In vision, attention is either applied in conjunction with convolutional networks, or used to replace certain components of convolutional networks while keeping their overall structure in place. We show that this reliance on CNNs is not necessary and a pure transformer applied directly to sequences of image patches can perform very well on image classification tasks. When pre-trained on large amounts of data and transferred to multiple mid-sized or small image recognition benchmarks (ImageNet, CIFAR-100, VTAB, etc.), Vision Transformer (ViT) attains excellent results compared to state-of-the-art convolutional networks while requiring substantially fewer computational resources to train.[1]

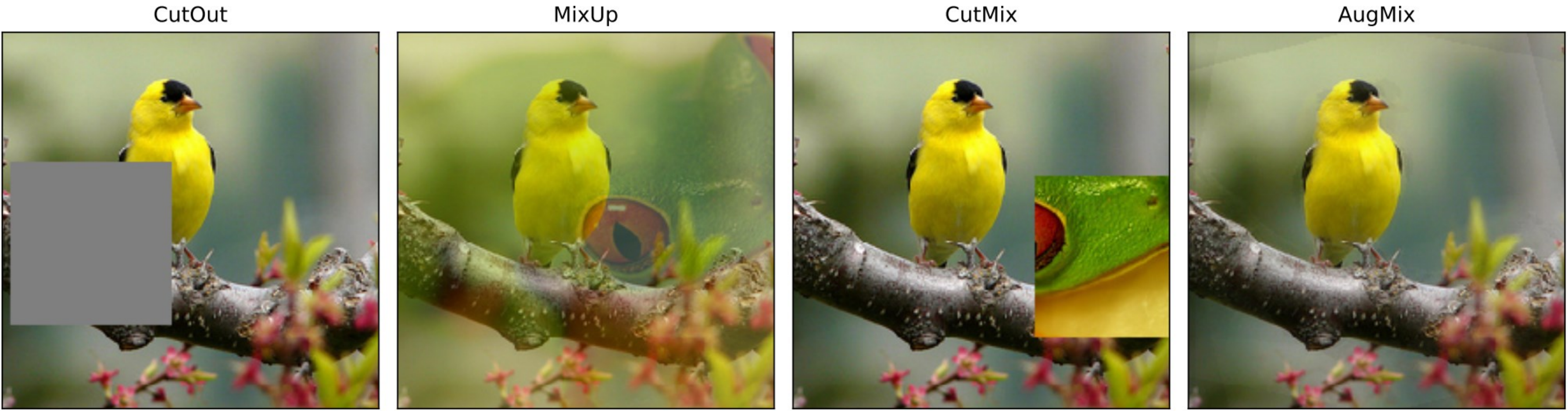
However, they are extremely sensitive to image corruptions



Typical behavior

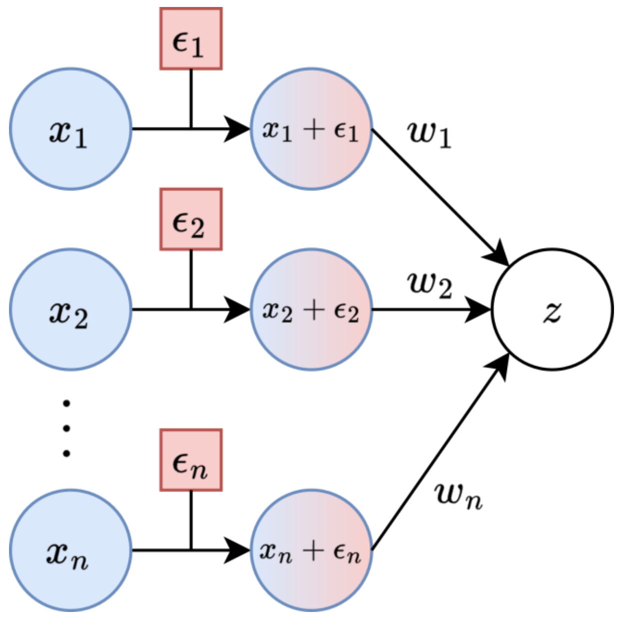
Desirable behavior

Previous works tackle this problem via carefully designed data augmentation techniques for images, thus enriching the training data to promote model robustness to corruptions.

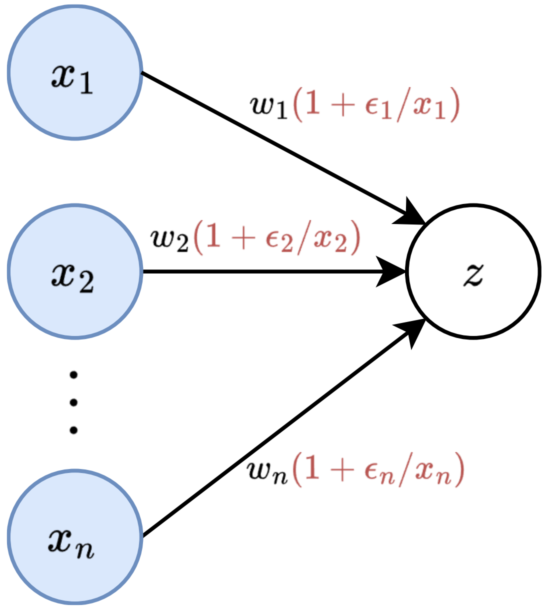


[1] Devries et al. (2017). Improved regularization of convolutional neural networks with Cutout.
[2] Zhang et al. (2017). mixup: Beyond Empirical Risk Minimization
[3] Yun et al. (2019). CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features
[4] Hendrycks et al. (2020). AugMix: A Simple Data Processing Method to Improve Robustness and Uncertainty

In this work, we introduce an alternative and domain-agnostic approach, which is to simulate input corruptions during training using multiplicative perturbations in the weight space.

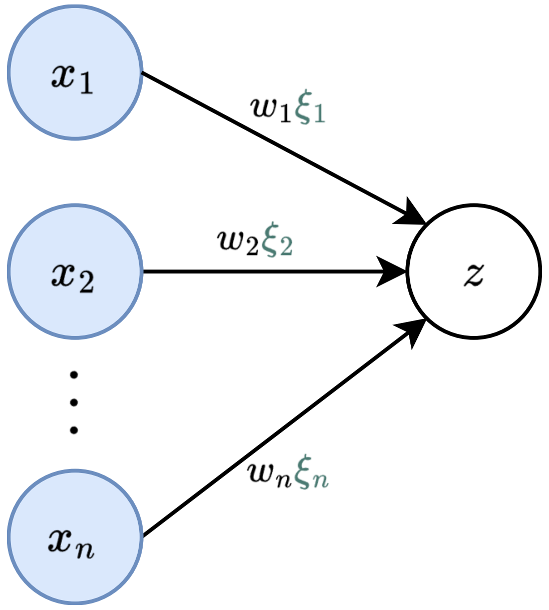


$$z = \mathbf{w}^\top (\mathbf{x} + \boldsymbol{\epsilon})$$



x_1
x_2
x_n
$w_1(1 + \epsilon_1 / x_1)$
$w_2(1 + \epsilon_2 / x_2)$
$w_n(1 + \epsilon_n / x_n)$
z

$$z = (\mathbf{w} \circ (1 + \boldsymbol{\epsilon}) / \mathbf{x})^\top \mathbf{x}$$



$$z = (\mathbf{w} \circ \boldsymbol{\xi})^\top \mathbf{x}, \quad \boldsymbol{\xi} \sim p(\boldsymbol{\xi})$$

Given a dataset $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N \subset \mathcal{X} \times \mathcal{Y}$, a corruption $\mathbf{g} : \mathcal{X} \to \mathcal{X}$,neural network weights $\boldsymbol{\omega} \in \mathcal{W}$, and a loss function $\mathcal{L}$. Under some weak assumptions about $\mathbf{g}$ and $\mathcal{L}$, there exists a multiplicative weight perturbation $\boldsymbol{\xi}_g \in \mathcal{W}$ and a constant $C_{\mathbf{g}} > 0$ such that:

$$\underbrace{\mathcal{L}(\boldsymbol{\omega}; \mathbf{g}(\mathcal{S}))}_{\text{Loss under corruption}} \leq \underbrace{\mathcal{L}(\boldsymbol{\omega} \circ \boldsymbol{\xi}_{\mathbf{g}}; \mathcal{S})}_{\text{Loss under perturbation}} + \underbrace{\frac{C_{\mathbf{g}}}{2} ||\boldsymbol{\omega}||_F^2}_{L_2 \text{ regularization}}$$

where $\mathbf{g}(\mathcal{S}) = \{(\mathbf{g}(\mathbf{x}_i), y_i)\}_{i=1}^N$.

Implication: The multiplicative weight perturbations (MWPs) simulate input corruptions during training, making the model robust to these simulated corruptions, which could also improve its robustness to real world corruptions.

Proposed method: Data augmentation via Multiplicative Perturbations (DAMP)

DAMP is an efficient training method that perturbs weights using multiplicative Gaussian random variable during training by minimizing the following loss function:

$$\mathcal{L}_{\text{DAMP}}(\boldsymbol{\omega}; \mathcal{S}) := \mathbb{E}_{\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{1}, \sigma^2 \mathbf{I})} \left[\mathcal{L}(\boldsymbol{\omega} \circ \boldsymbol{\xi}; \mathcal{S}) \right] + \frac{\lambda}{2} ||\boldsymbol{\omega}||_F^2$$

To efficiently estimate the expectation in the loss function, DAMP:

1. splits the training batch evenly into M sub-batches,
2. samples a weight perturbation for each sub-batch to calculate the sub-batch gradient,
3. averages over all sub-batch gradients to obtain the final gradient.

Therefore, DAMP is suitable for data parallelism in multi-GPU training.

Adaptive Sharpness-Aware Minimization optimizes deep neural networks under adversarial multiplicative weight perturbations

DAMP minimizes the expected loss under Gaussian MWPs:

$$\mathcal{L}_{\mathrm{DAMP}}(\boldsymbol{\omega}; \mathcal{S}) := \mathbb{E}_{\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{1}, \sigma^2 \mathbf{I})} \left[\mathcal{L}(\boldsymbol{\omega} \circ \boldsymbol{\xi}; \mathcal{S})\right] + \frac{\lambda}{2} ||\boldsymbol{\omega}||_F^2$$

Alternatively, we could minimize the loss under adversarial MWPs:

$$\mathcal{L}_{\mathrm{ASAM}}(\boldsymbol{\omega}; \mathcal{S}) := \max_{||\boldsymbol{\xi}||_2 \leq \rho} \mathcal{L}(\boldsymbol{\omega} \circ (\mathbf{1} + \boldsymbol{\xi}); \mathcal{S}) + \frac{\lambda}{2} ||\boldsymbol{\omega}||_F^2$$

Interestingly, by solving the inner maximization problem using the first-order Taylor expansion, we arrive at the same update rule of Adaptive Sharpness-aware minimization (ASAM)[1]

[1] Kwon et al. (2021). ASAM: Adaptive Sharpness-Aware Minimization for Scale-Invariant Learning of Deep Neural Networks.
[2] Foret et al. (2021). Sharpness-Aware Minimization for Efficiently Improving Generalization.

Adaptive Sharpness-Aware Minimization optimizes deep neural networks under adversarial multiplicative weight perturbations

Thus, ASAM optimizes DNNs under adversarial multiplicative weight perturbations, as opposed to its predecessor Sharpness-aware Minimization (SAM)[2] which optimizes DNNs under adversarial additive weight perturbations.
➔ This explains why ASAM leads to better corruption robustness than SAM, as shown in our experiment results. However, each iteration of ASAM and SAM takes twice as long as that of DAMP.

[1] Kwon et al. (2021). ASAM: Adaptive Sharpness-Aware Minimization for Scale-Invariant Learning of Deep Neural Networks.
[2] Foret et al. (2021). Sharpness-Aware Minimization for Efficiently Improving Generalization.

Experiment results: ResNet50 / ImageNet

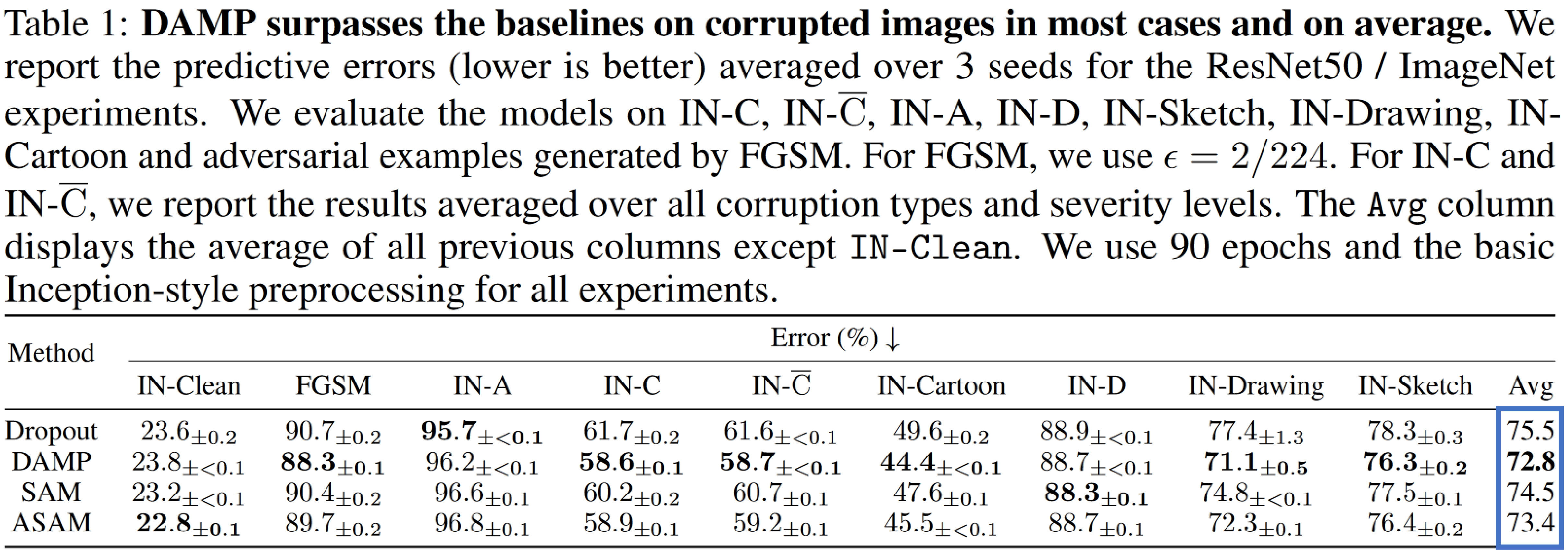


Table 1: **DAMP surpasses the baselines on corrupted images in most cases and on average.** We report the predictive errors (lower is better) averaged over 3 seeds for the ResNet50 / ImageNet experiments. We evaluate the models on IN-C, IN-$\overline{\text{C}}$, IN-A, IN-D, IN-Sketch, IN-Drawing, IN-Cartoon and adversarial examples generated by FGSM. For FGSM, we use $\epsilon = 2/224$. For IN-C and IN-$\overline{\text{C}}$, we report the results averaged over all corruption types and severity levels. The `Avg` column displays the average of all previous columns except `IN-Clean`. We use 90 epochs and the basic Inception-style preprocessing for all experiments.

Method	Error (%) ↓								
	IN-Clean	FGSM	IN-A	IN-C	IN-$\overline{\text{C}}$	IN-Cartoon	IN-D	IN-Drawing	IN-Sketch	Avg
Dropout	$23.6_{\pm 0.2}$	$90.7_{\pm 0.2}$	$\mathbf{95.7}_{\pm <0.1}$	$61.7_{\pm 0.2}$	$61.6_{\pm <0.1}$	$49.6_{\pm 0.2}$	$88.9_{\pm <0.1}$	$77.4_{\pm 1.3}$	$78.3_{\pm 0.3}$	75.5
DAMP	$23.8_{\pm <0.1}$	$\mathbf{88.3}_{\pm 0.1}$	$96.2_{\pm <0.1}$	$\mathbf{58.6}_{\pm 0.1}$	$\mathbf{58.7}_{\pm <0.1}$	$\mathbf{44.4}_{\pm <0.1}$	$88.7_{\pm <0.1}$	$\mathbf{71.1}_{\pm 0.5}$	$\mathbf{76.3}_{\pm 0.2}$	**72.8**
SAM	$23.2_{\pm <0.1}$	$90.4_{\pm 0.2}$	$96.6_{\pm 0.1}$	$60.2_{\pm 0.2}$	$60.7_{\pm 0.1}$	$47.6_{\pm 0.1}$	$\mathbf{88.3}_{\pm 0.1}$	$74.8_{\pm <0.1}$	$77.5_{\pm 0.1}$	74.5
ASAM	$\mathbf{22.8}_{\pm 0.1}$	$89.7_{\pm 0.2}$	$96.8_{\pm 0.1}$	$58.9_{\pm 0.1}$	$59.2_{\pm 0.1}$	$45.5_{\pm <0.1}$	$88.7_{\pm 0.1}$	$72.3_{\pm 0.1}$	$76.4_{\pm 0.2}$	73.4

Experiment results: ViT / ImageNet / Basic Augmentations

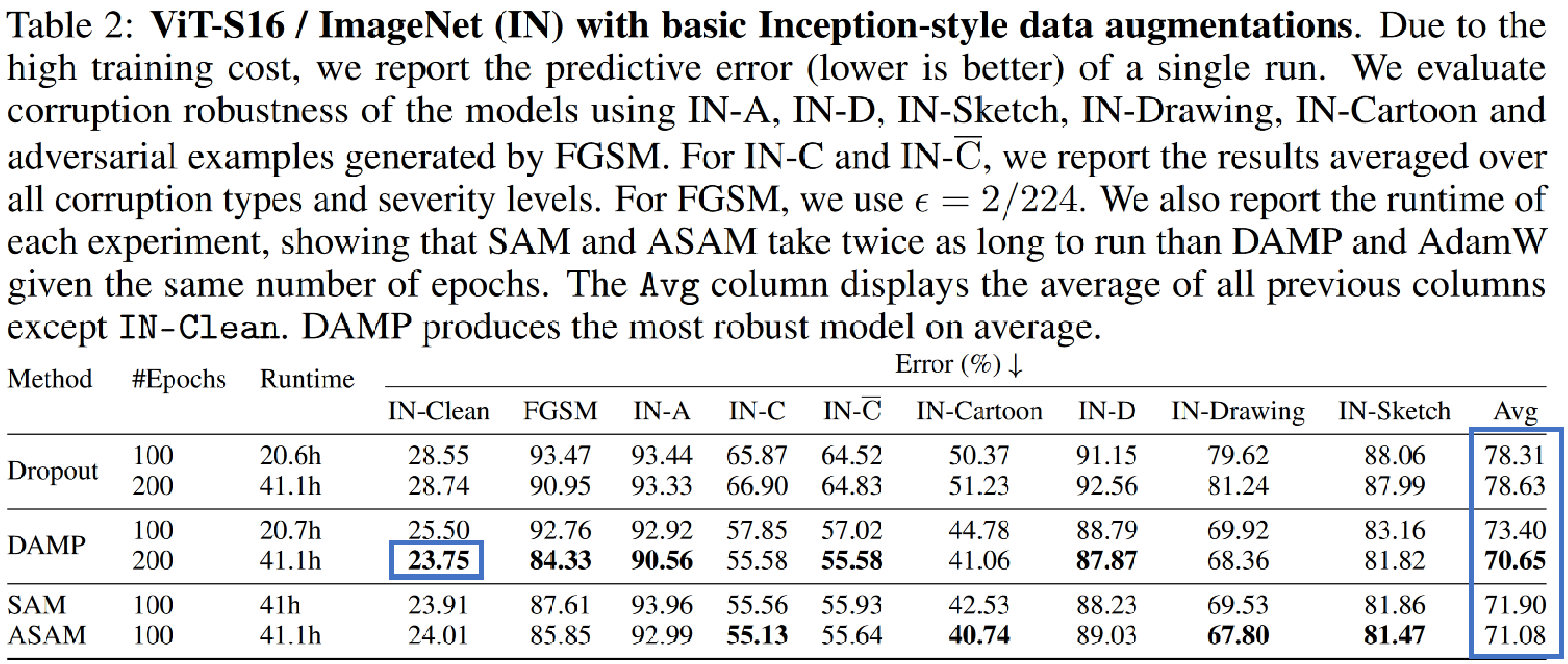


Table 2: **ViT-S16 / ImageNet (IN) with basic Inception-style data augmentations**. Due to the high training cost, we report the predictive error (lower is better) of a single run. We evaluate corruption robustness of the models using IN-A, IN-D, IN-Sketch, IN-Drawing, IN-Cartoon and adversarial examples generated by FGSM. For IN-C and IN-$\overline{\text{C}}$, we report the results averaged over all corruption types and severity levels. For FGSM, we use $\epsilon = 2/224$. We also report the runtime of each experiment, showing that SAM and ASAM take twice as long to run than DAMP and AdamW given the same number of epochs. The `Avg` column displays the average of all previous columns except `IN-Clean`. DAMP produces the most robust model on average.

Method	#Epochs	Runtime	Error (%) ↓									
			IN-Clean	FGSM	IN-A	IN-C	IN-$\overline{\text{C}}$	IN-Cartoon	IN-D	IN-Drawing	IN-Sketch	Avg
Dropout	100	20.6h	28.55	93.47	93.44	65.87	64.52	50.37	91.15	79.62	88.06	78.31
	200	41.1h	28.74	90.95	93.33	66.90	64.83	51.23	92.56	81.24	87.99	78.63
DAMP	100	20.7h	25.50	92.76	92.92	57.85	57.02	44.78	88.79	69.92	83.16	73.40
	200	41.1h	**23.75**	**84.33**	**90.56**	55.58	**55.58**	41.06	**87.87**	68.36	81.82	**70.65**
SAM	100	41h	23.91	87.61	93.96	55.56	55.93	42.53	88.23	69.53	81.86	71.90
ASAM	100	41.1h	24.01	85.85	92.99	**55.13**	55.64	**40.74**	89.03	**67.80**	**81.47**	71.08

Experiment results: ViT / ImageNet / Advanced Augmentations

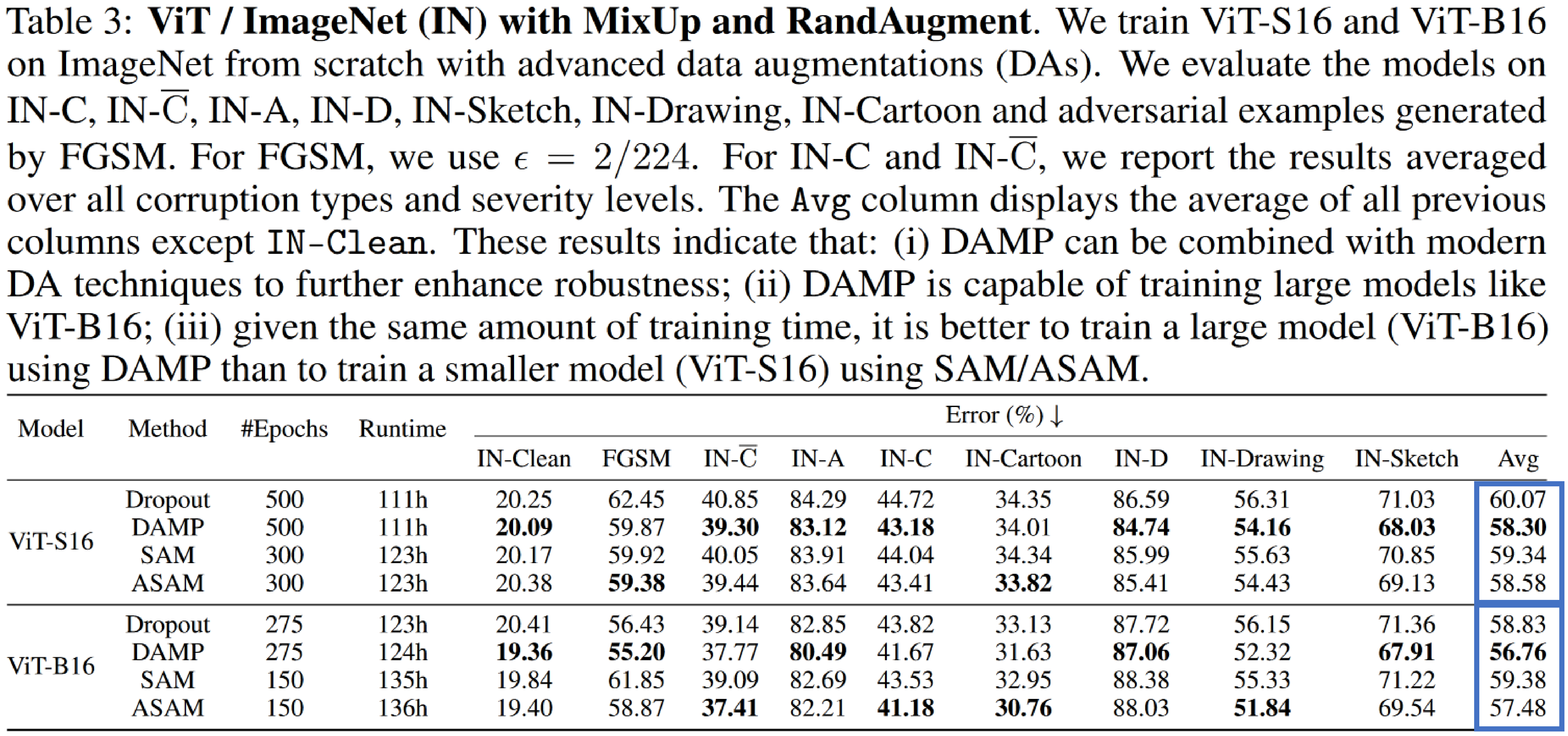


Table 3: **ViT / ImageNet (IN) with MixUp and RandAugment**. We train ViT-S16 and ViT-B16 on ImageNet from scratch with advanced data augmentations (DAs). We evaluate the models on IN-C, IN-$\overline{\text{C}}$, IN-A, IN-D, IN-Sketch, IN-Drawing, IN-Cartoon and adversarial examples generated by FGSM. For FGSM, we use $\epsilon = 2/224$. For IN-C and IN-$\overline{\text{C}}$, we report the results averaged over all corruption types and severity levels. The `Avg` column displays the average of all previous columns except `IN-Clean`. These results indicate that: (i) DAMP can be combined with modern DA techniques to further enhance robustness; (ii) DAMP is capable of training large models like ViT-B16; (iii) given the same amount of training time, it is better to train a large model (ViT-B16) using DAMP than to train a smaller model (ViT-S16) using SAM/ASAM.

Model	Method	#Epochs	Runtime	Error (%) ↓									
				IN-Clean	FGSM	IN-$\overline{\text{C}}$	IN-A	IN-C	IN-Cartoon	IN-D	IN-Drawing	IN-Sketch	Avg
ViT-S16	Dropout	500	111h	20.25	62.45	40.85	84.29	44.72	34.35	86.59	56.31	71.03	60.07
	DAMP	500	111h	**20.09**	59.87	**39.30**	**83.12**	**43.18**	34.01	**84.74**	**54.16**	**68.03**	**58.30**
	SAM	300	123h	20.17	59.92	40.05	83.91	44.04	34.34	85.99	55.63	70.85	59.34
	ASAM	300	123h	20.38	**59.38**	39.44	83.64	43.41	**33.82**	85.41	54.43	69.13	58.58
ViT-B16	Dropout	275	123h	20.41	56.43	39.14	82.85	43.82	33.13	87.72	56.15	71.36	58.83
	DAMP	275	124h	**19.36**	**55.20**	37.77	**80.49**	41.67	31.63	**87.06**	52.32	**67.91**	**56.76**
	SAM	150	135h	19.84	61.85	39.09	82.69	43.53	32.95	88.38	55.33	71.22	59.38
	ASAM	150	136h	19.40	58.87	**37.41**	82.21	**41.18**	**30.76**	88.03	**51.84**	69.54	57.48

Conclusion

- Multiplicative weight perturbations (MWPs) improve robustness of DNNs to a wide range of input corruptions.
- We thus introduce DAMP, a simple algorithm which perturbs weights using Gaussian MWPs during training while having the same cost as standard SGD.
- We also show that ASAM can be viewed as optimizing DNNs under adversarial MWPs.
- Our experiments show that DAMP improves corruption robustness of different architectures (ResNet, ViT), and can be combined with modern augmentations (MixUp, RandAugment) to further boost robustness.
- As DAMP is domain-agnostic, one future direction is to investigate its effectiveness in other domains (NLP, RL).

For more information, visit the website:

https://trungtrinh44.github.io/DAMP/