Semantic Perturbations with Normalizing Flows for Improved Generalization

Oğuz Kaan Yüksel [1] Sebastian U. Stich [1] Martin Jaggi [1] Tatjana Chavdarova [1]

Abstract

Several methods from two separate lines of works, namely, data augmentation (DA) and adversarial training techniques, rely on perturbations done in *latent* space. Often, these methods are either non-interpretable due to their non-invertibility or are notoriously difficult to train due to their numerous hyperparameters. We exploit the exactly reversible encoder-decoder structure of normalizing flows to perform perturbations in the latent space. We demonstrate that these on-manifold perturbations match the performance of advanced DA techniques—reaching 96.6% test accuracy for CIFAR-10 using ResNet-18 and outperform existing methods particularly in low data regimes—yielding 10–25$\%$ relative improvement of test accuracy from classical training. We find our latent *adversarial* perturbations, adaptive to the classifier throughout its training, are most effective.

1. Introduction

Successfully applying Deep Neural Networks (DNNs) in real world tasks in large part depends on the availability of large annotated datasets for the task at hand. Thus, besides several overfitting techniques, *data-augmentation* (DA) often remains a "mandatory" step when deploying DNNs in practice. Traditional DA techniques consist of applying a predefined set of transformations to the training samples that do not change the class label. As this approach is limited to making the classifier robust *solely* to the fixed set of hard-coded transformations, advanced methods incorporate more loosely defined transformations in the data space (Zhang et al., 2018a; DeVries & Taylor, 2017; Yun et al., 2019). Furthermore, recently proposed DA techniques exploit the *latent space* to perform such transformations (Antoniou et al., 2017; Zhao et al., 2018), while typically solving the model's non-invertability by training a *separate* model (Zhao et al., 2018), thus making them hard to train.

A separate line of work focuses on *adversarial training* (see (Biggio & Roli, 2018) and references therein), where the final model is trained with samples perturbed in a way that makes its prediction incorrect, called *adversarial* samples. However, further empirical studies showed that such training reduces the "clean" samples accuracy, indicating the two objectives are competing (Tsipras et al., 2019b; Su et al., 2018). Stutz et al. (2019) postulate that this robustness-generalization trade-off appears due to using off-manifold adversarial attacks that leave the data-manifold, and that 'on-manifold adversarial attacks' can improve generalization. Thus, the authors proposed to use perturbations in the latent space of a generative model, VAE-GAN (Larsen et al., 2016; Rosca et al., 2017). However, as this method relies on the VAE-GAN model which is particularly hard to train—since in addition to GAN training it involves hard to tune hyperparamaters balancing the VAE and GAN objectives—its usage remained limited.

Motivated by the advantages of *normalizing flows* (NF) relevant to these two lines of works, we employ NFs (*e.g. Glow,* Kingma & Dhariwal, 2018), to define entirely unsupervised augmentations—contrasting with pre-defined fixed transformations—with the same goal of improving the generalization of deep classifiers. In particular, NF models offer appealing advantages for latent space perturbations, such as: (i) exact latent-variable inference and log-likelihood evaluation, and (ii) efficient inference and synthesis that can be parallelized (Kingma & Dhariwal, 2018).

Related works. Several works learn useful DA policies, for instance by optimization (Fawzi et al., 2016; Ratner et al., 2017), Reinforcement Learning techniques (Cubuk et al., 2019; 2020; Zhang et al., 2020b), specifically trained generator networks (Peng et al., 2018) or assisted by generative adversarial networks (Perez & Wang, 2017; Antoniou et al., 2017; Zhang et al., 2018b; Tran et al., 2020). Several methods traverse the latent space to find virtual data samples that are missclassified (Baluja & Fischer, 2017; Song et al., 2018; Xiao et al., 2018; Zhang et al., 2020a). Complementary, the connection of adversarial learning and generalization has also been studied in (Tanay & Griffin, 2016; Rozsa et al., 2016; Jalal et al., 2017; Tsipras et al., 2019a; Gilmer et al., 2018; Zhao et al., 2018).

[1]EPFL, Lausanne, Switzerland. Correspondence to: Oğuz Kaan Yüksel <oguz.yuksel@epfl.ch>.

Third workshop on *Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models* (ICML 2021). Copyright 2021 by the author(s).

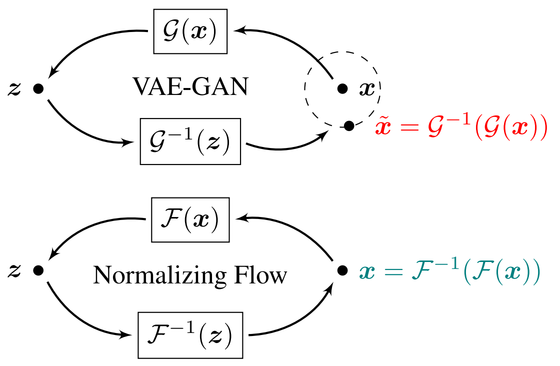
$\mathcal{G}(\boldsymbol{x})$
$\boldsymbol{z}$
VAE-GAN
$\boldsymbol{x}$
$\mathcal{G}^{-1}(\boldsymbol{z})$
$\tilde{\boldsymbol{x}} = \mathcal{G}^{-1}(\mathcal{G}(\boldsymbol{x}))$
$\mathcal{F}(\boldsymbol{x})$
$\boldsymbol{z}$
Normalizing Flow
$\boldsymbol{x} = \mathcal{F}^{-1}(\mathcal{F}(\boldsymbol{x}))$
$\mathcal{F}^{-1}(\boldsymbol{z})$

Figure 1. **Exactness of NF encoding-decoding**. Here $\mathcal{F}$ denotes the bijective NF model, and $\mathcal{G}/\mathcal{G}^{-1}$ encoder/decoder pair of inexact methods such as VAE or VAE-GAN which, due to inherent decoder noise, is only approximately bijective.

Overview of contributions. We exploit the *exactly reversible* encoder-decoder structure of NFs to perform efficient and controllable augmentations in the learned on-manifold space. We demonstrate that our on-manifold perturbations consistently outperform the standard training on CIFAR-10/100 using ResNet-18. Moreover, in a low-data regime, such training yields up to 25% relative improvement from classical training, of which—as most effective—we find the adversarial perturbations that are adaptive to the classifier throughout its training, see §3.

2. Perturbations in Latent Space

The invertibility of normalizing flows enables bidirectional transitions between image and latent space, see Figure 1. This in turn allows for applying perturbations directly in the latent space rather than image space. We denote a trained NF model, mapping data manifold $\mathcal{X}$ to latent space $\mathcal{Z}$ as, $\mathcal{F}: \mathcal{X} \to \mathcal{Z}$. Given a perturbation function $\mathcal{P}: \mathcal{Z} \to \mathcal{Z}$, defined over the latent space, we define its counterpart in image space as $\mathcal{F}^{-1}(\mathcal{P}(\mathcal{F}(\boldsymbol{x})))$.

Our goal is to define latent perturbation function $\mathcal{P}(\cdot)$ such that we obtain identity-preserving semantic modifications over the original image $\boldsymbol{x}$ in the image domain. To this end, we limit the structure of possible $\mathcal{P}$ in two-ways. Firstly, we directly consider incremental perturbations of the form $\boldsymbol{z} + \mathcal{P}(\boldsymbol{z})$. Secondly, we use an ϵ parameter to control the size of perturbation allowed. More precisely, we have:

$$\mathcal{F}^{-1}\big(\mathcal{F}(\boldsymbol{x}) + \mathcal{P}(\mathcal{F}(\boldsymbol{x}), \epsilon)\big) .$$

For brevity, we refer to $\mathcal{P}$ as "latent attacks" (LA) and we consider two variants, described below.

2.1. Randomized Latent Attacks

At training time, given a datapoint $\boldsymbol{x}_i$, with $1 \leq i \leq n$, using trained normalizing flow we obtain its corresponding latent code $\boldsymbol{z}_i = \mathcal{F}(\boldsymbol{x}_i)$.

Primarily, as perturbation function we consider a simplistic Gaussian noise in the latent space:

$$\mathcal{P}_{rand}(\cdot, \epsilon) = \epsilon \cdot \mathcal{N}(0, \mathbf{I}) , \qquad \textbf{(Randomized–LA)}$$

which is independent from $\boldsymbol{z}_i$. Any such distribution around the original $\boldsymbol{z}_i$ is equivalent to sampling from the learned manifold. In this case, the normalizing flow *pushes forward* this simple Gaussian distribution centered around $\boldsymbol{z}_i$ to a distribution on the image space around $\boldsymbol{x}_i = \mathcal{F}^{-1}(\boldsymbol{z}_i)$. Thus, sampling from the simple prior distribution $\mathcal{N}(0, \mathbf{I})$ is equivalent to sampling from a complex conditional distribution around the original image over the data manifold.

We also define norm truncated versions as follows:

$$\mathcal{P}^{\ell_p}_{rand}(\cdot, \epsilon) = \Pi(\epsilon \cdot \mathcal{N}(0, \mathbf{I})) ,$$

where ℓ_p denotes the selected norm, e.g. ℓ_2 or ℓ_∞. For ℓ_2 norm, Π is defined as ℓ_2 norm scaling, and for ℓ_∞, Π is the component-wise clipping operation defined below:

$$(\Pi(\boldsymbol{x}))_i := \max(-\epsilon, \min(+\epsilon, \boldsymbol{x}_i)) .$$

2.2. Adversarial Latent Attacks

Analogous to the above randomized latent attacks, at train time, given a datapoint $\boldsymbol{x}_i$ and it's associated label l_i, with $1 \leq i \leq n$, using trained normalizing flow we obtain its corresponding latent code $\boldsymbol{z}_i = \mathcal{F}(\boldsymbol{x}_i)$.

We search for $\Delta_{\boldsymbol{z}_i} \in \mathcal{Z}$ such that the loss obtained of the generated image $\tilde{\boldsymbol{x}}_i = \mathcal{F}^{-1}(\boldsymbol{z}_i + \Delta_{\boldsymbol{z}_i})$ is maximal:

$$\Delta^\star_{\boldsymbol{z}_i} = \underset{\|\Delta_{\boldsymbol{z}_i}\|_{\ell_p} \leq \epsilon}{\arg\max}\ \mathcal{L}_\theta(\mathcal{F}^{-1}(\boldsymbol{z}_i + \Delta_{\boldsymbol{z}_i}), l_i) ,$$
$$\mathcal{P}^{\ell_p}_{adv}(\boldsymbol{z}_i, \epsilon) = \Delta^\star_{\boldsymbol{z}_i} , \qquad \textbf{(Adversarial–LA)}$$

where $\mathcal{L}_\theta$ is the loss function of the classifier, and ℓ_p denotes the selected norm, *e.g.* ℓ_2 or ℓ_∞.

In practice, we define the number of steps k and the step size α to optimize for $\Delta^\star_{\boldsymbol{z}_i} \in \mathcal{Z}$ (as in Stutz et al. (2019); Wong & Kolter (2021)), and we have the following procedure:

- Initialize a random $\Delta^0_{\boldsymbol{z}_i}$ with $\|\Delta^0_{\boldsymbol{z}_i}\|_{\ell_p} \leq \epsilon$.
- Iteratively update $\Delta^j_{\boldsymbol{z}_i}$ for $j = 1, \ldots, k$ number of steps with step size α as follows:

$$\Delta^j_{\boldsymbol{z}_i} = \Pi\Big(\Delta^{j-1}_{\boldsymbol{z}_i} + \alpha \cdot \frac{\nabla \mathcal{L}_\theta(\mathcal{F}^{-1}(\boldsymbol{z}_i + \Delta^{j-1}_{\boldsymbol{z}_i}), l_i)}{\|\nabla \mathcal{L}_\theta(\mathcal{F}^{-1}(\boldsymbol{z}_i + \Delta^{j-1}_{\boldsymbol{z}_i}), l_i)\|_{\ell_p}}\Big)$$

  where Π is the projection operator that ensures $\|\Delta^j_{\boldsymbol{z}_i}\|_{\ell_p} \leq \epsilon$ and the gradient is with respect to $\Delta^j_{\boldsymbol{z}_i}$.
- Output $\mathcal{P}_{adv}(\boldsymbol{z}_i, \epsilon) = \Delta^k_{\boldsymbol{z}_i}$

Table 1. Test accuracy (%) on **CIFAR-10**, in the *low-data regime* compared to the *full train set*. For the former, we train generative models and classifiers on the same 5% of the training set and evaluate classifiers on the full test set. In addition to standard training, we consider training with commonly used data augmentations (DA) in the image space, which includes rotation and horizontal flips (Zagoruyko & Komodakis, 2016), as well as more recent *Cutout* (DeVries & Taylor, 2017) and *Mixup* (Zhang et al., 2018a) methods. See §3.1 for a discussion.

Method	**Low-data**	**Full-set**
Standard (no DA)	49.8	89.7
Standard + common DA	64.1	95.2
VAE-GAN (Stutz et al., 2019)	58.9	94.2
Cutout (DeVries & Taylor, 2017)	66.8	96.0
Mixup (Zhang et al., 2018a)	73.4	95.9
Randomized–LA (ours)	70.1	96.3
Adversarial–LA (ours)	**80.4**	**96.6**

For the case of ℓ_∞, we replace normalization of gradient with $sign(\cdot)$ operator, i.e.:

$$\Delta_{\boldsymbol{z}_i}^{j} = \Pi\Big(\Delta_{\boldsymbol{z}_i}^{j-1} + \alpha \cdot sign\big(\nabla\mathcal{L}_\theta(\mathcal{F}^{-1}(\boldsymbol{z}_i + \Delta_{\boldsymbol{z}_i}^{j-1}), l_i)\big)\Big)$$

and use component-wise clipping for projection, equivalent to the standard ℓ_∞ PGD adversarial attack of Madry et al..

Similarly, as NFs directly models the underlying data manifold, this perturbation is equivalent to a search over the *on-manifold* adversarial samples (Stutz et al., 2019).

3. Experiments

3.1. Generalization on CIFAR-10

We are primarily interested in the performance of our perturbations in the low-data regime, when using only a small subset of CIFAR-10 as the training set. We train ResNet-18 classifiers on only 5% percent of the full training set and evaluate models on the full test set.

We compare our methods with some of the most commonly used data augmentations methods such as *Cutout* (DeVries & Taylor, 2017) and *Mixup* (Zhang et al., 2018a), as well as with the VAE-GAN based approach (Stutz et al., 2019). For (Stutz et al., 2019), we use the authors' implementation.

For (DeVries & Taylor, 2017), we report the best test accuracy observed among a grid search on the learning rate $\eta \in \{0.1, 0.01\}$. Similarly, for (Zhang et al., 2018a), we report the best accuracy among grid search on learning rate $\eta \in \{0.1, 0.01\}$ and mixup coefficient $\lambda \in \{.1, .2, .3, .4, 1.0\}$. For Randomized–LA, we use $\ell = \ell_\infty, \epsilon = 0.25$, and for Adversarial–LA, we use $\ell = 2, \epsilon = 1.0, \alpha = 0.5, k = 3$.

Table 1 summarizes our generalization experiments in the low data regime—using only 5% of CIFAR-10 for training, compared to the full CIFAR-10 training set. Both Randomized–LA and Adversarial–LA notably outperform the standard training baseline. In particular, we observe that (i) our simplistic Randomized–LA method already outperforms some recent strong data augmentation methods, and (ii) Adversarial–LA achieves *best* test accuracy for both low-data and full-set regimes. See §3.3 below for additional benchmarks with VAE-GAN (Stutz et al., 2019) and §B.3 for additional results with different attack parameters.

Table 2. **Cross-datasets experiments.** *Test accuracy* (%) on **CIFAR-100**, in the *low-data regime*, where we use 10% of the training set and the full test set. The normalizing flow used to generate training samples is trained on **CIFAR-10**.

Method	**Test**	**Improvement**
Standard	36.4	–
Randomized–LA, ℓ=ℓ_∞, ϵ=.2	39.7	+3.3
Randomized–LA, ℓ=ℓ_∞, $\epsilon = .3$	41.0	+4.6
Randomized–LA, ℓ=ℓ_2, $\epsilon = 10$	40.4	+4.0
Randomized–LA, ℓ=ℓ_2, $\epsilon = 20$	**42.3**	**+5.9**
Adversarial–LA, ℓ=ℓ_2, α=.5, k=3	**45.0**	**+8.6**

3.2. Cross-Dataset Experiments

To further analyze potential applications of our NF based latent attacks on real-world use cases, we conduct the following experiment. Assuming we have available a relevant large-scale dataset, a question arises if the NF within our approach can be pre-trained on it, and used for training the classifier on a different dataset. In particular, we use CIFAR-10 to train the NF model, and then our latent attacks to train a classifier on 10% of the CIFAR-100 dataset.

Table 2 shows our results for a selection of latent attacks. Randomized–LA and Adversarial–LA achieve 16% and 24% percent improvements over the standard baseline, respectively. The results indicate that NFs are capable of transferring useful augmentations across datasets. See §B.2 for an additional results with SVHN and CIFAR-10.

3.3. Additional Comparison with VAE-GAN

We study the performance of our latent perturbation-based training strategies in varying settings, starting from low-data regime to full-set. We reproduce the classifier and the hyperparameter setup used in (Stutz et al., 2019), and use analogous setup for our method. For the reported VAE-GAN results, we used the source code provided by the authors[1]. For our Randomized–LA, we use perturbations of size ϵ=0.15 and for Adversarial–LA, we use ϵ=0.05, α=0.01 and number of steps $k = 10$.

[1]https://github.com/davidstutz/disentangling-robustness-generalization

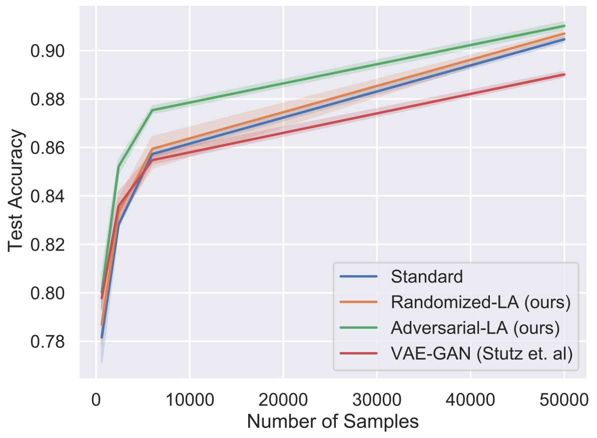

Figure 2. Test accuracy (y-axis)—on full test set, for a varying number of *training* samples (x-axis), on **FashionMNIST**. To replicate the setup of VAE-GAN (Stutz et al., 2019), only a portion of the dataset (x-axis) is used to train the classifier, while the corresponding generative model is trained on the full dataset. We run each experiment with three different random seeds, and report the mean and standard deviation of the test accuracy. See §3.3.

Figure 2 shows our average results for 3 runs with training sizes in $\{600, 2400, 6000, 50000\}$. We observe that Randomized–LA performs comparatively to the standard training baseline, whereas Adversarial–LA outperforms the standard baseline across all train set sizes. Note that the difference to the standard baselines shrinks as we increase the number of samples available to the classifiers.

Inline with our results, Stutz et al. (2019) report diminishing performance gains for increasingly challenging datasets such as FashionMNIST to CelebA, when using therein VAE-GAN based approach. One potential cause could be the *approximate* encoding and decoding mappings, and/or sensitivity to hyper-parameter tuning. Indeed, our results support the numerous appealing advantages of NF models for latent space perturbations, and indicate that they have better capacity to produce useful augmented training samples.

4. Discussion

Exact Coding. As noted in §2, normalizing flows can perform exact encoding and decoding by their construction. That is, the decoding operation is exactly the reverse of the encoding operation. Any continuous encoder maps a neighborhood of a sample to some neighborhood of its latent representation. However, the invertibility of normalizing flows also maps *any neighborhood of latent code to a neighborhood of the original sample*. This property has significant advantages over any other approximate invertible encoder-decoder methods including VAE-GANs, for defining perturbations in latent space.

Increasing Effective Dataset Size. The primary advantage of exact coding is that the generated samples via latent perturbations improve the generalization performance of classifiers, as we show in §3.1. To understand why this occurs, consider the limit case $\epsilon \to 0$ for a latent perturbation. For a numerically stable NF, this implies that we recover the original data samples, hence the original data manifold. As we increase the ϵ, we "enlarge" our manifold simultaneously from all data samples. Thus, by increasing ϵ, we add further plausible data points to our training set as long the learned latent representation is a good approximation of the underlying data manifold. This does not necessarily hold for approximate mappings due to inherent *decoder noise*.

Controllability. In §2, we introduced two variants of latent perturbations with normalizing flows. These two variants define different local objectives around the latent code of the original sample. The Randomized–LA defines a sampling operation on the data manifold, whereas the Adversarial–LA defines a stochastic search procedure to find samples attaining high classifier losses. Here, we exploit the diffeomorphism provided by normalizing flow to efficiently map a complex sampling operation—sampling from data manifold, or a complex search operation—finding on-manifold adversarial samples, to the latent space. Combined with simple prior structure of normalizing flows, this allow for future possibilities on designing efficient algorithms tackling various on manifold problems §5.

Compatibility with Data Augmentations. It is important to note that our method is orthogonal to image space data augmentation methods. In other words, we can train normalizing flows with commonly used data augmentations. Indeed in our experiments, we observe that trained models apply some of the training-time augmentations such as cropping. This allows us to encode and decode *augmented* samples as well as original samples of CIFAR-10. Additionally, we can use more advanced methods such as DeVries & Taylor (2017); Zhang et al. (2018a) concurrently with our latent perturbations to train classifiers.

5. Conclusion

Motivated by the numerous advantages of normalizing flows, we propose flow-based latent perturbation methods to augment the training datasets, to train a classifier. Our empirical results on several real-world datasets demonstrate the efficacy of these generative models for improved test accuracy both in full and in low-data regimes.

Further directions include exploiting potentially more complex prior structures to design efficient flow-based algorithms tackling on-manifold sampling or optimization problems. For example, using NF models with explicit parametrization of specific semantic transformations (e.g., zoom or orientation) would enable the training of more robustly generalizing classifiers.

References

Antoniou, A., Storkey, A., and Edwards, H. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340*, 2017.

Ardizzone, L., Lüth, C., Kruse, J., Rother, C., and Köthe, U. Guided image generation with conditional invertible neural networks. *arXiv preprint arXiv:1907.02392*, 2019.

Baluja, S. and Fischer, I. Adversarial transformation networks: Learning to generate adversarial examples. *arXiv preprint arXiv:1703.09387*, 2017.

Biggio, B. and Roli, F. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331, 2018. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2018.07.023.

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. V. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 113–123, 2019.

Cubuk, E. D., Zoph, B., Shlens, J., and Le, Q. V. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020.

DeVries, T. and Taylor, G. W. Improved Regularization of Convolutional Neural Networks with Cutout. *arXiv:1708.04552*, 2017. URL http://arxiv.org/abs/1708.04552. arXiv: 1708.04552.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using Real NVP. In *International Conference on Learning Representations (ICLR)*, 2017.

Fawzi, A., Samulowitz, H., Turaga, D., and Frossard, P. Adaptive data augmentation for image classification. In *IEEE International Conference on Image Processing (ICIP)*, 2016.

Gilmer, J., Metz, L., Faghri, F., Schoenholz, S. S., Raghu, M., Wattenberg, M., and Goodfellow, I. Adversarial spheres. *arXiv preprint arXiv:1801.02774*, 2018.

Jalal, A., Ilyas, A., Daskalakis, C., and Dimakis, A. G. The robust manifold defense: Adversarial training using generative models. *arXiv preprint arXiv:1712.09196*, 2017.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, D. P. and Dhariwal, P. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in neural information processing systems*, pp. 10215–10224, 2018.

Krizhevsky, A. and Hinton, G. Learning Multiple Layers of Features from Tiny Images. pp. 60, 2009.

Larsen, A. B. L., Sønderby, S. K., and Winther, O. Autoencoding beyond pixels using a learned similarity metric. In *ICML*, 2016.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998. ISSN 1558-2256. doi: 10.1109/5.726791. Conference Name: Proceedings of the IEEE.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations (ICLR)*, 2018.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading digits in natural images with unsupervised feature learning. 2011.

Peng, X., Tang, Z., Yang, F., Feris, R. S., and Metaxas, D. Jointly optimize data augmentation and network training: Adversarial data augmentation in human pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Perez, L. and Wang, J. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*, 2017.

Ratner, A. J., Ehrenberg, H., Hussain, Z., Dunnmon, J., and Ré, C. Learning to compose domain-specific transformations for data augmentation. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/f26dab9bf6a137c3b6782e562794c2f2-Paper.pdf.

Rosca, M., Lakshminarayanan, B., Warde-Farley, D., and Mohamed, S. Variational approaches for auto-encoding generative adversarial networks. *arXiv preprint arXiv:1706.04987*, 2017.

Rozsa, A., Günther, M., and Boult, T. E. Are accuracy and robustness correlated. In *15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 227–232. IEEE Computer Society, 2016. doi: 10.1109/ICMLA.2016.0045. URL https://doi.org/10.1109/ICMLA.2016.0045.

Song, Y., Shu, R., Kushman, N., and Ermon, S. Constructing unrestricted adversarial examples with generative models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/8cea559c47e4fbdb73b23e0223d04e79-Paper.pdf.

Stutz, D., Hein, M., and Schiele, B. Disentangling adversarial robustness and generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6976–6987, 2019.

Su, D., Zhang, H., Chen, H., Yi, J., Chen, P., and Gao, Y. Is robustness the cost of accuracy? - A comprehensive study on the robustness of 18 deep image classification models. In *ECCV*, 2018.

Tanay, T. and Griffin, L. A boundary tilting persepective on the phenomenon of adversarial examples. *arXiv preprint arXiv:1608.07690*, 2016.

Tran, N.-T., Tran, V.-H., Nguyen, N.-B., Nguyen, T.-K., and Cheung, N.-M. On data augmentation for GAN training. *arXiv preprint arXiv:2006.05338*, 2020.

Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., and Madry, A. Robustness may be at odds with accuracy. In *International Conference on Learning Representations (ICLR)*, 2019a.

Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., and Madry, A. Robustness may be at odds with accuracy. *arXiv preprint arXiv:1805.12152*, 2019b.

Wong, E. and Kolter, J. Z. Learning perturbation sets for robust machine learning. In *International Conference on Learning Representations (ICLR)*, 2021.

Xiao, C., Li, B., yan Zhu, J., He, W., Liu, M., and Song, D. Generating adversarial examples with adversarial networks. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 3905–3911, 7 2018. doi: 10.24963/ijcai.2018/543. URL https://doi.org/10.24963/ijcai.2018/543.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., and Yoo, Y. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 6023–6032, 2019.

Zagoruyko, S. and Komodakis, N. Wide Residual Networks. In *Procedings of the British Machine Vision Conference 2016*, pp. 87.1–87.12, York, UK, 2016. British Machine Vision Association. ISBN 978-1-901725-59-9. doi: 10.5244/C.30.87. URL http://www.bmva.org/bmvc/2016/papers/paper087/index.html.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. mixup: Beyond Empirical Risk Minimization. In *International Conference on Learning Representations (ICLR)*, 2018a. URL http://arxiv.org/abs/1710.09412.

Zhang, L., Yu, M., Chen, T., Shi, Z., Bao, C., and Ma, K. Auxiliary training: Towards accurate and robust models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020a.

Zhang, R., Che, T., Ghahramani, Z., Bengio, Y., and Song, Y. Metagan: An adversarial approach to few-shot learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31. Curran Associates, Inc., 2018b. URL https://proceedings.neurips.cc/paper/2018/file/4e4e53aa080247bc31d0eb4e7aeb07a0-Paper.pdf.

Zhang, X., Wang, Q., Zhang, J., and Zhong, Z. Adversarial autoaugment. In *International Conference on Learning Representations (ICLR)*, 2020b.

Zhao, Z., Dua, D., and Singh, S. Generating natural adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2018.

A. Details on the implementation

In this section, we list all details on the implementation.

Source Code. Our source code is provided in this repository: https://github.com/okyksl/flow-lp.

A.1. Architectures

Generative model (NF) architecture. We use *Glow* (Kingma & Dhariwal, 2018) for the normalizing flow architecture. For the MNIST (Lecun et al., 1998) and FashionMNIST (Xiao et al., 2017) experiments, we use a conditional, 12-step, Glow-coupling-based architecture similar to (Ardizzone et al., 2019). See Table 3 for details. For the CIFAR-10/100 (Krizhevsky & Hinton, 2009) and SVHN (Netzer et al., 2011) experiments, we use the original Glow architecture described in (Kingma & Dhariwal, 2018), *i.e.*, 3 scales of 32 steps each containing activation normalization, affine coupling and invertible 1×1 convolution. We adapt an existing PyTorch implementation in [2] to better match the original Tensorflow implementation in [3]. For more details on multi-scale architecture in normalizing flows, see (Dinh et al., 2017).

Table 3. Normalizing flow architectures used for our experiments on **MNIST** and **FashionMNIST**. With $c_{in} \to y_{out}$ we denote the number of channels of the input and output of the layer. With $\oplus$ we denote concatenation operation. We use the implementation provided in https://github.com/VLL-HD/FrEIA. For more details on affine coupling layers, see (Kingma & Dhariwal, 2018).

Generative Model
Input: $\boldsymbol{x} \in \mathbb{R}^{784}, \boldsymbol{y} \in \mathbb{R}^{10}$
GLOWCouplingBlock
PermuteRandom
GLOWCouplingBlock
PermuteRandom
...
x9
...
GLOWCouplingBlock
PermuteRandom

GLOWCouplingBlock
Input: $\boldsymbol{x} \in \mathbb{R}^{784}, \boldsymbol{y} \in \mathbb{R}^{10}$
split $\boldsymbol{x} \to \boldsymbol{x}_1, \boldsymbol{x}_2$ $(784 \to 392, 392)$
subnet $\boldsymbol{x}_2 \oplus \boldsymbol{y} \to \mathrm{s}_1, \mathbf{t}_1$ $(402 \to 392, 392)$
affine coupling $\boldsymbol{x}_1, \mathrm{s}_1, \mathbf{t}_1 \to \boldsymbol{z}_1$ $(3\times 392 \to 392)$
subnet $\boldsymbol{z}_1 \oplus \boldsymbol{y} \to \mathrm{s}_2, \mathbf{t}_2$ $(402 \to 392, 392)$
affine coupling $\boldsymbol{x}_2, \mathrm{s}_2, \mathbf{t}_2 \to \boldsymbol{x}_2^{'}$ $(3\times 392 \to 392)$
concat. $\boldsymbol{z}_1 \oplus \boldsymbol{z}_2$ $(392, 392 \to 784)$

Subnets
Input: $\boldsymbol{x} \in \mathbb{R}^{402}$
linear $(402 \to 512)$
ReLU
linear $(512 \to 784)$
split $(784 \to 392, 392)$

Classifier architecture. For our experiments on MNIST, we use LeNet-5 (Lecun et al., 1998) with replaced nonlinearity– instead of $\tanh$ we use $ReLU$, and we initialize the network parameters with truncated normal distribution $\sigma = 0.1$. For the FashionMNIST experiments, we use the same classifier as used in (Stutz et al., 2019). See Table 4 for more details. For CIFAR-10/100 and SVHN, we use the ResNet-18 architecture as implemented in (DeVries & Taylor, 2017; Zhang et al., 2018a). This ResNet-18 includes slight modifications over the standard ResNet-18 architecture in order to achieve better performance on CIFAR-10/100. See [4] and [5] for implementation. In particular, the first layer is changed to a 3×3 convolution with stride 1 and padding 1, from the original 7×7 convolution with stride 2 and padding 3. Additionally, the following max-pooling layer is removed.

A.2. Hyperparameters

Generative Models. For MNIST and FashionMNIST, we use the Adam (Kingma & Ba, 2014) optimizer with a batch size of 100 and learning rate of 10^{-6} for 100 epochs to train normalizing flows. For CIFAR-10 and SVHN, we use the Adamax (Kingma & Ba, 2014) optimizer with learning rate of 0.0005 and weight decay of 0.00005. We use warmup learning rate scheduling for the first 500.000 steps of the training. That is, the learning rate is linearly increased from 0 to the base learning rate 0.0005 in 500.000 steps.

For VAE-GAN training, we run the implementation provided by authors[6] with the default architectures and parameters. That

[2]https://github.com/chrischute/glow
[3]https://github.com/openai/glow
[4]https://github.com/facebookresearch/mixup-cifar10
[5]https://github.com/uoguelph-mlrg/Cutout
[6]https://github.com/davidstutz/disentangling-robustness-generalization

Table 4. Convolutional Neural Network (CNN) architectures used for our experiments on **MNIST** and **FashionMNIST**. We use *ker* and *pad* to denote *kernel* and *padding* for the convolution layers, respectively. With $h \times w$ we denote the kernel size. With $c_{in} \rightarrow y_{out}$ we denote the number of channels of the input and output of the layer.

LeNet-5
Input: $\boldsymbol{x} \rightarrow \mathbb{R}^{1\times28\times28}$
convolution (ker: 5×5, $1 \rightarrow 6$; stride: 1; pad:2)
ReLU
AvgPool2d (ker: 2×2)
convolution (ker: 5×5, $6 \rightarrow 16$; stride: 1; pad:0)
ReLU
AvgPool2d (ker: 2×2)
Flatten ($16\times5\times5 \rightarrow 400$)
linear ($400 \rightarrow 120$)
ReLU
linear ($120 \rightarrow 84$)
ReLU
linear ($120 \rightarrow 10$)
ReLU

CNN from (Stutz et al., 2019)
Input: $\boldsymbol{x} \in \mathbb{R}^{1\times28\times28}$
convolution (ker: 4×4, $1 \rightarrow 16$; stride: 2; pad:1)
Batch Normalization
ReLU
convolution (ker: 4×4, $16 \rightarrow 32$; stride: 2; pad:1)
Batch Normalization
ReLU
convolution (ker: 4×4, $32 \rightarrow 64$; stride: 2; pad:1)
Batch Normalization
ReLU
Flatten ($64\times3\times3 \rightarrow 576$)
linear ($576 \rightarrow 100$)
linear ($100 \rightarrow 10$)

is, for FashionMNIST, we use $\beta = 2.75$, $\gamma = 1$, $\eta = 0$ and latent space size of 10. We use Adam optimizer with a batch size of 100, learning rate of 0.005, weight decay of 0.0001 and train VAE-GANs for 60 epochs with an exponential decay scheduling of 0.9 for the learning rate. For CIFAR-10, we use the CelebA setup provided (the only 3-channel color dataset provided) and thus use $\beta = 3.0$, latent space size of 25 and 30 epochs instead. Note that we report "On-Learned-Manifold Adversarial Training" from (Stutz et al., 2019) which uses class-specific VAE-GANs. That is, 10 separate VAE-GAN architectures are trained for both FashionMNIST and CIFAR-10 datasets.

Discussion on Hyperparameters of Generative Models. As normalizing flows directly optimize log-likelihood of the data, there are no hyperparameters in their loss function. Additionally, the normalizing flow models that we use have a fixed latent dimension equal to the input dimension due to their architectural design. This is in contrast to VAE-GAN used in (Stutz et al., 2019) where the training involves optimizing separate losses for three networks (namely, encoder, decoder and discriminator) concurrently. Coefficients called β, γ and η are used to scale different loss terms involved such as reconstruction, decoder and discriminator loss. Additionally, the latent size for VAE-GAN is hand-picked for each dataset.

Classifiers. For MNIST, we use the Adam optimizer with a learning rate of 0.001 and weight decay of 0.001. We train LeNet-5 classifiers for 20 epochs with exponential learning decay of rate 0.1 for 10.000 steps. For FashionMNIST, we use the training setup used in (Stutz et al., 2019). That is, we use Adam optimizer with a learning rate of 0.01 and weight decay of 0.0001. We train classifiers for 20 epochs with exponential learning decay of rate 0.9 for 500 steps. For CIFAR10/100, we use the training setup used in (DeVries & Taylor, 2017; Zhang et al., 2018a). More precisely, we use Stochastic Gradient Descent (SGD) with a batch size of 128, a learning rate of 0.1, weight decay of 0.0005, and Nesterov momentum of 0.9. We train ResNet-18 classifiers for 200 epochs and multiply the learning rate by 0.2 at epochs $\{60, 120, 160\}$. For SVHN, we use the same optimizer with a weight decay of 0.0001. We train ResNet-18 classifiers for 120 epochs and multiply the learning rate by 0.1 at epochs $\{30, 60, 90\}$.

Data Augmentation. For CIFAR-10/100, we use standard data augmentation akin to (Zagoruyko & Komodakis, 2016). That is, we zero-pad images with 4 pixels on each side, take a random crop of size 32×32, and then mirror the resulting image horizontally with 50% probability. We use such data augmentation for both training the generative and the classifier models. Hence, our normalizing flow model is capable of encoding-decoding operations on augmented samples as well. Advanced data augmentation baselines we use in Table 1 (DeVries & Taylor, 2017; Zhang et al., 2018a) also include the same standard data augmentations. However, (Stutz et al., 2019) does not use data augmentation in their generative model. To provide a more direct comparison between the performance of two generative models, in §B.3 we conduct an additional study without any data augmentations.

B. Additional Results

B.1. Results on MNIST

Table 5 summarizes our results on MNIST in full data regime. Although the baseline has very good performances on this dataset, we observe improved generalization.

Table 5. Train and test accuracy (%) as well as *loss* on **MNIST**. Comparison with standard training, versus ours—latent-space perturbations ($\mathcal{P}_{rand}$ & $\mathcal{P}_{adv}$).

Perturbation	**Train Accuracy**	**Train Loss**	**Test Accuracy**	**Test Loss**
Standard	99.80	0.0069	99.24	0.0288
$\mathcal{P}^{\ell_\infty}_{rand}, \epsilon{=}0.15$	99.78	0.0076	99.28	0.0262
$\mathcal{P}^{\ell_\infty}_{adv}, \epsilon{=}0.05, \alpha{=}0.01, k{=}10$	99.26	0.0230	99.43	0.0216

B.2. Results on SVHN

Table 6 summarizes our results on SVHN in low-data regime. Similarly to §3.2, we conduct a cross-dataset experiment between CIFAR-10 and SVHN, where we use pre-trained normalizing flows on CIFAR-10 to train classifiers on SVHN. To provide a baseline on the effect of using different datasets for normalizing flows and classifiers, we also provide results with pre-training on SVHN. Latent attacks transferred from CIFAR-10 achieve superior performance to direct pre-training on SVHN, indicating that transferring augmentations across datasets is promising.

Table 6. *Test accuracy* (%) on **SVHN**, in the *low-data regime*, where we use 5% of the training set and the full test set. Comparison with normalizing flows trained on **CIFAR-10**, versus **SVHN**.

	SVHN			**CIFAR-10**	
Perturbation	**Test**	**Improvement**	**Perturbation**	**Test**	**Improvement**
Standard	81.2	–	Standard	81.2	–
$\mathcal{P}^{\ell_2}_{rand}, \epsilon{=}15.0$	84.9	+3.7	$\mathcal{P}^{\ell_2}_{rand}, \epsilon{=}15.0$	90.0	+8.8
$\mathcal{P}^{\ell_2}_{adv}, \epsilon{=}0.5, \alpha{=}0.25, k{=}2$	86.9	+**5.7**	$\mathcal{P}^{\ell_2}_{adv}, \epsilon{=}0.3, \alpha{=}0.15, k{=}2$	90.5	+**9.3**

B.3. Additional Results on CIFAR-10

Results without Data Augmentation. To provide a direct comparison between two generative models and eliminate the effect of data augmentation, we run additional experiments. Table 7 shows results for our latent perturbations without any data augmentation to train the normalizing flow and the classifier. In line with our FashionMNIST results in §3.3, we observe that both Randomized–LA and Adversarial–LA overperform standard baseline and VAE-GAN based approach.

Table 7. Test accuracy (%) on **CIFAR-10**, in the *low-data regime*, where we use 5% of the training set and the full test set, without data augmentation.

Method	**Low-data**
Standard	49.8
VAE-GAN (Stutz et al., 2019)	49.4
Randomized–LA (ours)	54.9
Adversarial–LA (ours)	58.2

Results with Different Attack Parameters. In Table 8, we provide results with varying hyperparameters for the different attacks. Observe that in the "higher" Adversarial–LA perturbation setting–where $\epsilon = 2.0$, the classifier still didn't fully fit to the training set but the test performance is above the standard baseline.

Multi-step Training. We run additional experiments where we sequentially apply different attack hyperparameters in a multi-step training with weaker perturbations to increase the performance on the test set. The results are listed in Table 8, denoted with $+$.

Table 8. *Train and test accuracy* (%) as well as *loss* on **CIFAR-10**, using ResNet-18. All of the models are trained with the same hyperparameters listed in §A.2. Perturbations listed with the $+$ sign indicates a multi-step training. For example, last row lists the result of the model trained with $P_{adv}^{\ell_2}, \epsilon = 2.0, \alpha = 1.5, k = 2$ for 130 epochs, $P_{rand}, \epsilon = 0.25$ for 40 epochs and $P_{rand}^{\ell_2}, \epsilon = 10.0$ for 30 epochs. Note that, regardless of multi-step training, the hyperparameters, including the total number of training epochs (= 200), remain fixed across the experiments.

Perturbation	Train Accuracy	Train Loss	Test Accuracy	Test Loss
Baselines:				
Standard	100.0	0.002	95.2	0.194
$P_{PGD}^{\ell_2}, \epsilon$=2.0, α=0.5, k=10	61.13	0.895	75.7	0.731
$P_{PGD}^{\ell_\infty}, \epsilon$=0.03, α=0.008, k=10	77.3	0.521	86.3	0.442
Ours:				
$\mathcal{P}_{rand}^{\ell_2}, \epsilon$=10.0	99.8	0.007	95.8	0.161
$\mathcal{P}_{rand}^{\ell_\infty}, \epsilon$=0.25	99.5	0.015	**96.3**	0.142
$+\mathcal{P}_{rand}, \epsilon$=0.15	100.0	0.002	**96.4**	0.133
$\mathcal{P}_{adv}^{\ell_2}, \epsilon$=1.0, α=0.5, k=3	99.9	0.005	**96.6**	0.126
$\mathcal{P}_{adv}^{\ell_2}, \epsilon$=2.0, α=1.5, k=2	89.1	0.214	95.8	0.134
$+\mathcal{P}_{adv}^{\ell_2}, \epsilon$=1.0, α=0.75, k=2	99.2	0.030	96.5	0.114
$+\mathcal{P}_{adv}^{\ell_2}, \epsilon$=0.75, α=0.5, k=2	99.7	0.011	**96.7**	0.115
$+\mathcal{P}_{rand}, \epsilon$=0.25	100.0	0.002	96.5	0.132
$+\mathcal{P}_{rand}^{\ell_2}, \epsilon$=10.0	100.0	0.002	96.6	0.131

Generated Images. Figure 3 depicts samples obtained with our Randomized-LA and Adversarial-LA methods. In contrast to random image space perturbations, we observe that both Randomized-LA and Adversarial-LA yield perturbations dependent on the semantic content of the input image.

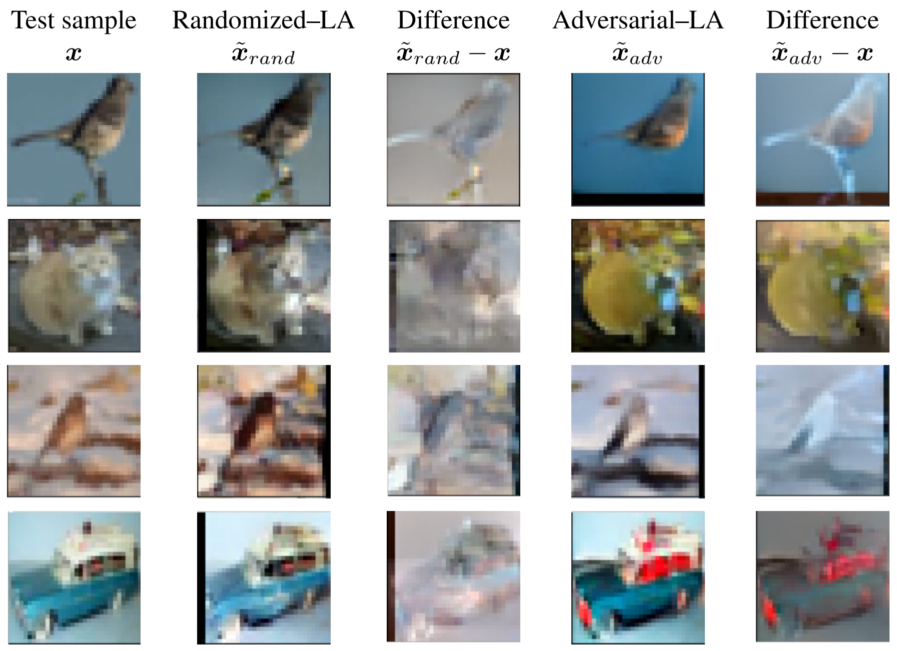
Test sample
$\boldsymbol{x}$
Randomized–LA
$\tilde{\boldsymbol{x}}_{rand}$
Difference
$\tilde{\boldsymbol{x}}_{rand} - \boldsymbol{x}$
Adversarial–LA
$\tilde{\boldsymbol{x}}_{adv}$
Difference
$\tilde{\boldsymbol{x}}_{adv} - \boldsymbol{x}$

Figure 3. **Illustrative results of our latent space perturbations.** The first column depicts a randomly selected samples from the test set $\boldsymbol{x}$. We depict the outputs obtained with Eq. **Randomized–LA** and Eq. **Adversarial–LA** as well as their difference with the test sample $\boldsymbol{x}$. By observing the difference images, we see that the added perturbations depend on the semantic content of the input image.