Pattern Recognition Letters 152 (2021) 382–390

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

On the benefits of defining vicinal distributions in latent space

Puneet Mangla*, Vedant Singh, Shreyas Havaldar, Vineeth Balasubramanian

Department of Computer Science and Engineering, IIT Hyderabad, Sangareddy, 502284, India

ARTICLE INFO

Article history:
Received 8 December 2020
Revised 10 October 2021
Accepted 18 October 2021
Available online 19 October 2021

Edited by : Jiwen Lu

Keywords:
Robustness
Calibration
Mixup
VRM
Common corruptions

ABSTRACT

The vicinal risk minimization (VRM) principle is an empirical risk minimization (ERM) variant that replaces Dirac masses with vicinal functions. There is strong numerical and theoretical evidence showing that VRM outperforms ERM in terms of generalization if appropriate vicinal functions are chosen. Mixup Training (MT), a popular choice of vicinal distribution, improves generalization performance of models by introducing globally linear behavior in between training examples. Apart from generalization, recent works have shown that mixup trained models are relatively robust to input perturbations/corruptions and at same time are calibrated better than their non-mixup counterparts. In this work, we investigate the benefits of defining these vicinal distributions like mixup in latent space of generative models rather than in input space itself. We propose a new approach - *VarMixup (Variational Mixup)* - to better sample mixup images by using the latent manifold underlying the data. Our empirical studies on CIFAR-10, CIFAR-100 and Tiny-ImageNet demonstrates that models trained by performing mixup in the latent manifold learned by VAEs are inherently more robust to various input corruptions/perturbations, are significantly better calibrated and exhibit more local-linear loss landscapes.

1. Introduction

Deep Neural Networks (DNNs) have become a key ingredient to solve many challenging tasks like classification, segmentation, object detection, speech recognition, etc. In most successful applications, these networks are trained to minimize the average error over the training dataset known as the Empirical Risk Minimization (ERM) principle [34]. However, various empirical and theoretical studies have shown that minimizing Empirical Risk over training datasets in over-parameterized settings leads to memorization and thus poor generalization on examples just outside the training distribution. Some classical results in learning theory [33] tells us that the convergence of ERM is guaranteed as long as the size of the learning machine (in terms of number of parameters or VC-complexity [11]) does not increase with the number of training data. To mitigate this problem of memorization in over-parameterized neural networks, Vicinal Risk Minimization (VRM) was proposed which essentially chooses to train networks on similar but different examples to the training data. This technique more popularly known as data augmentation [27], requires one to define a vicinity or neighbourhood around each training example (eg. in terms of brightness, contrast, imperceptible noise, to name a few). Once defined, more examples can be sampled from their vicinity to enlarge the support of training distribution.

One of the popular choices to create the vicinal distribution is Mixup. Mixup Training (MT) [43] has emerged as a popular technique to train models for better generalisation in the last couple of years. Recent works have also shown that the idea of Mixup and Mixup training can be leveraged during inference [25] and in many existing techniques like data augmentation [14], adversarial training [19], etc. to improve the robustness of models to various input perturbations [30,39] and corruptions [14]. Another variant of Mixup, known as Manifold Mixup [35] encourages neural networks to predict less confidently on interpolations of hidden representations by leveraging semantic interpolations as an additional training signal. As a result, neural networks trained with Manifold Mixup learn class representations with fewer directions of variance. Other efforts on Mixup [31] have shown that Mixup-trained networks are significantly better calibrated and less prone to over-confident predictions on out-of-distribution than the ones trained in the regular fashion.

Although still in its early phase, the above efforts [25,31,35,43] also indicate a trend to view Mixup from perspectives of robustness and calibration. In this work, we take another step in this direction and propose a new vicinal distribution/sampling technique called *VarMixup (Variational Mixup)* to sample better Mixup images during training to induce robustness as well as improve predictive uncertainty of models while preserving the clean data performance to extent possible. In particular,

* Corresponding author.
E-mail addresses: pmangla261@gmail.com (P. Mangla), cs18btech11047@iith.ac.in (V. Singh), cs18btech11042@iith.ac.in (S. Havaldar), vineethnb@cse.iith.ac.in (V. Balasubramanian).

https://doi.org/10.1016/j.patrec.2021.10.016
0167-8655/

we hypothesize that the latent unfolded manifold underlying the data (through a generative model, a Variational Autoencoder in our case) is linear by construction (manifolds unfold the locally linear structure of a high-dimensional data space), and hence more suitable for the defining vicinal distributions involving linear interpolations, such as Mixup. Importantly, we show that this choice of the distribution for Mixup plays an important role towards robustness and predictive uncertainty (Section 3). We note herein that our downstream task is image classification and the VAE used in our approach is an auxiliary tool rather than being the model to be trained in the first place.

Our contributions can be summarized as follows:

- We propose a new sampling technique called *VarMixup (Variational Mixup)* to sample better Mixup images during training by using the latent manifold learned by generative models. Our experiments on 3 standard datasets- CIFAR-10, CIFAR-100 and Tiny-ImageNet show that VarMixup significantly boosts the robustness to out-of-distribution shifts as well calibration of neural networks as compared to regular mixup or manifold-mixup training.
- We conduct additional analysis/studies which show that VarMixup significantly decreases the local linearity error of the neural network and generates samples that are slightly off-distribution from training examples or mixup generated samples, to provide robustness.

2. Background and related work

2.1. Notations and preliminaries

We denote a neural network as $F_w : \mathbb{R}^{c\times h\times w} \to \mathbb{R}^k$, with weight parameters w. F_w takes an image $x \in \mathbb{R}^{c\times h\times w}$ and outputs logits, $F_w^i(x)$ for each class $i \in \{1 \dots k\}$. Without loss of generality, we assume the classification task with $\mathcal{L}$ as the standard cross-entropy loss function. p_{actual} denotes the training data distribution, and the optimal weight parameter w^* is obtained by training the network using standard empirical risk minimization [34], i.e. $w^* = \arg\min_w \mathbb{E}_{(\mathbf{x},y)\sim p_{actual}}[\mathcal{L}(F_w(\mathbf{x}), y)]$, where y is the true label associated with input x.

2.2. Vicinal risk minimization

Given the data distribution p_{actual}, a neural network F_w and loss function $\mathcal{L}$, the *expected risk* (average of loss function over p_{actual}) is given by $R(F_w) = \int \mathcal{L}(F_w(x), y) \cdot dp_{actual}(x, y)$. In practice, the true distribution p_{actual} is unknown, and is approximated by the training dataset $D = \{(x_i, y_i)\}_{i=1}^N$, which represents the *empirical distribution*: $p_\delta(x, y) = \frac{1}{N} \cdot \sum_{i=1}^N \delta(x = x_i, y = y_i)$. Here, $\delta(x = x_i, y = y_i)$ is the Dirac delta function centered at (x_i, y_i). Using p_δ as an estimate to p_{actual}, we define *expected empirical risk* as:

$$R_\delta(F_w) = \frac{1}{N} \cdot \sum_{i=1}^{N} \mathcal{L}(F_w(x_i), y_i) \tag{1}$$

Minimizing Eqn 1 to find optimal F_{w^*} is typically termed *Empirical Risk Minimization (ERM)* [34]. However overparametrized neural networks can suffer from memorizing, leading to undesirable behavior of network outside the training distribution, p_δ [30,41]. Addressing this concern, [33] and [5] proposed *Vicinal Risk Minimization (VRM)*, where p_{actual} is approximated by a vicinal distribution p_ν, given by:

$$p_\nu(x, y) = \frac{1}{N} \cdot \sum_{i=1}^{N} \nu(x, y|x_i, y_i) \tag{2}$$

where ν is the *vicinal distribution* that calculates the probability of a data point (x, y) in the vicinity of other samples (x_i, y_i). Thus, using p_ν to approximate p_{actual}, *expected vicinal risk* is given by:

$$R_\nu(F_w) = \frac{1}{N} \cdot \sum_{i=1}^{N} g(F_w, \mathcal{L}, x_i, y_i) \tag{3}$$

where $g(F_w, \mathcal{L}, x_i, y_i) = \int \mathcal{L}(F_w(x), y) \cdot d\nu(x, y|x_i, y_i)$. The superiority of VRM over ERM has been theoretically as well as empirically verified by many recent works [4,10,24,42].

Popular examples of vicinal distributions include: (i) *Gaussian Vicinal distribution*: Here, $\nu_{gaussian}(x, y|x_i, y_i) = \mathcal{N}(x - x_i, \sigma^2) \cdot \delta(y = y_i)$, which is equivalent to augmenting the training samples with Gaussian noise; and (ii) *Mixup Vicinal distribution* : Here $\nu_{mixup}(x, y|x_i, y_i) = \frac{1}{n} \cdot \sum_{j=1}^N \mathbb{E}_\lambda[\delta(x = \lambda \cdot x_i + (1 - \lambda) \cdot x_j, y = \lambda \cdot y_i + (1 - \lambda) \cdot y_j)]$, where $\lambda \sim \beta(\eta, \eta)$ and $\eta > 0$.

2.3. Mixup

[43] proposed Mixup, a method to train models on the convex combination of pairs of examples and their labels. In other words, it constructs virtual training examples as: $x' = \lambda \cdot x_i + (1 - \lambda) \cdot x_j; y' = \lambda \cdot y_i + (1 - \lambda) \cdot y_j$, where x_i, x_j are input vectors; y_i, y_j are one-hot label encodings and λ is a mixup coefficient, usually sampled from a $\beta(\eta, \eta)$ distribution. By doing so, it regularizes the network to behave linearly in between training examples, thus inducing global linearity between them. A recent variant, Manifold Mixup [35], exploits interpolations at hidden representations, thereby obtaining neural networks with smoother decision boundaries at different levels of hidden representations. AugMix [14] mixes up multiple augmented images and uses a Jensen-Shannon Divergence consistency loss on them to achieve better robustness to common input corruptions [13]. In semi-supervised learning, MixMatch [2] obtains state-of-the-art results by guessing low-entropy labels for data-augmented unlabeled examples and mixes labeled and unlabeled data using Mixup. It has been shown that apart from better generalization, Mixup also improves the robustness of models to adversarial perturbations as well. To further boost this robustness at inference time, Pang et al. [25] recently proposed a Mixup Inference technique which performs a mixup of input x with a clean sample x_s and passes the corresponding mixup sample $(\lambda \cdot x + (1 - \lambda) \cdot x_s)$ into the classifier as the processed input.

Other efforts related to Mixup [31] have shown that Mixup-trained networks are better calibrated i.e., the predicted softmax scores are better indicators of the actual likelihood of a correct prediction than DNNs trained in the regular fashion. Additionally, they also observed that mixup-trained DNNs are less prone to over-confident predictions on out-of-distribution and random-noise data. None of these efforts however address Mixup from a generative latent space, which is the focus of this work. Efforts such as [25] and [31], in fact, have inferences that motivate the need to consider a latent Mixup space to address a model's robustness and predictive uncertainty.

From a different perspective, Xu et al. [37] used domain mixup to improve the generalization ability of models in domain adaptation. Adversarial Mixup Resynthesis [1] attempted mixing latent codes used by autoencoders through an arbitrary mixing mechanism that can recombine codes from different inputs to produce novel examples. This work however has a different objective and focuses on generative models in a Generative Adversarial Network (GAN)-like setting, while our work focuses on robustness and predictive uncertainty. The work by Liu et al. [21] may be closest to ours in terms of approach as they use an adversarial autoencoder (AAE) to impose a uniform distribution on the feature representations. However, their work deals with improving generalization performance, while ours looks at robustness and predictive uncertainty, as already stated. Other related works like [22,36,38] attempt to leverage Generative Adversarial Networks (GANs) to train

adversarially robust classifiers, while, we focus on VAEs, because of their ability to model the latent manifold explicitly, to train classifiers robust to commonly observed out-of-distribution shifts (eg. snow, fog etc.) (explained in detail in Section 3). Furthermore, we propose a new method, VarMixup, which focuses on directly exploiting the manifold learned by a Variational Autoencoder (VAE) (and do not regularize it unlike previous work) during Mixup and report improved adversarial robustness. We also present useful insights into the working of VarMixup (which is lacking in earlier work including [21]), thus making our contributions unique and more complete.

3. Methodology

In this work, we build on the recent success of using Mixup as a vicinal distribution by proposing the use of the latent spaces learned by a generative deep neural network model. The use of generative models such as Variational Autoencoders (VAEs) [17] to capture the latent space from which a distribution is generated provides us an unfolded manifold (the low-dimensional latent space), where the linearity in between training examples is more readily observed. Defining vicinal distributions by using neighbors on this latent manifold, which is more linear in the low-dimensional space, learned by generative models provides us more effective linear interpolations than the ones in input space. We hence leverage such an approach to capture the induced global linearity in between examples, and define Mixup vicinal distributions on this latent surface.

3.1. Our approach: Varmixup (variational mixup)

To capture the latent manifold of the training data through a generative model, we opt for a Variational Autoencoder (VAE). VAE [17] is an autoencoder which is trained using Variational Inference, which serves as an implicit regularizer to ensure that the obtained latent space allows us to generate new data from the same distribution as training data. Our rationale behind choosing VAEs over GANs to capture latent manifold of training data is that while VAEs are known for modeling latent variable models explicitly, GANs are implicit generative models, i.e. while we can generate images from latent variables, the reverse operation - getting latent variable samples corresponding to images - is not explicitly modeled. To obtain the latent embedding of an image x, one may have to solve the following optimization via backpropogation:

$$z^* = \arg\min_z \|G(z) - x\|_2 \tag{4}$$

where G is the generator. Clearly, such an optimization causes additional overhead in increased time and computation complexity. Since we work on defining vicinal distributions in the latent space, choosing VAEs directly allows us to obtain the latent embeddings via one forward pass.

We denote the encoding and decoding distribution of VAE as $q_\phi(z|x)$ and $p_\theta(x|z)$ respectively, parametrized by ϕ and θ respectively. Given $p(z)$ as the desired prior distribution for encoding, the general VAE objective is given by the loss function:

$$\mathcal{L}_{VAE} = -\gamma \cdot D(q_\phi(z)\|p(z)) + \mathbb{E}_{x\sim p_{actual}}\mathbb{E}_{z\sim q_\phi(z|x)}[\log(p_\theta(x|z))] \tag{5}$$

Here, D is any strict divergence, meaning that $D(q\|p) \geq 0$ and $D(q\|p) = 0$ if and only if $q = p$, and $\gamma > 0$ is a scaling coefficient. The second term in the objective acts as a image reconstruction loss and $q_\phi(z) = \mathbb{E}_{x\sim p_{actual}}[q_\phi(z|x)]$. The original VAE [17] uses KL-divergence in Eqn 5, and thus optimizes the following objective:

$$\mathcal{L}_{VAE} = \mathbb{E}_{x\sim p_{actual}}\left[-\gamma KL(q_\phi(z|x)\|p(z)) + \mathbb{E}_{z\sim q_\phi(z|x)}\log(p_\theta(x|z))\right] \tag{6}$$

However, using KL-divergence in Eqn 5 has some shortcomings, as pointed out in [6,28,29,45].

KL-divergence encourages the encoding $q_\phi(z|x)$ to be a random sample from $p(z)$ for each x, making them uninformative about the input. Also, it is not strong enough a regularizer compared to the reconstruction loss and tends to overfit data, consequently, learning a $q_\phi(z|x)$ that has high variance. Both the aforementioned shortcomings can affect the encoding distribution by making them uninformative of inputs with high variance. Since we use VAEs to better capture a linear latent manifold and subsequently define interpolations there, a bad latent distribution can affect our method significantly. Hence, we use a variant *Maximum Mean Discrepancy VAE* (MMD-VAE)[45] which uses a MMD Loss [8] instead of KL-divergence, and hence optimizes the following objective:

$$\begin{aligned}\mathcal{L}_{MMD-VAE} = \gamma \cdot MMD(q_\phi(z)\|p(z)) \\ + \mathbb{E}_{x\sim p_{actual}}\mathbb{E}_{z\sim q_\phi(z|x)}[\log(p_\theta(x|z))]\end{aligned} \tag{7}$$

A MMD-VAE doesn't suffer from the aforementioned shortcomings [45], as it maximizes mutual information between x and z by matching the distribution over encodings $q_\phi(z)$ with prior $p(z)$ only in expectation, rather than for every input. We hence train an MMD-VAE to characterize the training distribution more effectively. Once trained, we now define a Mixup vicinal distribution in the latent space of the trained VAE as:

$$\begin{aligned}\nu_{VarMixup}(z, y|x_i, y_i) = \tfrac{1}{n}\cdot\textstyle\sum_{j=1}^{N}\mathbb{E}_\lambda\Big[\delta\big(z = \lambda\cdot\mathbb{E}_z\big[q_\phi(z|x_i)\big] \\ + (1-\lambda)\cdot\mathbb{E}_z\big[q_\phi\big(z|x_j\big)\big], y = \lambda\cdot y_i + (1-\lambda)\cdot y_j\big)\Big]\end{aligned} \tag{8}$$

where $\lambda \sim \beta(\eta, \eta)$ and $\eta > 0$. Using the above vicinal distribution, $\nu_{VarMixup}$ and the MMD-VAE decoder, $p_\theta(x|z)$, we construct VarMixup samples as:

$$\begin{aligned}x' &= \mathbb{E}_x[p_\theta(x|\lambda\cdot\mathbb{E}_z[q_\phi(z|x_i)] + (1-\lambda)\cdot\mathbb{E}_z[q_\phi(z|x_j)])] \\ y' &= \lambda\cdot y_i + (1-\lambda)\cdot y_j\end{aligned} \tag{9}$$

From another perspective, one could view our new sampling technique as performing Manifold Mixup [35], however over the latent space of an MMD-VAE (instead of the neural network feature space) and using it for sample reconstruction. We compare against Manifold Mixup in our results to show the improved performance of the learned generative latent space in our VarMixup. Fig. 1 illustrates the conceptual idea behind VarMixup. The entire training methodology of VarMixup can be summarized in the following steps:

- Train an MMD-VAE using Eq. 7.
- Generate VarMixup samples $\{(x^{(i)}, y^{(i)})\}$ according to Eq. 9.
- Optimize model on generated VarMixup samples, $\{(x^{(i)}, y^{(i)})\}$ via standard cross-entropy loss.

4. Experiments and results

We now present our experimental studies and results using our method, VarMixup, on multiple datasets. We begin by describing the datasets, evaluation criteria and implementation details. Note that we focus explicitly on the usefulness of our approach on out-of-distribution test data and addressing predictive uncertainty

Datasets: We perform experiments on three well-known standard datasets: CIFAR-10, CIFAR-100 [18] and Tiny-ImageNet [7]. *CIFAR-10* is a subset of 80 million tiny images dataset and consists of 60,000 32×32 color images containing one of 10 object classes, with 6000 images per class. *CIFAR-100* is just like CIFAR-10, except that it has 100 classes containing 600 images each. There are 500 training images and 100 testing images per class. *Tiny-Imagenet* has 200 classes, with each class containing 500 training images, 50 validation images, and 50 test images. Each image here is of resolution 64×64.

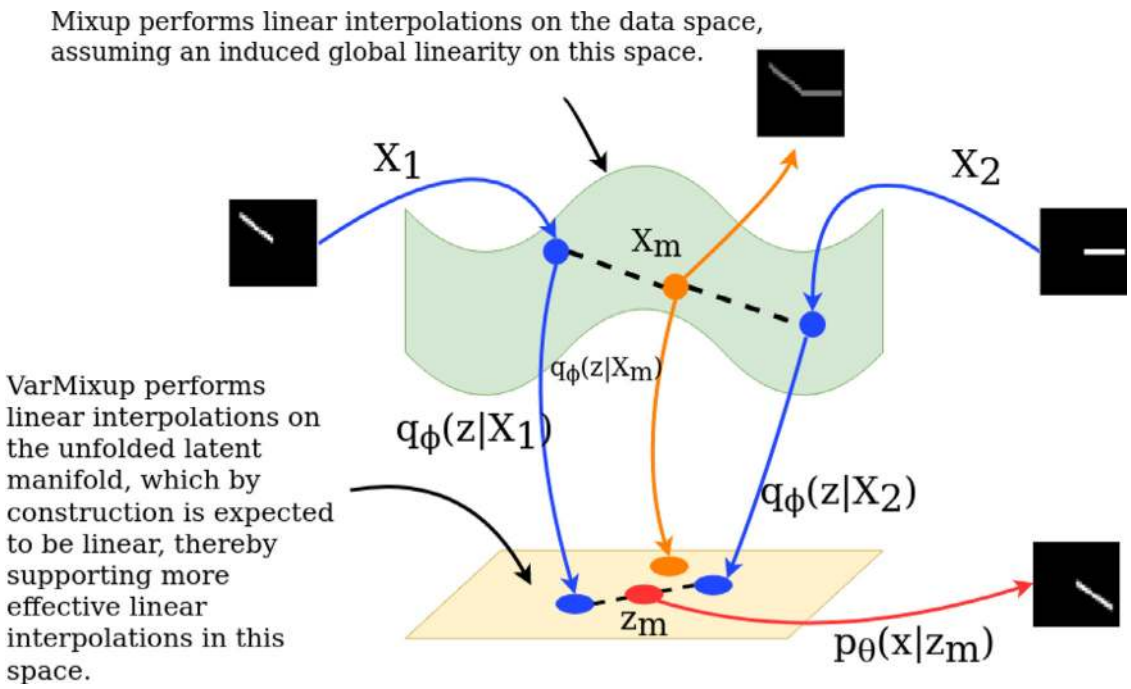

Fig. 1. Illustration of the conceptual idea behind VarMixup. We interpolate on the unfolded manifold, as defined by a generative model (VAE, in our case).

Evaluation Criteria: To measure the generalization of our models on out-of-distribution data, we evaluate their robustness on the newer CIFAR-10-C, CIFAR-100-C and Tiny-Imagenet-C datasets [13]. These datasets contain images, corrupted with 15 different distortions at 5 severity levels (Gaussian blur, Shot Noise, Impulse Noise, JPEG compression, Motion blur, frost, to name a few). For completeness, we also report accuracy on clean images and standard deviations over 10 trials (which captures standard generalization performance). We also measure the Expected Calibration Error (ECE) [9] of our trained models to quantify their predictive uncertainty.

Implementation Details: It has been shown [16] that adversarial robust training [23] removes irrelevant biases (e.g. texture biases) in their hidden representations, thus making them more informative. We hence hypothesize that the considered VAE, if trained in an adversarially robust fashion, will have more informative latent encoding than its regular equivalent. This would hence help improve the empirical/vicinal distributions like VarMixup. Empirically, we validate this hypothesis in our subsequent experiments and use prefix *adv-* (eg: *adv*-VarMixup) to distinguish them from their regular variants. We note that the aforementioned approach of adversarial robust training [23] is different from adversarial training used to train GAN-like architectures, and hence both should not be confused. In other words, the adversarial robust training that we are referring to, minimizes adversarial ELBO instead of standard ELBO (7) of an MMD-VAE. Given a dataset $\mathcal{D} = \{(x_i, y_i)\}$, we identify adversarial ELBO $\mathcal{L}^{adv}_{MMD-VAE}$ as follows:

$$
\begin{aligned}
&\because q_\phi(z) = \mathbb{E}_{x\sim p_{actual}}\left[q_\phi(z|x)\right] = \tfrac{1}{|\mathcal{D}|}\textstyle\sum_i q_\phi(z|x_i) \\
&\mathcal{L}_{MMD-VAE}\ (x_1, \ldots x_i) = \gamma \cdot MMD\Big(\tfrac{1}{|\mathcal{D}|}\textstyle\sum_i q_\phi(z|x_i)\ \Big\|\ p(z)\Big) \\
&\quad + \tfrac{1}{|\mathcal{D}|}\textstyle\sum_i \mathbb{E}_{z\sim q_\phi(z|x_i)}\ [\ \log\left(p_\theta\left(x_i \mid z\right)\right)\ ] \\
&x_i^* = \max_{x_i\ \in\ \mathcal{B}(x_i,\epsilon)} \mathcal{L}_{MMD-VAE}\ (x_1, \ldots x_i) \\
&\mathcal{L}^{adv}_{MMD-VAE}\ (x_1, \ldots x_i) = \gamma \cdot MMD\Big(\tfrac{1}{|\mathcal{D}|}\textstyle\sum_i q_\phi\left(z|x_i^*\right)\ \Big\|\ p(z)\Big) \\
&\quad + \tfrac{1}{|\mathcal{D}|}\textstyle\sum_i \mathbb{E}_{z\sim q_\phi(z|x_i^*)}\ \Big[\ \log\Big(p_\theta\left(x_i^*\ \Big|\ z\right)\Big)\ \Big]
\end{aligned}
\tag{10}
$$

We choose Resnet-34 [12] and WideResNet-28-10 [40] (SOTA backbone [23,35,43]) as the backbone architecture for evaluating our approach and baselines.

Baseline Models: We compare our method, VarMixup, against an exhaustive set of baselines including non-VRM variants, mixup variants and state-of-the-art adversarial techniques. Below are their details:

1. *ERM* - Vanilla Empirical Risk Minimization (Eqn 1) using Adam optimizer ($lr = 1e-3$) for 100 epochs on all datasets.
2. *Mixup* - Vanilla Mixup training [43] using Adam optimizer ($lr = 1e-3$) for 150 epochs on all datasets. Mixup coefficient is sampled from $\beta(1, 1)$.
3. *Mixup-R* - Mixup training on MMD-VAE's reconstructed image space [43] using Adam optimizer ($lr = 1e-3$) for 150 epochs on all datasets. Mixup coefficient is sampled from $\beta(1, 1)$.
4. *Manifold Mixup* - Manifold Mixup training [35] using Adam optimizer ($lr = 1e-3$) for 150 epochs on all datasets. Mixup coefficient is sampled from $\beta(2, 2)$.
5. *AT and TRADES* - l_∞ PGD/TRADES adversarial training [23,44] with $\epsilon = 8/255$ and step-size $\alpha = 2/255$. Models are trained using Adam optimizer ($lr = 1e-3$) for 250 epochs on all datasets.
6. *IAT* - l_∞ Interpolated adversarial training [19] with $\epsilon = 8/255$ and step-size $\alpha = 2/255$. Interpolation coefficient is sampled from $\beta(1, 1)$. Models are trained using Adam optimizer ($lr = 1e-3$) for 350 epochs on all datasets.

Generalization Performance and Robustness to Out-of-Distribution shifts

We first evaluate the trained models on their robustness to various common input corruptions, along with their generalization performance on "clean data" (test data without corruptions). Hendrycks et al [13] recently proposed the CIFAR-10-C, CIFAR-100-C, and Tiny-Imagenet-C datasets, which are extensions of CIFAR-10, CIFAR-100 and Tiny-Imagenet containing images corrupted with 15 different distortions and 5 levels of severity. We report the mean classification accuracy over all distortions on these datasets in Table 1 and Table 2 for ResNet-34 [12] and WideResNet-28-10 [40] architectures respectively. The results show that our method - VarMixup/*adv*-VarMixup achieves superior performance by a margin of $\sim 2-10\%$ consistently across the datasets.

Table 1
Robustness to common input corruptions on CIFAR-10-C, CIFAR-100-C and Tiny-Imagenet-C [13] datasets using ResNet-34 [12] backbone. Best results in **bold** and second best underlined. Clean accuracy is reported in parentheses using gray colour.

Method	CIFAR-10-C	CIFAR-100-C	Tiny-Imagenet-C
AT [23]	73.12 ± 0.31 (85.58 ± 0.14)	45.09 ± 0.31 (60.28 ± 0.13)	15.74 ± 0.36 (22.33 ± 0.16)
TRADES [23]	75.46 ± 0.21 (88.11 ± 0.43)	45.98 ± 0.41 (63.3 ± 0.32)	16.20 ± 0.23 (26.12 ± 0.38)
IAT [19]	81.05 ± 0.42 (89.7 ± 0.33)	50.71 ± 0.25 (62.7 ± 0.21)	18.69 ± 0.45 (18.08 ± 0.34)
ERM	69.29 ± 0.21 (94.5 ± 0.14)	47.3 ± 0.32 (64.5 ± 0.10)	17.34 ± 0.27 (49.96 ± 0.12)
Mixup	74.74 ± 0.34 (95.5 ± 0.35)	52.13 ± 0.43 (76.8 ± 0.41)	21.55 ± 0.37 (53.83 ± 0.17)
Mixup-R	74.27 ± 0.22 (89.88 ± 0.11)	43.54 ± 0.15 (62.24 ± 0.21)	21.34 ± 0.32 (53.5 ± 0.28)
Manifold-Mixup	72.54 ± 0.14 (95.2 ± 0.18)	41.42 ± 0.23 (75.3 ± 0.48)	-
VarMixup	82.57 ± 0.42 (93.91 ± 0.45)	52.57 ± 0.39 (73.2 ± 0.44)	24.87 ± 0.32 (50.98 ± 0.11)
adv-VarMixup	82.12 ± 0.46 (92.19 ± 0.32)	54.0 ± 0.41 (72.13 ± 0.34)	25.36 ± 0.21 (50.58 ± 0.23)
Mixup + VarMixup	**83.36** ± **0.46** (94.1 ± 0.13)	**54.36** ± **0.05** (75.3 ± 0.23)	**26.87** ± **0.21** (52.58 ± 0.47)

Table 2
Robustness to common input corruptions on CIFAR-10-C and CIFAR-100-C [13] datasets using WideResNet-28-10 [40] backbone. Best results in **bold** and second best underlined. Clean accuracy is reported in parentheses using gray colour.

Method	CIFAR-10-C	CIFAR-100-C
AT	74.8 ± 0.34 (87.32 ± 0.11)	46.1 ± 0.06 (62.5 ± 0.20)
TRADES	77.39 ± 0.21 (89.97 ± 0.53)	46.7 ± 0.22 (65.6 ± 0.33)
IAT	82.25 ± 0.44 (91.3 ± 0.09)	52.3 ± 0.56 (63.67 ± 0.77)
ERM	72.46 ± 0.12 (96.0 ± 0.43)	46.7 ± 0.22 (77.27 ± 0.39)
Mixup	75.62 ± 0.16 (97.1 ± 0.51)	52.46 ± 0.11 (80.53 ± 0.37)
Manifold-Mixup	73.78 ± 0.31 (97.3 ± 0.08)	45.6 ± 0.33 (81.2 ± 0.26)
VarMixup	84.39 ± 0.22 (95.81 ± 0.13)	53.78 ± 0.42 (77.24 ± 0.56)
adv-VarMixup	**84.7** ± **0.08** (94.2 ± 0.17)	54.72 ± 0.48 (75.97 ± 0.35)
Mixup + VarMixup	83.92 ± 0.10 (96.78 ± 0.36)	**58.22** ± **0.25** (79.3 ± 0.27)

We observe a slight drop in the clean accuracy of VarMixup models (shown in parentheses in Table 1 and Table 2) which we believe is due to the tradeoff between robustness and clean accuracy, a common trend observed in robustness literature [32]. However, in an attempt to strike a balance between both (i.e reduce the tradeoff), we conduct an additional experiment where we exploit the benefits that Mixup or VarMixup offer individually. More specifically in each training iteration, we randomly choose (with probability of 0.5) to sample either using Mixup or VarMixup distribution. We refer this experiment as Mixup + VarMixup in Table 1 and Table 2 and observe that it leads to clean test accuracy comparable to regular Mixup/Manifold-Mixup whilst improving or atleast maintaining similar performance on corrupted benchmarks.

Moreover, we would also like to point out that this trade-off between robust and clean accuracies has been a subject of research itself, where efforts like [19,20] have proposed methodologies to narrow the gap between the two accuracies. In this work, our primary goal is to improve the robustness of neural networks while preserving performance on clean data to the extent possible. Further study on reducing the trade-off between robust-clean accuracy in this setting is left as a direction of future work.

Calibration: A recent study [31] showed that DNNs trained with Mixup are significantly better calibrated than DNNs trained in a regular fashion. Calibration [9] measures how good softmax scores are as indicators of the actual likelihood of a correct prediction. We measure the *Expected Calibration Error (ECE)* [9,31] of the proposed method, following [31]: predictions (total N predictions) are grouped into M interval bins (B_m) of equal size. The accuracy and confidence of B_m are defined as:

$$acc(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} 1 \cdot (\hat{y}_i = y_i)$$

and

$$conf(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \hat{p}_i$$

where $\hat{p}_i$, $\hat{y}_i$, y_i are the confidence, predicted label and true label of sample i respectively. The *Expected Calibration Error (ECE)* is then defined as:

$$ECE = \sum_{m=1}^{M} \frac{|B_m|}{N} \cdot \mid acc(B_m) - conf(B_m) \mid \tag{11}$$

Fig. 2 shows the calibration error on CIFAR-10 and CIFAR-100 datasets using Mixup, VarMixup, adv-VarMixup and Mixup + VarMixup. The figure illustrates that our VarMixup models (and their combinations with regular Mixup) are also better calibrated than regular Mixup.

5. Discussion and ablations

Local linearity on loss landscapes: [26] showed that the local linearity of loss landscapes of neural networks is related to model robustness. The more the loss landscapes are linear, the more the adversarial robustness. To further study this observation using our method, we analyze the local linearity of loss landscapes of VarMixup and regular mixup trained models. Qin et al. [26] defines local linearity at a data-point x within a neighbourhood $B(\epsilon)$ as $\gamma(\epsilon, x, y) =$

$$\max_{\delta \in B(\epsilon)} |\mathcal{L}(F_w(x+\delta), y) - \mathcal{L}(F_w(x), y) - \delta^T \nabla_x \mathcal{L}(F_w(x), y)| \tag{12}$$

Fig. 3 shows the average local linear error (over test set) with increasing L_∞ max-perturbation ϵ on CIFAR-10 and CIFAR-100 datasets. As noticeable, VarMixup/*adv*-VarMixup makes the local linear error significantly ($\times$ 2) lesser as compared to regular mixup, thus inducing robustness.

Analyzing VarMixup samples:
Fig. 5 shows sample data generated by regular Mixup, VarMixup, and *adv*-VarMixup on two images. Although mixup or VarMixup samples look perceptually similar, they are quite different at a statistical level. We measure the Frechet Inception Distance (FID) [15] and Kernel Inception Distance [3] between regu-

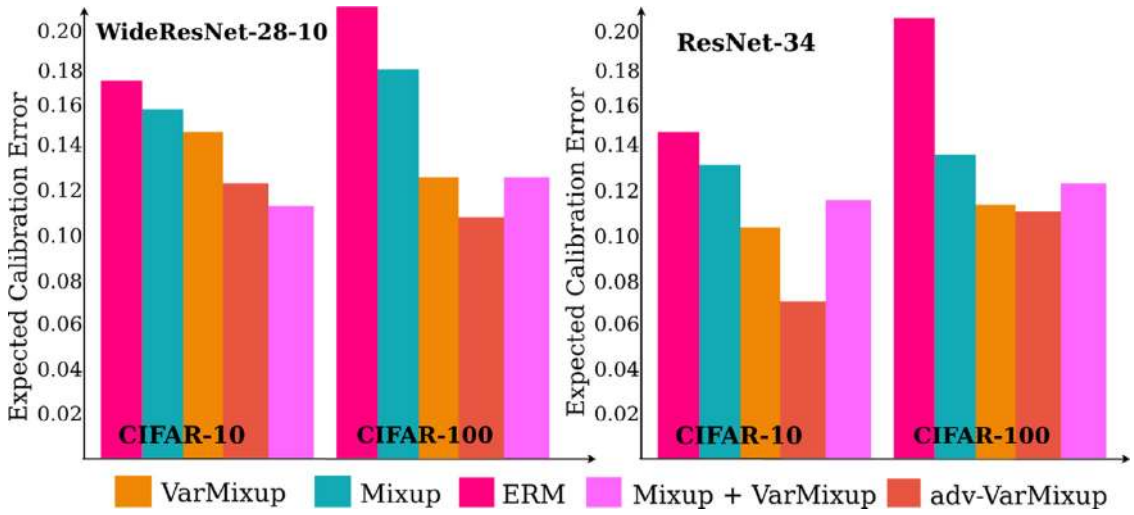

Fig. 2. Expected Calibration Error (ECE) [9] of ERM, Mixup, VarMixup, adv-VarMixup, Mixup + VarMixup trained models.

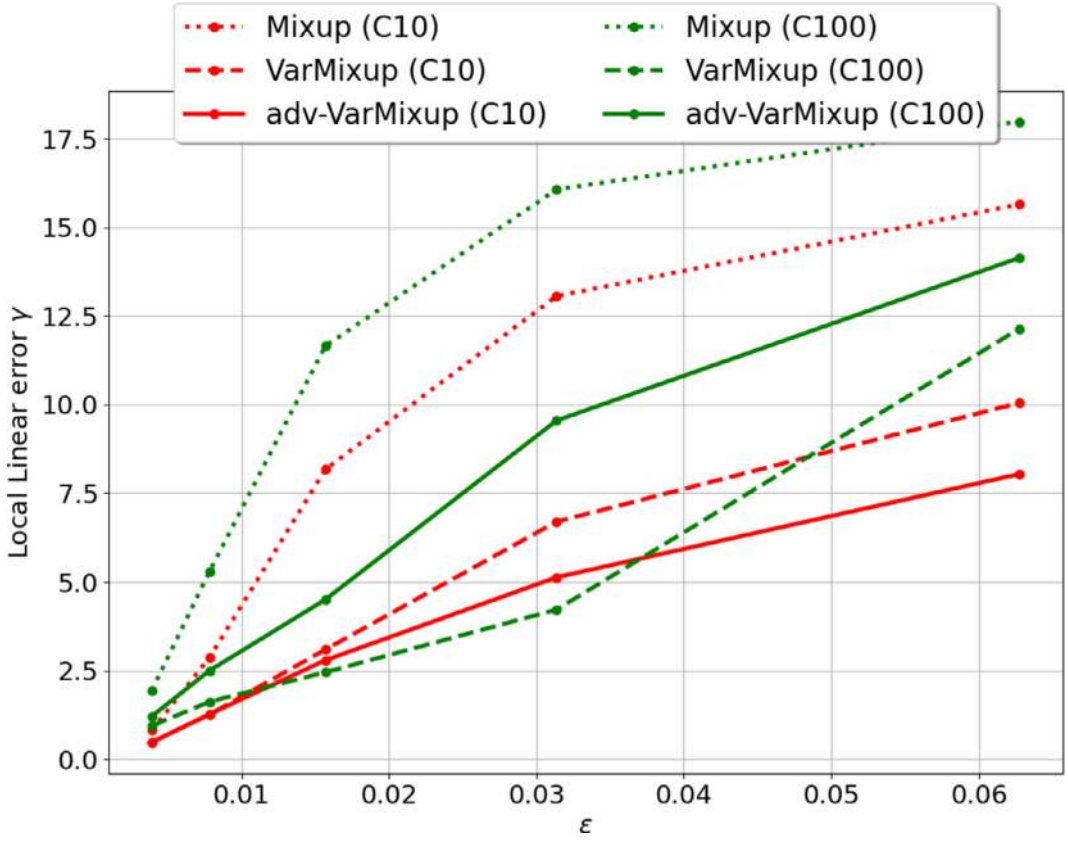

Fig. 3. Local linear error of loss landscapes of the models trained on CIFAR-10/-100 (denoted as C10 and C100).

lar training data and training data generated by mixup/VarMixup/*adv*-VarMixup. These scores summarize how similar the two groups are in terms of statistics on computer vision features of the raw images calculated using the Inceptionv3 model used for image classification.

Lower scores indicate the two groups of images are more similar, or have more similar statistics, with a perfect score being 0.0 indicating that the two groups of images are identical. Fig. 4 reports these metrics on CIFAR-10 and CIFAR-100 respectively. The greater FID and KID scores indicate that we are adding off-manifold samples (w.r.t. the manifold characterized by training data) to the training using our approach.

Computational Overhead: We compare the computational time of our trained models using VarMixup/*adv*-VarMixup with commonly used adversarial training techniques: AT and TRADES. VarMixup, *adv*-VarMixup, AT and TRADES take around 3, 5, 8.8 and 15 hours respectively for training. The training time of the MMD-VAE was also considered here. While already significantly faster than AT and TRADES, the proposed method will be more scalable and time-efficient, if a VAE trained on a dataset such as ImageNet can be directly used to generate VarMixup samples for other datasets. This is a typical transfer learning setting, and we hence study the performance of training VarMixup models on CIFAR-10 and CIFAR-100 datasets using MMD-VAE trained on the

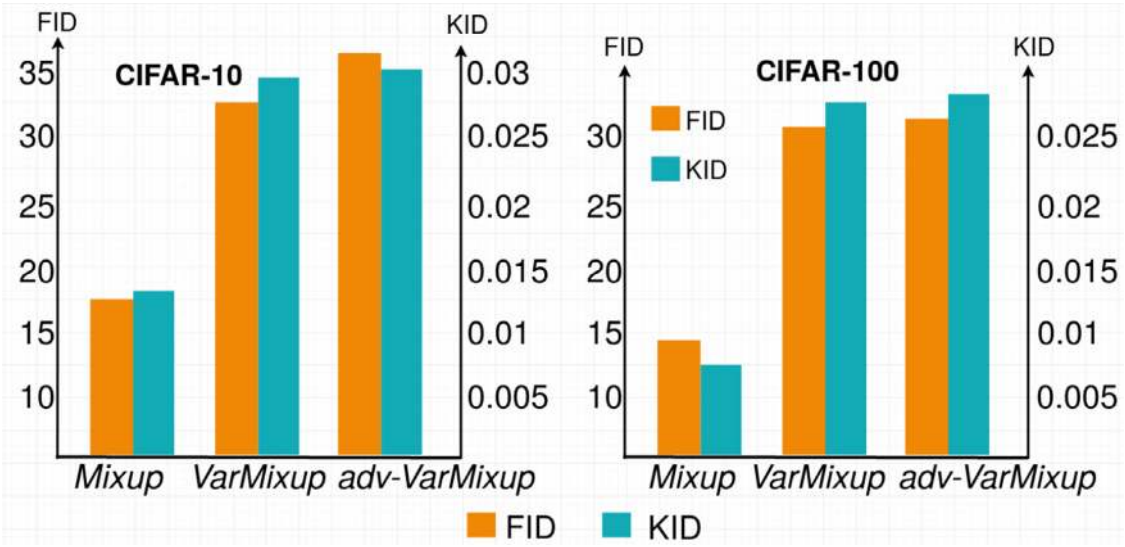

Fig. 4. FID and KID scores between training set and Mixup/VarMixup generated samples on CIFAR-10 and CIFAR-100.

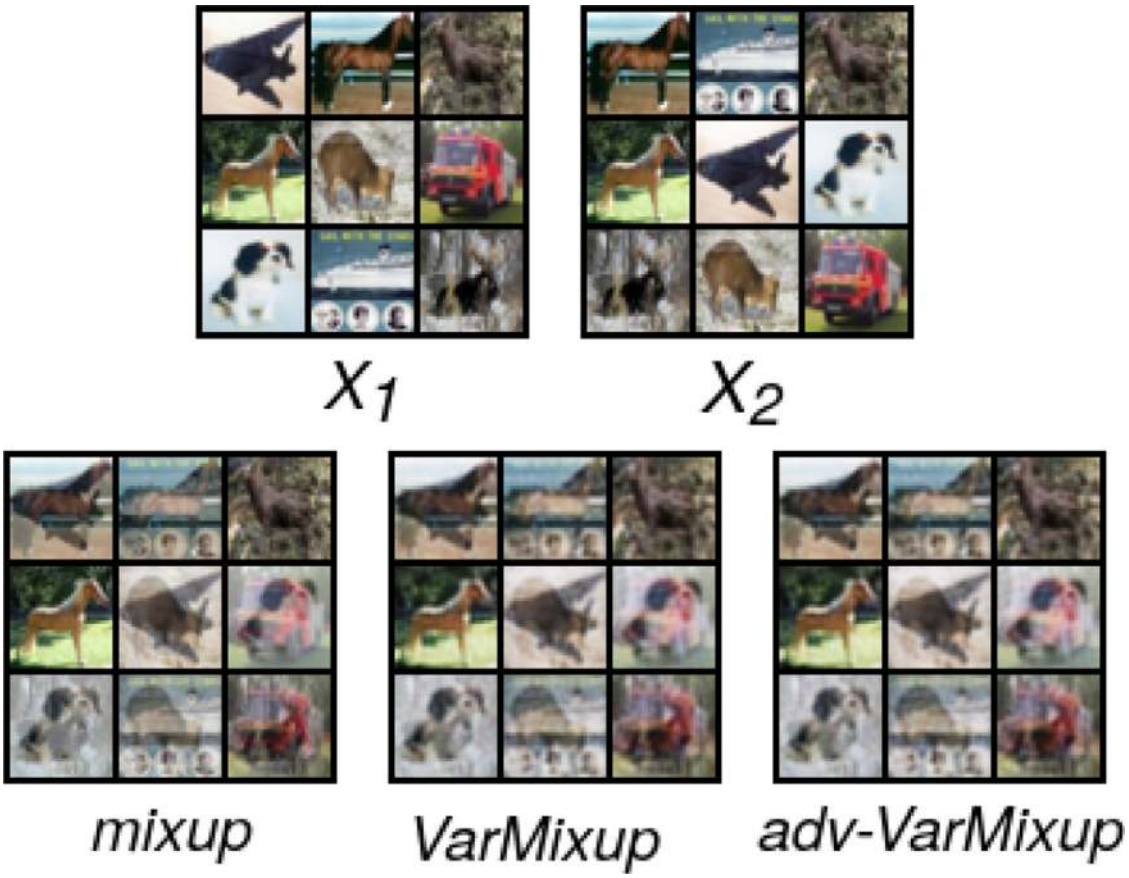

Fig. 5. Samples generated by mixup, VarMixup and *adv*-VarMixup on CIFAR-10 (Mixup coefficient $\lambda = 0.5$).

Tiny-Imagenet dataset. Respectively, VarMixup obtained mean corruption accuracy of 82.0% and 53.25% on CIFAR-10-C and CIFAR-100-C benchmarks, thus making our approach time-efficient and also scalable.

6. Conclusions

In this work, we proposed a Mixup-based vicinal distribution, VarMixup, which performs linear interpolation on an unfolded latent manifold where linearity in between training examples is likely to be preserved by construction. We show that VarMixup trained models are more robust to common input corruptions, are better calibrated and have significantly lower local-linear loss than regular Mixup models. As expected and noted earlier, in some places, we do observe a trade-off between clean and robust accuracy, and leave this as a direction for future works to explore. Additionally, our experiments indicate that VarMixup adds more off-manifold images to training than regular mixup, which we hypoth-

esize is a key reason for the observed robustness. Our work highlights the efficacy of defining vicinal distributions by using neighbors on unfolded latent manifold rather than data manifold and we believe that our work can open a discussion around this notion of robustness and choice of vicinal distributions on generative latent spaces.

Confirmation of Authorship

Please save a copy of this file, complete and upload as the "Confirmation of Authorship" file. As corresponding author I, Puneet Mangla, hereby confirm on behalf of all authors that:

1. This manuscript, or a large part of it, has not been published, was not, and is not being submitted to any other journal.

2. If presented at or submitted to or published at a conference(s), the conference(s) is (are) identified and substantial justification for re-publication is presented below. A copy of conference paper(s) is(are) uploaded with the manuscript.

3. If the manuscript appears as a preprint anywhere on the web, e.g. arXiv, etc., it is identified below. The preprint should include a statement that the paper is under consideration at Pattern Recognition Letters.

4. All text and graphics, except for those marked with sources, are original works of the authors, and all necessary permissions for publication were secured prior to submission of the manuscript.

5. All authors each made a significant contribution to the research reported and have read and approved the submitted manuscript.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowlgedgments

This work has been partly supported by the funding received from DST, Govt of India, through the IMPRINT program (IMP/2019/000250). We also acknowledge IIT-Hyderabad and JICA for provision of GPU servers for the work.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.patrec.2021.10.016

References

[1] C. Beckham, S. Honari, V. Verma, A.M. Lamb, F. Ghadiri, R.D. Hjelm, Y. Bengio, C. Pal, On Adversarial Mixup Resynthesis, in: Advances in Neural Information Processing Systems 32, Curran Associates, Inc., 2019, pp. 4346–4357.
[2] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, C.A. Raffel, Mixmatch: A Holistic Approach to Semi-supervised Learning, in: Advances in Neural Information Processing Systems 32, Curran Associates, Inc., 2019, pp. 5049–5059.
[3] M. BiÅkowski, D.J. Sutherland, M. Arbel, A. Gretton, Demystifying MMD GANs, in: International Conference on Learning Representations, 2018.
[4] Y. Cao, P.I. Rockett, The use of vicinal-risk minimization for training decision trees, Appl. Soft Comput. 31 (C) (2015).
[5] O. Chapelle, J. Weston, L. Bottou, V. Vapnik, Vicinal Risk Minimization, in: T.K. Leen, T.G. Dietterich, V. Tresp (Eds.), Advances in Neural Information Processing Systems 13, MIT Press, 2001, pp. 416–422.
[6] X. Chen, D.P. Kingma, T. Salimans, Y. Duan, P. Dhariwal, J. Schulman, I. Sutskever, P. Abbeel, Variational lossy autoencoder, CoRR abs/1611.02731 (2016).
[7] S. CS231N, Tiny ImageNet Visual Recognition Challenge, https://tiny-imagenet.herokuapp.com/.
[8] A. Gretton, K. Borgwardt, M. Rasch, B. Schölkopf, A.J. Smola, A Kernel Method for the Two-sample-problem, in: B. Schölkopf, J.C. Platt, T. Hoffman (Eds.), Advances in Neural Information Processing Systems 19, MIT Press, 2007, pp. 513–520.
[9] C. Guo, G. Pleiss, Y. Sun, K.Q. Weinberger, On calibration of modern neural networks, CoRR abs/1706.04599 (2017).
[10] L. Hai-Yan, J. Hua, Vicinal risk minimization based probability density function estimation algorithm using svm, in: 2010 Third International Conference on Information and Computing, 4, 2010, pp. 161–164.
[11] N. Harvey, C. Liaw, A. Mehrabian, Nearly-tight vc-dimension bounds for piecewise linear neural networks, CoRR abs/1703.02930 (2017).
[12] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.
[13] D. Hendrycks, T. Dietterich, Benchmarking neural network robustness to common corruptions and perturbations, Proceedings of the International Conference on Learning Representations (2019).
[14] D. Hendrycks*, N. Mu*, E.D. Cubuk, B. Zoph, J. Gilmer, B. Lakshminarayanan, Augmix: A simple method to improve robustness and uncertainty under data shift, in: International Conference on Learning Representations, 2020.
[15] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, G. Klambauer, S. Hochreiter, Gans trained by a two time-scale update rule converge to a nash equilibrium, CoRR abs/1706.08500 (2017).
[16] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, A. Madry, Adversarial Examples Are Not Bugs, They Are Features, in: Advances in Neural Information Processing Systems 32, Curran Associates, Inc., 2019, pp. 125–136.
[17] D.P. Kingma, M. Welling, Auto-encoding variational bayes, CoRR abs/1312.6114 (2013).
[18] A. Krizhevsky, Learning multiple layers of features from tiny images, 2009.
[19] A. Lamb, V. Verma, J. Kannala, Y. Bengio, Interpolated adversarial training: Achieving robust neural networks without sacrificing too much accuracy, 2019. AISec'19
[20] S. Lee, H. Lee, S. Yoon, Adversarial vertex mixup: Toward better adversarially robust generalization, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020.
[21] X. Liu, Y. Zou, L. Kong, Z. Diao, J. Yan, J. Wang, S. Li, P. Jia, J. You, Data augmentation via latent space interpolation for image classification, in: 2018 24th International Conference on Pattern Recognition (ICPR).
[22] A. Look, O. Kirschner, S. Riedelbauch, Building robust classifiers with generative adversarial networks for detecting cavitation in hydraulic turbines, ICPRAM, 2018.
[23] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, A. Vladu, Towards deep learning models resistant to adversarial attacks, in: International Conference on Learning Representations, 2018.
[24] J. Ni, P.I. Rockett, Training genetic programming classifiers by vicinal-risk minimization, Genet. Program. Evolvable Mach. 16 (1) (2015) 3–25.
[25] T. Pang*, K. Xu*, J. Zhu, Mixup inference: Better exploiting mixup to defend adversarial attacks, in: International Conference on Learning Representations, 2020.
[26] C. Qin, J. Martens, S. Gowal, D. Krishnan, A. Fawzi, S. De, R. Stanforth, P. Kohli, et al., Adversarial robustness through local linearization, arXiv preprint arXiv:1907.02610 (2019).
[27] P.Y. Simard, Y.A. LeCun, J.S. Denker, B. Victorri, Transformation Invariance in Pattern Recognition – Tangent Distance and Tangent Propagation, Springer Berlin Heidelberg.
[28] C.K. Sønderby, T. Raiko, L. Maaløe, S.K. Sønderby, O. Winther, Ladder variational autoencoders, in: Proceedings of the 30th International Conference on Neural Information Processing Systems, Curran Associates Inc., Red Hook, NY, USA, 2016, pp. 3745–3753. NIPS'16
[29] C.K. Sønderby, T. Raiko, L. Maaløe, S.K. Sønderby, O. Winther, Ladder variational autoencoders, in: Proceedings of the 30th International Conference on Neural Information Processing Systems, Curran Associates Inc., Red Hook, NY, USA, 2016, pp. 3745–3753. NIPS'16
[30] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, R. Fergus, Intriguing properties of neural networks, arXiv preprint arXiv:1312.6199 (2013).
[31] S. Thulasidasan, G. Chennupati, J.A. Bilmes, T. Bhattacharya, S. Michalak, On Mixup Training: Improved Calibration and Predictive Uncertainty for Deep Neural Networks, in: Advances in Neural Information Processing Systems 32, Curran Associates, Inc., 2019, pp. 13888–13899.
[32] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, A. Madry, Robustness May Be at Odds with Accuracy, arXiv e-prints.
[33] V.N. Vapnik, The nature of statistical learning theory, Springer-Verlag, Berlin, Heidelberg, 1995.
[34] V.N. Vapnik, Statistical learning theory, Wiley-Interscience, 1998.
[35] V. Verma, A. Lamb, C. Beckham, A. Najafi, I. Mitliagkas, D. Lopez-Paz, Y. Bengio, Manifold mixup: Better representations by interpolating hidden states, in: Proceedings of the 36th International Conference on Machine Learning, 2019, pp. 6438–6447.
[36] H. Wang, C.-N. Yu, A direct approach to robust deep learning using adversarial networks, in: International Conference on Learning Representations, 2019.
[37] M. Xu, J. yu Zhang, B. Ni, T. Li, C. Wang, Q. Tian, W. Zhang, Adversarial domain adaptation with domain mixup, ArXiv abs/1912.01805 (2019).
[38] X. Yin, S. Kolouri, G.K. Rohde, Gat: Generative adversarial training for adversarial example detection and robust classification, in: International Conference on Learning Representations, 2020.
[39] X. Yuan, P. He, Q. Zhu, R.R. Bhat, X. Li, Adversarial examples: attacks and defenses for deep learning, CoRR abs/1712.07107 (2017).
[40] S. Zagoruyko, N. Komodakis, Wide residual networks, BMVC, 2016.

[41] C. Zhang, S. Bengio, M. Hardt, B. Recht, O. Vinyals, Understanding deep learning requires rethinking generalization, ICLR abs/1611.03530 (2017).
[42] C. Zhang, M.-H. Hsieh, D. Tao, Generalization bounds for vicinal risk minimization principle, arXiv preprint arXiv:1811.04351 (2018).
[43] H. Zhang, M. Cissé, Y.N. Dauphin, D. Lopez-Paz, Mixup: beyond empirical risk minimization, CoRR abs/1710.09412 (2017).
[44] H. Zhang, Y. Yu, J. Jiao, E.P. Xing, L.E. Ghaoui, M.I. Jordan, Theoretically principled trade-off between robustness and accuracy, CoRR abs/1901.08573 (2019).
[45] S. Zhao, J. Song, S. Ermon, Infovae: information maximizing variational autoencoders, CoRR abs/1706.02262 (2017).