Patch-level Neighborhood Interpolation: A General and Effective Graph-based Regularization Strategy

Abstract

Regularization plays a crucial role in machine learning models, especially for deep neural networks. The existing regularization techniques mainly rely on the i.i.d. assumption and only consider the knowledge from the current sample, without the leverage of the neighboring relationship between samples. In this work, we propose a general regularizer called **Patch-level Neighborhood Interpolation (Pani)** that conducts a non-local representation in the computation of networks. Our proposal explicitly constructs patch-level graphs in different layers and then linearly interpolates neighborhood patch features, serving as a general and effective regularization strategy. Further, we customize our approach into two kinds of popular regularization methods, namely Virtual Adversarial Training (VAT) and MixUp as well as its variants. The first derived **Pani VAT** presents a novel way to construct non-local adversarial smoothness by employing patch-level interpolated perturbations. The second derived **Pani MixUp** method extends the MixUp, and achieves superiority over MixUp and competitive performance over state-of-the-art variants of MixUp method with a significant advantage in computational efficiency. Extensive experiments have verified the effectiveness of our Pani approach in both supervised and semi-supervised settings.

Keywords: Graph-based Regularization, Interpolation, (Semi-)Supervised Learning.

1. Introduction

In the statistical learning theory, regularization techniques are typically leveraged to achieve the trade-off between empirical error minimization and the control of model complexity (Vapnik and Chervonenkis, 2015). In contrast to the classical convex empirical risk minimization where regularization can rule out trivial solutions, regularization plays a rather different role in deep learning due to its highly non-convex optimization nature (Zhang et al., 2016). Among all the explicit and implicit regularization, regularization with stochastic transformation, perturbations and randomness, such as adversarial training (Goodfellow et al., 2014), dropout and MixUp (Zhang et al., 2017), play a key role in the deep learning models due to their superiority in the performance (Berthelot et al., 2019b; Zhang et al., 2017; Miyato et al., 2018; Berthelot et al., 2019a; Yao et al., 2022; Kim et al., 2021).

Nevertheless, the vast majority of regularization methods, including the aforementioned ones, assume that the training samples are drawn independently and identically from an unknown data-generating distribution. However, this i.i.d. assumption is commonly violated in realistic scenarios where batches or sub-groups of training samples are likely to have internal correlations. Accounting for the correlations in real-world training data can lead to statistically significant improvements in accuracy (Dundar et al., 2007; Yun et al., 2019) and other benefits, including the robustness (Svoboda et al., 2018; Yao et al., 2022). For example, Peer-Regularized Networks (PeerNet) (Svoboda et al., 2018) applied graph convolutions (Velickovic et al., 2017; Kipf and Welling, 2016) to harness information of peer samples, and verified its effectiveness on defending adversarial attacks. Motivated by these facts, we design a general regularization strategy called Patch-level Neighborhood Interpolation (Pani) that can fully utilize the internal relationship between samples by explicitly constructing a graph within a mini-batch in order to consistently improve the generalization of deep neural networks in both semi- and supervised settings. Our contributions can be summarized as folllows:

- We propose a simple yet general effective non-local regularization called **Patch-level Neighborhood Interpolation (Pani)**. Pani linearly interpolates on the neighboring patch features and yields a non-local representation that additionally captures the relationship of neighboring patch features in different layers.

- We explicitly customize our Pani approach into two classes of popular and general regularization strategies, i.e., Virtual Adversarial Regularization and MixUp, resulting in **Pani VAT** and **Pani MixUp**. Pani VAT yields non-local adversarial perturbations, providing a more informative adversarial smoothness in the semi-supervised learning setting. Pani MixUp and MixMatch perform better than their vanilla versions with computational efficiency by mixing fine-grained patch features and supervised signals.

- Extensive experiments demonstrate that both of the two derived regularization strategies can outperform other state-of-the-art approaches in both supervised and semi-supervised tasks, presenting the generality and superiority of our Pani method.

1.1. Related Work

Virtual Adversarial Training (VAT) and Its Variants. Adversarial Training (Goodfellow et al., 2014; Madry et al., 2017; Zhang et al., 2019; Tsipras et al., 2018) can provide a new form of regularization beyond that provided by other generic regularization strategies, such as dropout, pretraining and model averaging. VAT (Miyato et al., 2018) extends the adversarial training through adversarially smoothing the posterior output distribution with the leverage of unlabeled data, achieved great success in image classification (Miyato et al., 2018), text classification (Miyato et al., 2016) and node classification (Sun et al., 2019). There is a flurry of VAT variants (Luo et al., 2017; Yu et al., 2019), most of which heavily rely on generative models to construct data manifold. Tangent-Normal Adversarial Regularization (TNAR) (Yu et al., 2019) extended VAT by considering the data manifold and applied VAT along the tangent space and the orthogonal normal space of the data manifold, outperforming previous semi-supervised approaches. VAT+SNTG (Luo et al., 2017) constructed a graph based on the predictions of the teacher model to smooth the representation on the low-dimensional manifold in the semi-supervised setting. By contrast, our Pani method is a fine-grained patch-level and more general regularization that can be leveraged to refine the representation of deep neural networks in both semi- and supervised scenarios *without the requirement of any generative model.*

MixUp and Its Variants. MixUp (Zhang et al., 2017) has been widely used in various machine learning tasks. However, MixUp samples tend to be unnatural and locally ambiguous, which may confuse the model (Yun et al., 2019). To overcome this issue, CutMix (Yun et al., 2019) replaces the image region with a patch from another training sample in a more straightforward way. MixUp variants, including Manifold MixUp (Verma et al., 2019), AdaMixUp (Guo et al., 2019), SmoothMix (Lee et al., 2020), Puzzle Mix (Kim et al., 2020) and Co-MixUp (Kim et al., 2021), have been proposed through various kinds of interpolation and transformations. In addition, C-MixUp (Yao et al., 2022) extends MixUp into the regression setting, investigating both its improvement on in-distribution generalization and out-of-distribution robustness. In semi-supervised learning, FixMatch (Sohn et al., 2020) and MixMatch (Berthelot et al., 2019b) serve as a natural extension of MixUp and achieve state-of-the-art accuracy. In contrast with VAT, MixMatch (Berthelot et al., 2019b) utilizes one specific form of consistency regularization, i.e., using the standard data augmentation for images, such as random horizontal flips, rather than computing adversarial perturbations to smooth the posterior distribution of the classifier. Our Pani method is proposed differently from other MixUp variants, and shares similarities with CutMix. However, CutMix randomly cut patches, while our Pani linearly interpolates patch features and is more general. Pani Method can also easily enhance the performance of MixMatch in the semi-supervised setting.

Manifold Regularization. There is a flurry of papers introducing regularization from classical manifold learning based on the assumption that the data can be modeled as a low-dimensional manifold in the data space. As demonstrated in(Hinton et al., 2012; Ioffe and Szegedy, 2015), regularizers that work well in the input space can also be applied to the hidden layers of a deep network, which could further improve the generalization performance. Our Patch-level Neighborhood Interpolation can be easily extended from input to the hidden layers, enjoying the benefits of manifold regularization.

Non-local Image Filtering. Past non-local image filter methods (Tomasi and Manduchi, 1998; Buades et al., 2005; Sochen et al., 1998) leveraged both the pixel intensities and their pixel neighbors together with their locations to design these non-shift-invariant filters. Recently, Non-local Neural Networks (Wang et al., 2018) presented one effective non-local operation that serves as a generic component for capturing long-range dependencies with deep neural networks. Similarly, our Pani still can capture the correlation knowledge of patch features within a batch, therefore yielding an improvement in performance for the derived methods. Moreover, our method also serves as a novel non-i.i.d. regularization and can reasonably generalize well to broader settings especially when the natural correlation in the sub-group exists.

2. Preliminary

Virtual Adversarial Training (VAT). VAT replaces true labels y of samples in the formulation of adversarial training by current estimate $p(y|x;\hat{\theta})$ from the model:

$$\min_{\theta} \max_{r,\|r\|\leq\epsilon} D\left[p(y|x;\hat{\theta}), p(y|x+r;\theta)\right], \tag{1}$$

where $D[q,p]$ measures the divergence between two distributions q and p. r is the adversarial perturbation depending on the current sample x that can further provide smoothness in SSL. Then the VAT regularization $\mathcal{R}_{\text{vadv}}(x,r;\theta)$ is derived from the inner maximization:

$$\mathcal{R}_{\text{vadv}}(x,r;\theta) = \max_{r,\|r\|\leq\epsilon} D\left[p(y|x;\hat{\theta}), p(y|x+r;\theta)\right] \tag{2}$$

One elegant part of VAT is that it utilized the second-order Taylor's expansion of virtual adversarial loss to compute the perturbation r, which can be computed efficiently by power iteration with finite differences. Once the desired perturbation r^* has been obtained, we can conduct forward and back propagation to optimize the full loss function:

$$\min_{\theta} \mathcal{L}_0 + \beta \mathbb{E}_{x \sim \mathcal{D}} \mathcal{R}_{\text{vadv}}(x, r^*; \theta), \tag{3}$$

where $\mathcal{L}_0$ is the original supervised loss and β is the hyper-parameter to control the degree of virtual adversarial smoothness.

MixUp. MixUp (Zhang et al., 2017) augments the training data with linear interpolation on both input features and target. The resulting feature-target vectors are shown as follows:

$$\tilde{x} = \lambda x_i + (1-\lambda) x_j, \quad \tilde{y} = \lambda y_i + (1-\lambda) y_j, \tag{4}$$

where (x_i, y_i) and (x_j, y_j) are two feature-target vectors drawn randomly from the training data. $\lambda \sim \text{Beta}(a, a)$ and $a \in (0, \infty)$. MixUp can be understood as a form of data augmentation that encourages decision boundaries to transit linearly between classes. It is a kind of generic regularization that provides a smoother estimate of uncertainty, yielding the improvement of generalization.

Peer-Regularized Networks (PeerNet). The centerpiece of PeerNet (Svoboda et al., 2018) is the learnable *Peer Regularization* (PR) layer designed to focus on improving the adversarial robustness of deep neural networks. PR layer can be flexibly added to the feature maps of deep models. Let $\mathbf{Z}^1, ..., \mathbf{Z}^N$ be $n \times d$ matrices as the feature maps of N images, where n is the number of pixels and d represents the dimension of the feature in each pixel, i.e., number of channel in the feature map. The core of PeerNet is to find the K nearest neighboring pixels for each pixel among all the pixels of N peer images via constructing a K nearest neighbor graph in the d-dimensional space. Particularly, for the p-th pixel in the i-th image $\mathbf{z}_p^i$, the k-th nearest pixel neighbor can be denoted as $\mathbf{z}_{q_k}^{j_k}$ taken from the pixel q_k of the peer image j_k. Then the learnable PR layer is constructed by a variant of Graph Attention Networks (GAT) (Velickovic et al., 2017):

$$\tilde{\mathbf{z}}_p^i = \sum_{k=1}^{K} \alpha_{ij_kpq_k} \mathbf{z}_{q_k}^{j_k}, \quad \alpha_{ij_kpq_k} = \frac{\text{LeakyReLU}\left(\exp\left(f_a\left(\mathbf{z}_p^i, \mathbf{z}_{q_k}^{j_k}\right)\right)\right)}{\sum_{k'=1}^{K} \text{LeakyReLU}\left(\exp\left(f_a\left(\mathbf{z}_p^i, \mathbf{z}_{q_{k'}}^{j_{k'}}\right)\right)\right)}, \tag{5}$$

where $\alpha_{ij_kpq_k}$ is the attention score determining the importance of the q_k-th pixel of the j-th peer image on the representation of current p-th pixel $\tilde{\mathbf{z}}_p^i$ taken from the i-th image. $f_a()$ is a fully connected layer mapping from $2d$-dimensional input to scalar output. Therefore, the resulting learnable PR layer involves non-local filtering by leveraging the wisdom of pixel neighbors from peer images, showing robustness against adversarial attacks.

3. Our Method: Patch-level Neighborhood Interpolation

For our method, one related work is PeerNet (Svoboda et al., 2018) that designed graph-based layers to defend against adversarial attacks, but unfortunately, the construction of pixel-level K-NN graphs in PeerNet (Svoboda et al., 2018) is costly in computation. By

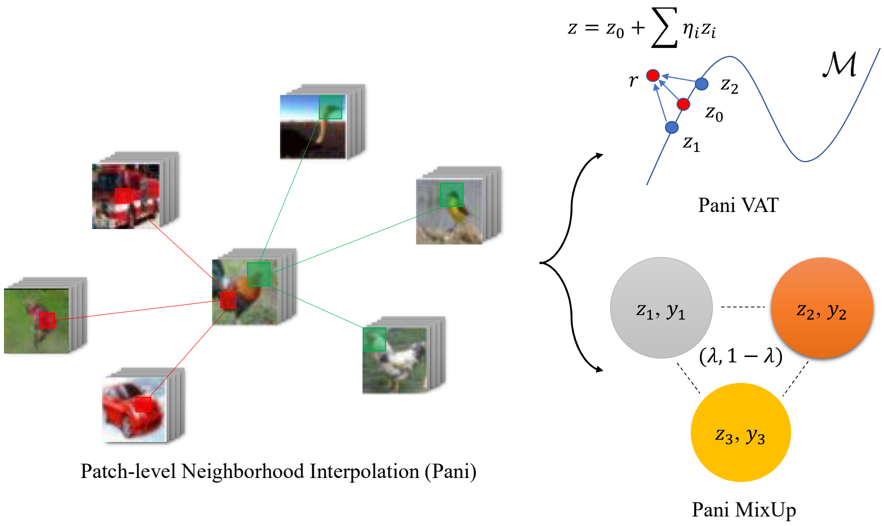

Figure 1: Pipeline of our Patch-level Neighborhood Interpolation followed by two derived regularizations, i.e., Pani VAT and Pani MixUp. r represents the perturbation constructed by our method and $(\lambda, 1-\lambda)$ is the mixing coefficient pair.

contrast, our motivation is to develop a general regularization that can consistently boost the performance of deep neural networks in both semi- and supervised settings rather than the adversarial scenario. Besides, the construction way of a non-local layer in our method is more flexible and can be determined by the specific objective function, as elaborated in the following part. Moreover, our patch-level method can achieve the computational advantage over pixel-level regularization, and incorporates more meaningful semantic correlations in different layers. Particularly, a flexible patch size can be chosen according to the size of the receptive field in different layers, yielding a more informative graph-based representation and better regularization performance.

Step 1: Construction of K nearest neighbor patch graphs. As illustrated in Figure 1, in the first step of Patch-level Neighborhood Interpolation (Pani) we determine the candidate peer images set $\mathcal{S}_i$ for each image i. This can be achieved by random matching or computing the semantically nearest image neighbors using e.g. the cosine distance. Next, we construct the whole patches set $\mathcal{P}_i$ on the candidate peer images set $\mathcal{S}_i$ for each image i by clipping the corresponding patches in the different locations on an input or a feature map. Following the establishment of patch set $\mathcal{P}_i$, we construct K nearest neighbor patch graphs based on the distance of patch features in order to find the neighbors of each patch in a patch set $\mathcal{P}_i$ for $\forall i = 1, .., N$. Mathematically, following the definition in the PeerNet, let $\mathbf{z}_p^i$ be the p-th patch on the input or feature map $\mathbf{Z}^i$ for the i-th image within one batch. Then denote the k-th nearest patch neighbor for $\mathbf{z}_p^i$ as $\mathbf{z}_{q_k}^{j_k}$ taken from the patch q_k of the peer image j_k in the candidate set $\mathcal{S}_i$.

Step 2: Linear interpolation. Next, in order to leverage the knowledge from neighbors, different from the graph attention mechanism in PeerNet, we apply a more straightforward linear interpolation on the neighboring patches for the current patch $\mathbf{z}_p^i$. Then, the general formulation of our Patch-level Neighborhood Interpolation can be presented as follows:

$$\tilde{\mathbf{z}}_p^i = \mathbf{z}_p^i + \sum_{k=1}^{K} \eta_{ipk}(\mathbf{z}_{q_k}^{j_k} - \mathbf{z}_p^i), \tag{6}$$

where η_{ipk} is the combination coefficient for the p-th patch of i-th image w.r.t its k-th patch neighbor, which can be computed through the power iteration similar to the manner of VAT, or through random sampling from a specific distribution in randomness-based regularization, e.g., MixUp and its variants. Moreover, the choice of linear interpolation in Eq. 6 enjoys a computational advantage over the nonlinear GAT form in PeerNet in the computation of networks. Finally, after the patch-level linear interpolation on patch features, we can obtain the refined graph-based representation $\tilde{\mathbf{Z}}^i$ for i-th image, $\forall i = 1, ..., N$. Note that our proposed method can explicitly combine the advantages of manifold regularization and non-local filtering in a flexible way. To further demonstrate the generality and effectiveness of our Pani method, we propose the Pani version of two typical regularization strategies, i.e., VAT and MixUp as well its variant MixMatch.

3.1. Pani VAT

Based on our Pani framework, we can construct a novel Pani VAT that utilizes the linear interpolation of patch neighbors for each sample to manipulate the non-local perturbations, thus providing more informative adversarial smoothness in the semi-supervised setting. Consider a more general composite function form of the classifier f, i.e.,$f(x) = g(z)$ and $z = h(x)$ where z denotes the hidden feature of input x or the input itself when the reduced form happens. Combining VAT formulation Eq. 2, and Pani formulation Eq. 6, we reformulate Pani VAT with perturbations on L layers in a deep neural network as follows:

$$\max_{\eta} D[g(z), g(\tilde{z}(\eta))] \quad s.t. \quad \sum_{l=1}^{L} w_l^2 \|\eta^{(l)}\|^2 \leq \epsilon^2, \tag{7}$$

where D measures the divergence between two distributions. $\eta = \{\eta_{ipk}\}$ denotes the generic perturbations from our Pani method and $\eta^{(l)}$ indicates the perturbations in l-th layer of

Algorithm 1 Pani VAT within a Batch

```
Input: neighbors $K_1$, $K_2$, classifier $f$, batch size $B$, perturbed layers $L$
Initialization: combination coefficient $\eta$
Compute $K_1$ nearest image neighbors based on the distance of the second last layer
  output from $f$ and obtain $K_1$ ($K_1 \leq B$) peer images set $\mathcal{S}_i$ for each image $i$.
for $l = 1$ to $L$ do:
    Compute the patch set $\mathcal{P}_i$ for all $K_1$ peer images on layer $l$ for each image $i$ .
    Construct a $K_2$ nearest patch neighbors graph for each patch in each image $i$.
    Conduct Patch-level Neighborhood Interpolation via Eq. 6 for each patch.
end for
Conduct power iteration and finite difference to compute $\eta^*$ constrained by Eq. 7.
Return $\mathcal{R}_{\text{vadv}}(x, \eta^*; \theta)$
```

network. $\tilde{z}(\eta) = \{\tilde{\mathbf{z}}_p^i\}$ represents the smoothed feature map imposed by perturbation η considering all patches in the way shown in Eq. 6. In particular, when $L = 1$, adversarial perturbations are only imposed on the input feature, which is similar to the traditional virtual adversarial perturbations. Additionally, w_l is the hyper-parameter, adjusting the weight of perturbation $\eta^{(l)}$ in different layers with the overall perturbations restrained in an ϵ-ball. Next, we still utilize the similar power iteration and finite difference proposed in VAT to compute the desired perturbation η^*. The resulting full loss function is defined as:

$$\min_\theta \mathcal{L}_0 + \beta \mathbb{E}_{x\sim\mathcal{D}} \mathcal{R}_{\text{vadv}}(x, \eta^*; \theta), \tag{8}$$

where $\mathcal{L}_0$ is the original supervised loss and β controls the degree of adversarial smoothness. $\mathcal{R}_{\text{vadv}}(x, y, \eta^*) = D[g(z), g(\tilde{z}(\eta^*))]$ can be attained after solving the optimization problem in Eq. 7. For the implementation details, we describe them in Algorithm 1.

Remark. As shown in the adversarial part of Figure 1, the rationality of our Pani VAT method lies in the fact that the constructed perturbations can entail more non-local information coming from neighbors of the current sample. Through the delicate patch-level interpolation among neighbors of each patch, the resulting non-local virtual adversarial perturbations are expected to provide more informative smoothness, thus enhancing the performance of the classifier in the semi-supervised setting.

3.2. Pani MixUp

Next, we leverage Patch-level Neighborhood Interpolation to derive Pani MixUp. The core formulation of Pani MixUp can be written as:

$$\begin{aligned} \tilde{\mathbf{z}}_p^i &= (1 - \sum_{k=1}^{K} \eta_{ipk}) \mathbf{z}_p^i + \sum_{k=1}^{K} \eta_{ipk} \mathbf{z}_{q_k}^{j_k} \\ \tilde{y}_i &= (1 - \sum_{k=1}^{K}\sum_{p=1}^{P} \frac{\eta_{ipk}}{P}) y_i + \sum_{k=1}^{K}\sum_{p=1}^{P} \frac{\eta_{ipk}}{P}\ y_{j_k}, \quad s.t.\ \lambda = 1 - \sum_{k=1}^{K}\sum_{p=1}^{P} \frac{\eta_{ipk}}{P}, \end{aligned} \tag{9}$$

where $(\mathbf{z}^i, y^i)$ are the feature-target pairs randomly drawn from the training data. P is the number of patches in each image and $\lambda \sim \text{Beta}(a, b)$ represents the importance of the current input or target while conducting MixUp. To compute η_{ipk}, we firstly sample λ from $\text{Beta}(a, b)$ and η_{ipk}^0 from a uniform distribution respectively, then we normalize η_{ipk}^0 according to the ratio of λ to satisfy the constraint in Eq. 9 and thus obtain η_{ipk}. It should be noted that due to the unsymmetric property of λ in our framework, we should tune both a and b in our experiments. For simplicity, we fix $b = 1$ and only consider the a as the hyper-parameter to pay more attention to the importance of the current patch, which is inspired by the similar approach in MixMatch (Berthelot et al., 2019b). Here we reformulate Eq. 9 to illustrate that Pani MixUp is naturally derived from our Pani framework by additionally considering several constraints:

$$\begin{aligned} &\tilde{\mathbf{z}}_p^i = \mathbf{z}_p^i + \sum_{k=1}^{K} \eta_{ipk}(\mathbf{z}_{q_k}^{j_k} - \mathbf{z}_p^i) \\ &s.t.\ \lambda = 1 - \sum_{k=1}^{K}\sum_{p=1}^{P} \frac{\eta_{ipk}}{P}, \forall i = 1, ..., N, \lambda \sim \mathrm{Beta}(a, b), \eta_{ipk} \in [0, 1], \forall i, p, k \end{aligned} \tag{10}$$

where the first constraint in Eq. 10 can be achieved through normalization via λ. Meanwhile, we impose $\eta_{ipk} \in [0, 1]$ as η_{ipk} represents the interpolation coefficient. Further, we elaborate on the procedure of Pani MixUp in Algorithm 2.

Algorithm 2 Pani MixUp within a Batch

```
Input: neighbors $K$, classifier $f$, batch size $B$, perturbed layers $L$, parameter $a$
Compute peer images by random matching and obtain peer images set $\mathcal{S}_i$ for each image $i$.
for $l = 1$ to $L$ do:
    Compute the patch set $\mathcal{P}_i$ on layer $l$ for each image $i$.
    Construct a $K$ nearest patch neighbors graph for each patch in each image $i$.
    Sample initial coefficient $\eta_0^{(l)} = \{\eta_{ipk}^0\}$ from $U(0, 1)$ and $\lambda$ from Beta$(a, 1)$.
    Normalize $\eta_0^{(l)}$ according to the ratio $\lambda$ via Eq. 10 to compute $\eta^{(l)}$.
    Conduct Pani MixUp over patch features and labels via Eq. 10 for each patch.
end for
Return supervised loss based on mixed features and labels.
```

Remark. Different from the role of η in the aforementioned Pani VAT where η serves as the interpolated perturbations, the physical meaning of η in our Pani MixUp approach is the linear interpolation coefficient to conduct MixUp. Despite this distinction, both of the two extended regularization methods are naturally derived from our Pani framework, further demonstrating the generality and superiority of our Pani strategy.

4. Experiments

In this section, we conduct extensive experiments for Pani VAT and Pani MixUp and its variant Pani MixMatch on both semi- and supervised settings.

4.1. Pani VAT

Implementation Details. For fair comparison with VAT and its variants, e.g., VAT + SNTG (Luo et al., 2017) and TNAR (Yu et al., 2019), we choose the standard large convolutional network as the classifier as in (Miyato et al., 2018). For the option of dataset, we focus on the standard semi-supervised setting on CIFAR-10 with 4,000 labeled data. Unless otherwise noted, all the experimental settings in our method are the identical with those in the Vanilla VAT (Miyato et al., 2018). In particular, we conduct our Pani VAT on input layer and one additional hidden layer, yielding two variants Pani VAT (input)

Method	CIFAR-10 (4,000 labels)
VAT (Miyato et al., 2017)	13.15 ± 0.2
VAT + SNTG (Luo et al., 2017)	12.49 ± 0.36
Π model (Laine and Aila, 2016)	16.55 ± 0.29
Mean Teacher (Tarvainen and Valpola, 2017)	17.74 ± 0.30
CCLP (Kamnitsas et al., 2018)	18.57 ± 0.41
ALI (Dumoulin et al., 2016)	17.99 ± 1.62
Improved GAN (Salimans et al., 2016)	18.63 ± 2.32
Tripple GAN (Li et al., 2017)	16.99 ± 0.36
Bad GAN (Dai et al., 2017)	14.41 ± 0.30
LGAN (Qi et al., 2018)	14.23 ± 0.27
Improved GAN + JacobRegu + tangent (Kumar et al., 2017)	16.20 ± 1.60
Improved GAN + ManiReg (Lecouat et al., 2018)	14.45 ± 0.21
TNAR (with generative models) (Yu et al., 2019)	12.06 ± 0.35
Pani VAT (input)	12.33 ± 0.091
Pani VAT (+hidden)	$\mathbf{11.98 \pm 0.106}$

Table 1: Classification errors (%) of compared methods on CIFAR-10 dataset without data augmentation. The results of our Pani methods are the average ones under 4 runs.

and Pani VAT (+hidden). *For the option of hyper-parameters, we conduct the delicate line search for the best performance.* In Pani VAT (input), we choose patch size as 2, $K_1 = 10$ for the number of peer images, $K_2 = 10$ to construct the nearest patch neighbor graph, perturbation size ϵ and adjustment coefficient w_1 as 2.5 and 1.0, respectively. For our Pani VAT (+hidden) method, we opt $K_1 = 10$ and overall perturbation size $\epsilon = 2.1$. On the considered two layers, we choose K_2 as 10 and 50, patch size as 2 and 1 and the adjustment coefficient w as 1 and 4, respectively.

Our Results. Table 1 presents the state-of-the-art performance achieved by Pani VAT (+hidden) compared with other baselines on CIFAR-10. We focus on the baseline methods, especially along the direction of variants of VAT and refer to the results from TNAR (with generative models) method (Yu et al., 2019), the previous state-of-the-art variant of VAT that additionally leverages the data manifold by generative models to decompose the directions of virtual adversarial smoothness. It is worthy to remark that the performance of relevant GAN-based approaches, such as Localized GAN (LGAN) (Qi et al., 2018) as well as TNAR (with generative models) in Table 1, heavily rely on the established data manifold by the generative models. It is well-known that one might come across practical difficulties while implementing and deploying these generative models. By contrast, **without the requirement of generative models**, our approach can eliminate this disturbance and can still outperform these baselines. In addition, our Pani VAT (+hidden) achieves slight improvement compared with Pani VAT (input), which serves as an ablation study, and thus verifies the superiority of manifold regularization mentioned in our Pani framework part. Overall, the desirable flexibility along with desirable stability (lower standard deviation shown in Table 1) of Pani VAT further demonstrates the effectiveness of our Pani strategy.

Analysis of Computational Cost. Another noticeable advantage of our approach is the negligible increase in computation cost compared with Vanilla VAT. In particular, one crucial operation in our approach is the construction of patch set $\mathcal{P}$ and it can be accomplished efficiently by *as.strided* function in Python or through the specific convolution operation in

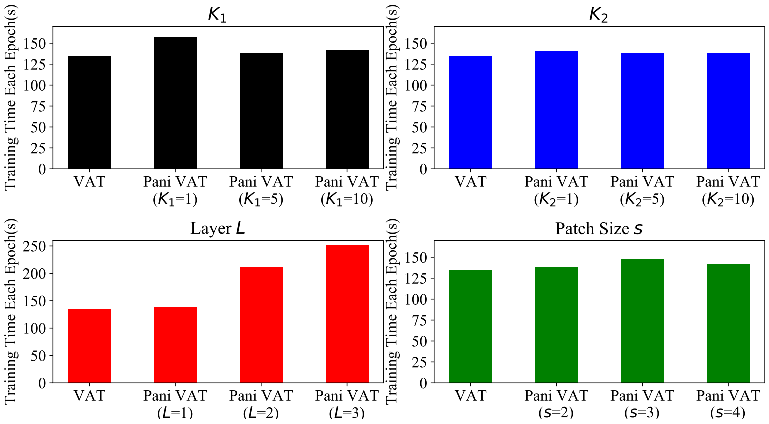

Figure 2: Average training time each epoch with respect to parameters K_1, K_2, number of layers L and patch size.

Pytorch or TensorFlow. The index of K nearest neighbor graph can be efficiently attained through *topk* operation. We conduct further sensitivity analysis on the computational cost of our method concerning other parameters, i.e., K_1 (number of peer images), K_2 (number of patch neighbors), L (number of perturbed layers) and patch size s. As shown in Figure 2, the variation of all parameters has negligible impact on the training time each epoch compared with Vanilla VAT except the number of perturbed layers. The increasing of computational cost presents an almost linear tendency with the increasing of the number of the perturbed layers as the amount of floating-point calculation is proportional to the number of perturbation elements, i.e., η, if we temporarily neglect the difference of time in the back propagation process for different layers. Combining results from Table 1 and Figure 2, we argue that better performance can be expected if we construct perturbations on more hidden layers despite the increase of computation.

4.2. Pani MixUp

Implementation Details. We strictly follow the codebase of MixUp (Zhang et al., 2017), Co-MixUp (Kim et al., 2021) MixMatch (Berthelot et al., 2019b) for a fair comparison, respectively, on CIFAR-10, CIFAR-100 and TinyImageNet datasets. After the line search of hyper-parameters for the best performance, we choose patch size as 16, parameter a in Beta distribution as 2.0 for the data augmentation setting while we choose the patch size 8, $a = 2.5$ on the setting without data augmentation across all neural architectures on CIFAR-10 and CIFAR-100. On the TinyImageNet dataset, we set the image dimension as 64×64 and batch size as 100. We also introduce a mask mechanism on η to avoid overfitting, which randomly set $\eta_{ipk} = 0$ based on a ratio. In practice, we set the mask ratio as 0.6 in the data augmentation setting while setting is as 0.4 in the scenario without data augmentation.

Result 1: Comparison with MixUp. Based on the codebase of MixUp, we compare ERM (Empirical Risk Minimization), MixUp training and our approach for different neural architectures. For a fair comparison with input MixUp, we conduct our approach only on the input layer. Table 2 presents the consistent superiority of Pani MixUp over ERM (nor-

Dataset	**Model**	**Aug**	**ERM**	**MixUp** ($a = 1$)	**Pani MixUp** (input)
CIFAR-10	PreAct ResNet-18	✓	5.43 ± 0.16	4.24 ± 0.16	**3.93 ± 0.12**
		×	12.81 ± 0.46	9.88 ± 0.25	**8.12 ± 0.09**
	PreActResNet-34	✓	5.15 ± 0.12	3.72 ± 0.20	**3.36 ± 0.15**
		×	12.67 ± 0.26	10.60 ± 0.57	**8.13 ± 0.32**
	WideResNet-28-10	✓	4.59 ± 0.06	3.21 ± 0.13	**3.02 ± 0.11**
		×	8.78 ± 0.20	8.08 ± 0.39	**5.79 ± 0.03**
CIFAR-100	PreAct ResNet-18	✓	24.96 ± 0.51	22.15 ± 0.72	**20.90 ± 0.21**
		×	39.64 ± 0.65	41.96 ± 0.27	**32.03 ± 0.34**
	PreActResNet-34	✓	24.85 ± 0.14	21.49 ± 0.68	**19.46 ± 0.29**
		×	39.41 ± 0.80	41.96 ± 0.24	**34.48 ± 0.86**
	WideResNet-28-10	✓	21.00 ± 0.09	18.58 ± 0.16	**17.39 ± 0.16**
		×	31.91 ± 0.77	35.16 ± 0.33	**27.71 ± 0.63**
TinyImageNet	PreAct ResNet-18	✓	44.90 ± 0.28	42.84 ± 0.35	**42.20 ± 0.39**
		×	54.95 ± 0.63	60.58 ± 0.83	**52.25 ± 0.75**
	PreActResNet-34	✓	40.66 ± 1.64	43.18 ± 0.84	**40.03 ± 0.61**
		×	51.03 ± 0.57	55.91 ± 1.09	**49.56 ± 0.94**
	WideResNet-28-10	✓	42.30 ± 0.51	40.64 ± 0.77	**38.97 ± 0.81**
		×	48.47 ± 0.24	51.19 ± 1.19	**46.26 ± 0.70**

Table 2: Test error in comparison with ERM, MixUp and Pani MixUp (input). All results are averaged under 5 runs. Our implementation is based on MixUp (Zhang et al., 2017).

mal training) as well as Vanilla MixUp over different deep neural network architectures. It is worth noting that the superiority of our approach in the setting without data augmentation can be more easily observed than that with data augmentation. Another interesting phenomenon is that MixUp suffers from one kind of collapse in performance as the accuracy of MixUp is even inferior to the ERM on CIFAR-100 and TinyImageNet on the setting without data augmentation. By contrast, our approach exhibits consistent advantages ERM and MixUp across various settings and network architectures.

Result 2: Comparison with Variants of MixUp. We empirically demonstrate that Pani MixUp can achieve comparable accuracy compared with variants of MixUp methods, e.g., Co-MixUp. We implement our Pani MixUp method based on the code base of (Kim et al., 2021) with the step size 0.05 and additionally compare the performance on WRN16-8 and ResNeXt29-4-24 architectures on CIFAR-100 as opposed to Table 2. In Table 3, all results are evaluated with the data augmentation. It suggests that Pani MixUp outperforms its similar baseline CutMix across all considered settings, which is also based on patches. Our Pani MixUp is also on par with the state-of-the-art Co-MixUp on CIFAR-100, although Co-MixUp performs better than other baselines, especially on Tiny-ImageNet. **However, we next show our method is significantly efficient in computation compared with Co-MixUp.**

Dataset (Model)	Vanilla	Input	Manifold	CutMix	Puzzle Mix	Co-MixUp	Pani MixUp
CIFAR-100 (PreActResNet18)	23.59	22.43	21.64	21.29	20.62	**19.87**	20.90
CIFAR-100 (WRN16-8)	21.70	20.08	20.55	20.14	19.24	19.15	**19.10**
CIFAR-100 (ResNeXt29-4-24)	21.79	21.70	22.28	21.86	21.12	**19.78**	21.13
Tiny-ImageNet (PreActResNet18)	43.40	43.48	40.76	43.11	36.52	**35.85**	40.19

Table 3: Test Error compared with more variants of MixUp, including CutMix and Co-MixUp on CIFAR-100 and Tiny-ImageNet. Results are averaged over 3 seeds after 1,200 epochs. Our implementation is based on Co-MixUp (Kim et al., 2021).

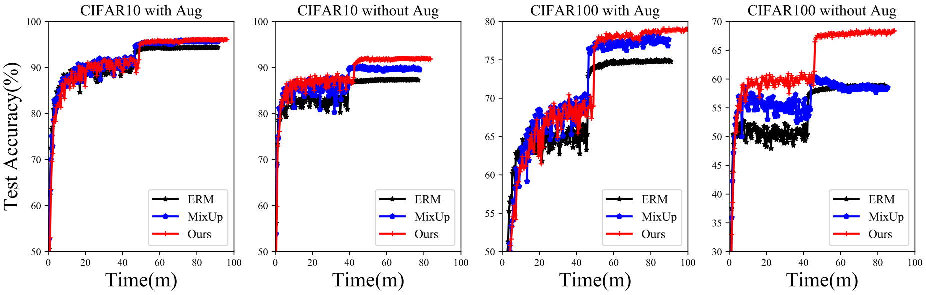

Figure 3: Learning curves with respect to the training time over ERM, MixUp and our approach, where m indicates minutes, "with Aug" and "without Aug" denote the settings with and without data augmentation, respectively.

Pani MixUp is Computationally Efficient. We first validate Pani MixUp is as computationally efficient as vanilla MixUp when compared with the normal training (ERM). Figure 3 presents all learning curves concerning the training time, where we choose ResNet-18 as the basic test model. We can easily observe that with the negligible increase in computation overhead, Pani MixUp achieves better performance than MixUp. In addition, we also compare the computation cost of Pani MixUp with variants of MixUp, especially the state-of-the-art Co-MixUp. In Figure 4, we reveal that Co-MixUp is more expensive in computation even in the parallel version, although it can achieve favorable performance as suggested in Table 3. By contrast, Pani MixUp achieves comparable results with Co-MixUp on a large number of settings in Table 3, which enjoys significant advantage of the efficiency in computation.

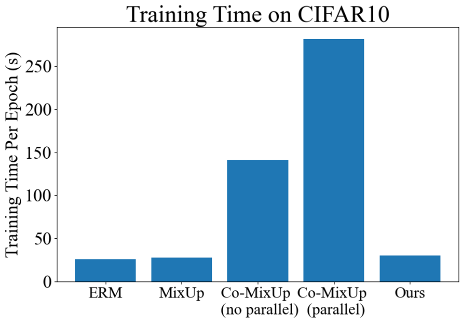

Figure 4: Comparison of Training Time per epoch on CIFAR-10.

Methods	CIFAR-10 (4,000 labels)
PiModel (Laine and Aila, 2016)	17.41 ± 0.37
PseudoLabel (Lee, 2013)	16.21 ± 0.11
MixUp (Zhang et al., 2017)	13.15 ± 0.20
VAT (Miyato et al., 2017)	11.05 ± 0.31
MeanTeacher (Tarvainen and Valpola, 2017)	10.36 ± 0.25
MixMatch (Berthelot et al., 2019b)	6.24 ± 0.06
Pani MixMatch	**6.08 ± 0.074**

Table 4: Performance of our Pani MixMatch in semi-supervised setting on CIFAR-10 with 4000 labels. The reported result is evaluated via the median of last 20 epoch while training average under 4 runs.

Further Extension to MixMatch. We further incorporate our approach into MixMatch (Berthelot et al., 2019b) that naturally extends MixUp to the semi-supervised setting. The resulting approach, which we call Pani MixMatch, elegantly replaces the MixUp part in the MixMatch with our Pani MixUp, thus imposing Pani MixUp by additionally incorporating patch neighborhood correlation knowledge. We apply the same hyper-parameters involved in Pani as in Pani MixUp and other ones are the same as MixMatch. Results in Table 4 reveals that Pani MixMatch can further improve the performance of MixMatch in the standard semi-supervised setting, thus verifying the effectiveness and flexibility of our Patch-level Neighborhood Interpolation method.

5. Discussion and Conclusion

Since the proposed Pani framework is general and flexible, more regularizations and applications could be considered in the future, such as more regularization methods and applications in natural language processing tasks. We leave this exploration as future works. In addition, an analysis of the theoretical properties of Pani is also valuable in the future.

The recent tendency of the regularization design attaches more importance to consistency and flexibility in various kinds of settings. Along this way, we propose a general regularization motivated by additional leverage of neighboring information existing in the sub-group of samples, e.g., within one batch, which can elegantly extend previous prestigious regularization approaches in a wider range of scenarios. Our proposed Patch-level Neighborhood Interpolation (Pani) method is flexible and efficient, which can be further incorporated in VAT and MixUp as well as its variants. Our work paves the way toward better understanding and leveraging the knowledge of relationships between samples to design better regularization and improve generalization over a wide range of settings.

References

David Berthelot, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Kihyuk Sohn, Han Zhang, and Colin Raffel. Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring. *arXiv preprint arXiv:1911.09785*, 2019a.

David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. Mixmatch: A holistic approach to semi-supervised learning. *Conference on Neural Information Processing Systems*, 2019b.

Antoni Buades, Bartomeu Coll, and J-M Morel. A non-local algorithm for image denoising. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 60–65. IEEE, 2005.

Zihang Dai, Zhilin Yang, Fan Yang, William W Cohen, and Ruslan R Salakhutdinov. Good semi-supervised learning that requires a bad gan. In *Advances in Neural Information Processing Systems*, pages 6510–6520, 2017.

Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*, 2016.

Murat Dundar, Balaji Krishnapuram, Jinbo Bi, and R Bharat Rao. Learning classifiers when the training data is not iid. In *IJCAI*, pages 756–761, 2007.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *International Conference on Learning Representations*, 2014.

Hongyu Guo, Yongyi Mao, and Richong Zhang. Mixup as locally linear out-of-manifold regularization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3714–3722, 2019.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *International Conference on Machine Learning*, 2015.

Konstantinos Kamnitsas, Daniel C Castro, Loic Le Folgoc, Ian Walker, Ryutaro Tanno, Daniel Rueckert, Ben Glocker, Antonio Criminisi, and Aditya Nori. Semi-supervised learning via compact latent space clustering. *arXiv preprint arXiv:1806.02679*, 2018.

Jang-Hyun Kim, Wonho Choo, and Hyun Oh Song. Puzzle mix: Exploiting saliency and local statistics for optimal mixup. In *International Conference on Machine Learning*, pages 5275–5285. PMLR, 2020.

Jang-Hyun Kim, Wonho Choo, Hosan Jeong, and Hyun Oh Song. Co-mixup: Saliency guided joint mixup with supermodular diversity. *ICLR*, 2021.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations*, 2016.

Abhishek Kumar, Prasanna Sattigeri, and Tom Fletcher. Semi-supervised learning with gans: Manifold invariance with improved inference. In *Advances in Neural Information Processing Systems*, pages 5540–5550, 2017.

Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*, 2016.

Bruno Lecouat, Chuan-Sheng Foo, Houssam Zenati, and Vijay R Chandrasekhar. Semi-supervised learning with gans: Revisiting manifold regularization. *arXiv preprint arXiv:1805.08957*, 2018.

Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on Challenges in Representation Learning, ICML*, volume 3, page 2, 2013.

Jin-Ha Lee, Muhammad Zaigham Zaheer, Marcella Astrid, and Seung-Ik Lee. Smoothmix: a simple yet effective data augmentation to train robust classifiers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 756–757, 2020.

Chongxuan Li, Kun Xu, Jun Zhu, and Bo Zhang. Triple generative adversarial nets. *arXiv preprint arXiv:1703.02291*, 2017.

Yucen Luo, Jun Zhu, Mengxi Li, Yong Ren, and Bo Zhang. Smooth neighbors on teacher graphs for semi-supervised learning. *arXiv preprint arXiv:1711.00258*, 2017.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *International Conference on Learning Representations*, 2017.

Takeru Miyato, Andrew M Dai, and Ian Goodfellow. Adversarial training methods for semi-supervised text classification. *International Conference on Learning Representations*, 2016.

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *arXiv preprint arXiv:1704.03976*, 2017.

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993, 2018.

Guo-Jun Qi, Liheng Zhang, Hao Hu, Marzieh Edraki, Jingdong Wang, and Xian-Sheng Hua. Global versus localized generative adversarial nets. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016.

Nir Sochen, Ron Kimmel, and Ravi Malladi. A general framework for low level vision. *IEEE transactions on image processing*, 7(3):310–318, 1998.

Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *Advances in neural information processing systems*, 33:596–608, 2020.

Ke Sun, Zhouchen Lin, Hantao Guo, and Zhanxing Zhu. Virtual adversarial training on graph convolutional networks in node classification. In *Chinese Conference on Pattern Recognition and Computer Vision (PRCV)*, pages 431–443. Springer, 2019.

Jan Svoboda, Jonathan Masci, Federico Monti, Michael M Bronstein, and Leonidas Guibas. Peernets: Exploiting peer wisdom against adversarial attacks. *International Conference on Learning Representations*, 2018.

Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in neural information processing systems*, pages 1195–1204, 2017.

Carlo Tomasi and Roberto Manduchi. Bilateral filtering for gray and color images. In *Sixth international conference on computer vision (IEEE Cat. No. 98CH36271)*, pages 839–846. IEEE, 1998.

Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. *International Conference on Learning Representations*, 2018.

Vladimir N Vapnik and A Ya Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. In *Measures of complexity*, pages 11–30. Springer, 2015.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *International Conference on Learning Representations*, 2017.

Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. In *International conference on machine learning*, pages 6438–6447. PMLR, 2019.

Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018.

Huaxiu Yao, Yiping Wang, Linjun Zhang, James Y Zou, and Chelsea Finn. C-mixup: Improving generalization in regression. *Advances in Neural Information Processing Systems*, 35:3361–3376, 2022.

Bing Yu, Jingfeng Wu, Jinwen Ma, and Zhanxing Zhu. Tangent-normal adversarial regularization for semi-supervised learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10676–10684, 2019.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6023–6032, 2019.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *International Conference on Learning Representations*, 2016.

Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P Xing, Laurent El Ghaoui, and Michael I Jordan. Theoretically principled trade-off between robustness and accuracy. *International Conference on Machine Learning*, 2019.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *Conference on Neural Information Processing Systems*, 2017.