

A PRELIMINARY

A.1 VIRTUAL ADVERSARIAL TRAINING AND ITS VARIANTS

VAT (Miyato et al., 2018) extends the adversarial training by utilizing “virtual” adversarial perturbations to construct the adversarial smoothness, and obtains better performance in semi-supervised learning (SSL). Particularly, VAT replaces true labels y of samples in the formulation of adversarial training by current estimate $p(y|x; \hat{\theta})$ from model:

$$\min_{\theta} \max_{r, \|r\| \leq \epsilon} D \left[p(y|x; \hat{\theta}), p(y|x + r; \theta) \right], \quad (6)$$

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 the smoothness in SSL. Then the VAT regularization $\mathcal{R}_{\text{vadV}}(x, r; \theta)$ could be derived from the inner maximization:

$$\mathcal{R}_{\text{vadV}}(x, r; \theta) = \max_{r, \|r\| \leq \epsilon} D \left[p(y|x; \hat{\theta}), p(y|x + r; \theta) \right] \quad (7)$$

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 difference. Once the desired perturbation r^* has been obtained, we can conduct forward and back propagation to optimize the full loss function:

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

where \mathcal{L}_0 is the original supervised loss and β is the hyper-parameter to control the degree of virtual adversarial smoothness. There are 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. For instance, 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 in both semi- and supervised scenarios without the requirement of any generative model.

A.2 MIXUP AND ITS VARIANTS

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:

$$\begin{aligned} \tilde{x} &= \lambda x_i + (1 - \lambda) x_j \\ \tilde{y} &= \lambda y_i + (1 - \lambda) y_j, \end{aligned} \quad (9)$$

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. Recently, MixMatch (Berthelot et al., 2019b) has been proposed by integrating several sought-after strategies, working as a natural extension of Mixup and achieving the state-of-the-art accuracy in the semi-supervised scenario. Nevertheless, we show our Pani Method can easily enhance the performance of MixUp and MixMatch in both semi- and supervised settings.

A.3 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 into the feature maps of deep models.

Let $\mathbf{Z}^1, \dots, \mathbf{Z}^N$ 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):

$$\begin{aligned}\tilde{\mathbf{z}}_p^i &= \sum_{k=1}^K \alpha_{ij_kpq_k} \mathbf{z}_{q_k}^{j_k}, \\ \alpha_{ij_kpq_k} &= \frac{\text{LeakyReLU} \left(\exp \left(f_a \left(\mathbf{z}_p^i, \mathbf{z}_{q_k}^{j_k} \right) \right) \right)}{\sum_{k'=1}^K \text{LeakyReLU} \left(\exp \left(f_a \left(\mathbf{z}_p^i, \mathbf{z}_{q_{k'}}^{j_{k'}} \right) \right) \right)},\end{aligned}\tag{10}$$

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. $f_a(\cdot)$ 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.

B DISCUSSION ABOUT REGULARIZATION EFFECT

Manifold Regularization. There are a flurry of papers introducing regularization from the classical manifold learning based on the assumption that the data can be modeled as a low-dimensional manifold in the data space. More importantly, Hinton et al. (Hinton et al., 2012) and Ioffe et al. (Ioffe & Szegedy, 2015) demonstrated 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 & 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 and serves as a generic component for capturing long-range dependencies with deep neural networks. Similar with these approaches, our Patch-level Neighborhood Interpolation still has the capability to capture the correlation knowledge of patch features within a batch, therefore yielding an improvement of performance for the derived methods. Moreover, our Patch-level Neighborhood Interpolation can also serve 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.

C IMPLEMENTATION DETAILS

Pani VAT. 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.

Pani MixUp. 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 opt patch size 8, $a = 2.5$ on the setting without data augmentation across all neural architectures and datasets. On the TinyImageNet dataset, we set the image dimension as 64×64 and batch size as 100 expect WRN28-10 as 50 due to the memory limitation.

Mask Mechanism. To extend the flexibility of Pani MixUp, we additionally introduce the mask mechanism on the interpolation coefficient η to random drop η_{ipk} with certain ratio. The mask mechanism can be viewed as dropout or enforcing sparsity, which can help to reduce redundant information while conducting Pani MixUp. After tuning this hyper-parameter, we set the mask ratio

as 0.6 in the data augmentation setting while fixing the ratio as 0.4 in the scenario without data augmentation.