Regularizing activations in neural networks via distribution matching with the Wasserstein metric

Anonymous authors
Paper under double-blind review

ABSTRACT

Regularization and normalization have become indispensable components in training deep neural networks, resulting in faster training and improved generalization performance. We propose the projected error function regularization loss (PER) that encourages activations to follow the standard normal distribution. PER randomly projects activations onto one-dimensional space and computes the regularization loss in the projected space. PER is similar to the Pseudo-Huber loss in the projected space, thus taking advantage of both $L^1$ and $L^2$ regularization losses. Besides, PER can capture the interaction between hidden units by projection vector drawn from a unit sphere. By doing so, PER minimizes the upper bound of the Wasserstein distance of order one between an empirical distribution of activations and the standard normal distribution. To the best of the authors’ knowledge, this is the first work to regularize activations via distribution matching in the probability distribution space. We evaluate the proposed method on the image classification task and the word-level language modeling task.

1 INTRODUCTION

Training of deep neural networks is very challenging due to the vanishing and exploding gradient problem (Hochreiter, 1998; Glorot & Bengio, 2010), the presence of many flat regions and saddle points (Shalev-Shwartz et al., 2017), and the shattered gradient problem (Balduzzi et al., 2017). To remedy these issues, various methods for controlling hidden activations have been proposed such as normalization (Ioffe & Szegedy, 2015; Huang et al., 2018), regularization (Littwin & Wolf, 2018), initialization (Mishkin & Matas, 2015; Zhang et al., 2019), and architecture design (He et al., 2016).

Among various techniques of controlling activations, one well-known and successful path is controlling their first and second moments. Back in the 1990s, it has been known that the neural network training can be benefited from normalizing input statistics so that samples have zero mean and identity covariance matrix (LeCun et al., 1998; Schraudolph, 1998). This idea motivated batch normalization (BN) that considers hidden activations as the input to the next layer and normalizes scale and shift of the activations (Ioffe & Szegedy, 2015).

Recent works show the effectiveness of different sample statistics of activations for normalization and regularization. Deecke et al. (2019) and Kalayeh & Shah (2019) normalize activations to several modes with different scales and translations. Variance constancy loss (VCL) implicitly normalizes the fourth moment by minimizing the variance of sample variances, which enables adaptive mode separation or collapse based on their prior probabilities (Littwin & Wolf, 2018). BN is also extended to whiten activations (Huang et al., 2018; 2019), and to normalize general order of central moment in the sense of $L^p$ norm including $L^0$ and $L^\infty$ (Liao et al., 2016; Hoffer et al., 2018).

In this paper, we propose a projected error function regularization (PER) that regularizes activations in the Wasserstein probability distribution space. Specifically, PER pushes the distribution of activations to be close to the standard normal distribution. PER shares a similar strategy with previous approaches that dictates the ideal distribution of activations. Previous approaches, however, deal with single or few sample statistics of activations. On the contrary, PER regularizes the activations by matching the probability distributions, which considers different statistics simultaneously, e.g,
Figure 1: Limitation of statistics in terms of representing the probability distribution. In all subplots, $x$ has zero mean and unit variance and $y \sim \mathcal{N}(0, 1)$. In (a), $(x, y) \sim \mathcal{N}(0, I)$. In (b), $x \sim \mathcal{N}(0, 1)$ but correlated with $y$. In (c), $x$ follows a skewed distribution. In (d), $x$ follows a bi-modal distribution.

Standardization cannot differentiate (a)-(d) and whitening cannot differentiate (a), (c), and (d).

all orders of moments and correlation between hidden units. The extensive experiments on multiple challenging tasks show the effectiveness of PER.

2 RELATED WORKS

Many modern deep learning architectures employ BN as an essential building block for better performance and stable training even though its theoretical aspects of regularization and optimization are still actively investigated (Santurkar et al., 2018; Kohler et al., 2018; Bjorck et al., 2018; Yang et al., 2019). Several studies have applied the idea of BN that normalizes activations via the sample mean and the sample variance to a wide range of domains such as recurrent neural network (Lei Ba et al., 2016) and small batch size training (Wu & He, 2018).

Huang et al. (2018; 2019) propose normalization techniques whitening the activation of each layer. This additional constraint on the statistical relationship between activations improves the generalization performance of residual networks compared to BN. Although the correlation between activations are not explicitly considered, dropout prevents activations from being activated at the same time, called co-adaptation, by randomly dropping the activations (Srivastava et al., 2014), the weights (Wan et al., 2013), and the spatially connected activations (Ghiasi et al., 2018).

Considering BN as the normalization in the $L^2$ space, several works extend BN to other spaces, i.e., other norms. Streaming normalization (Liao et al., 2016) explores the normalization of a different order of central moment with $L^p$ norm for general $p$. Similarly, Hoffer et al. (2018) explores $L^1$ and $L^\infty$ normalization, which enable low precision computation. Littwin & Wolf (2018) proposes a regularization loss that reduces the variance of sample variances of activation that is closely related to the fourth moment.

The idea of controlling activations via statistical characteristics of activations also has motivated initialization methods. An example includes balancing variances of each layer (Glorot & Bengio, 2010; He et al., 2015), bounding scale of activation and gradient (Mishkin & Matas, 2015; Balduzzi et al., 2017; Gehring et al., 2017; Zhang et al., 2019), and norm preserving (Saxe et al., 2013). Although the desired initial state may not be maintained during training, experimental results show that they can stabilize the learning process as well.

Recently, the Wasserstein metric has gained much popularity in a wide range of applications in deep learning with some nice properties such as being a metric in a probability distribution space without requiring common supports of two distributions. For instance, it is successfully applied to a multi-labeled classification (Frogner et al., 2015), gradient flow of policy update in reinforcement learning (Zhang et al., 2018), training of generative models (Arjovsky et al., 2017; Gulrajani et al., 2017; Kolouri et al., 2019), and capturing long term semantic structure in sequence-to-sequence language model (Chen et al., 2019).

While the statistics such as mean and (co)variance are useful summaries of a probability distribution, they cannot fully represent the underlying structure of the distribution (Fig. 1). Therefore, regularizing or normalizing activation to follow the target distribution via statistics can be ineffective in
some cases. For instance, normalizing activations via single mean and variance such as BN and decorrelated BN (Huang et al., 2018) can be inadequate in learning multimodal distribution (Bilen & Vedaldi, 2017; Deecke et al., 2019). This limitation motivates us to investigate a more general way of regularizing the distribution of activations. Instead of controlling activations via statistics, we define the target distribution and then minimize the Wasserstein distance between the activation distribution and the target distribution.

3 PROJECTED ERROR FUNCTION REGULARIZATION

We consider a neural network with $L$ layers each of which has $d_l$ hidden units in layer $l$. Let $\mathcal{T} = \{(x_i, y_i)\}_{i=1}^n$ be $n$ training samples which are assumed to be i.i.d. samples drawn from a probability distribution $P_{x,y}$. In this paper, we consider the optimization by stochastic gradient descent with mini-batch of $b$ samples randomly drawn from $\mathcal{T}$ at each training iteration. For $i$-th element of the samples, the neural network recursively computes:

$$h_i^l = \phi \left(W^l h_i^{l-1} + b^l\right)$$  \hspace{1cm} (1)

where $h_i^0 = x_i \in \mathbb{R}^{d_0}$, $h_i^l \in \mathbb{R}^{d_l}$ is an activation in layer $l$, and $\phi$ is an activation function. In the case of recurrent neural networks (RNNs), the recursive relationship takes the form of:

$$h_i^l = \phi \left(W^l_{rec} h_i^{l-1} + W^l_{in} h_i^{l-1} + b^l\right)$$  \hspace{1cm} (2)

where $h_i^l$ is an activation in layer $l$ at time $t$ and $h_i^0$ is an initial state. Without loss of generality, we focus on activations in layer $l$ of feed-forward networks and the mini-batch of samples $\{(x_i, y_i)\}_{i=1}^b$. Throughout this paper, we let $f^l$ be a function made by compositions of recurrent relation in equation (1) up to layer $l$, i.e., $h_i^l = f^l(x_i)$, and $f^j$ be a $j$-th output of $f^l$.

This paper proposes a new regularization loss, called PER, that encourages activations to follow the target distribution via the Wasserstein metric. Let $\mu \in \mathcal{P}(\mathbb{R}^{d_l})$ be the Gaussian measure defined as $\mu(h) = \frac{1}{(2\pi)^{d_l/2}} \int_h \exp \left(-\frac{1}{2} \| x \|^2 \right) dx$ and $\nu_{h_i^l} = \frac{1}{b} \sum_{i=1}^b \delta_{h_i^l} \in \mathcal{P}(\mathbb{R}^{d_l})$ be the empirical measure of hidden activations where $\delta_{h_i^l}$ is the Dirac unit mass on $h_i^l$. Then, the Wasserstein metric of order $p$ between $\mu$ and $\nu_{h_i^l}$ is defined by:

$$W_p(\mu, \nu_{h_i^l}) = \left( \inf_{\pi \in \Pi(\mu, \nu_{h_i^l})} \int_{\mathbb{R}^{d_l} \times \mathbb{R}^{d_l}} d^p(x, \pi(x, y)) \pi(dx, dy) \right)^{1/p}$$  \hspace{1cm} (3)

where $\Pi(\mu, \nu_{h_i^l})$ is the set of all joint probability measures on $\mathbb{R}^{d_l} \times \mathbb{R}^{d_l}$ having the first and the second marginals $\mu$ and $\nu_{h_i^l}$, respectively.

Because direct computation of equation (3) is intractable, we consider the sliced Wasserstein distance (Rabin et al., 2011) approximating the Wasserstein distance by projecting the high dimensional distributions onto $\mathbb{R}$ (Fig. 2). It is proved by that the sliced Wasserstein and the Wasserstein are equivalent metrics (Santambrogio, 2015; Bonnotte, 2013). The sliced Wasserstein of order one between $\mu$ and $\nu_{h_i^l}$ can be formulated as:

$$SW_1(\mu, \nu_{h_i^l}) = \int_{\mathbb{S}^{d_l-1}} W_1(\mu_{\theta}, \nu_{h_i^l_{\theta}}) d\lambda(\theta) = \int_{\mathbb{S}^{d_l-1}} \int_{-\infty}^{\infty} |F_{\mu_{\theta}}(x) - \frac{1}{b} \sum_{i=1}^b 1(h_i^l, \theta) \leq x| dx d\lambda(\theta)$$  \hspace{1cm} (4)

where $\mathbb{S}^{d_l-1}$ is a unit sphere in $\mathbb{R}^{d_l}$, $\mu_{\theta}$ and $\nu_{h_i^l_{\theta}}$ represent the measures projected to the angle $\theta$, $\lambda$ is a uniform measure on $\mathbb{S}^{d_l-1}$, and $F_{\mu_{\theta}}(x)$ is a cumulative distribution function of $\mu_{\theta}$. Herein, equation (4) can be evaluated through sorting $\{(h_i^l, \theta)\}_i$ for each angle $\theta$.

While we can directly use the sliced Wasserstein in equation (4) as a regularization loss, it has a computational dependency on the batch dimension due to the sorting. The computational dependency between samples may not be desirable in distributed and large-batch training that is becoming more and more prevalent in recent years. For this reason, we remove the dependency by applying the
Under review as a conference paper at ICLR 2020

Figure 2: Illustration of minimization of the sliced Wasserstein distance between the current distribution and the target distribution. Note that it only concerns a distance in the projected dimension.

Algorithm 1 Backward pass under PER

**Input** The number of Monte Carlo evaluations $s$, an activation for $i$-th sample $h_i$, the gradient of the loss $\nabla_h L$, a regularization coefficient $\lambda$

1: $g \leftarrow 0$
2: for $k \leftarrow 1$ to $s$ do
3: \hspace{1em} Sample $v \sim \mathcal{N}(0, I)$
4: \hspace{1em} $\theta \leftarrow v / \|v\|_2$
5: \hspace{1em} Project $h_i' \leftarrow (h_i, \theta)$
6: \hspace{1em} $g_k \leftarrow \text{erf} \left( \frac{h_i'}{\sqrt{2}} \right)$
7: \hspace{1em} $g \leftarrow g + g_k \theta / s$
8: end for
9: return $\nabla_h L + \lambda g$

Minkowski inequality to equation 4 and obtain the regularization loss $L_{\text{per}}(\nu_{h_i})$:

$$SW_1(\mu, \nu_{h_i}) = \frac{1}{b} \int_{S^{d-1}} \sum_i \left( h_i, \theta \right) \text{erf} \left( \frac{h_i, \theta}{\sqrt{2}} \right) + \sqrt{\frac{2}{\pi}} \exp \left( -\frac{h_i, \theta}{2} \right) d\lambda(\theta)$$

$$\leq \frac{1}{b} \sum_i \int_{S^{d-1}} \left( h_i, \theta \right) \text{erf} \left( \frac{h_i, \theta}{\sqrt{2}} \right) + \sqrt{\frac{2}{\pi}} \exp \left( -\frac{h_i, \theta}{2} \right) d\lambda(\theta) = L_{\text{per}}(\nu_{h_i})$$

whose gradient with respect to $h_i$ is:

$$\nabla_{h_i} L_{\text{per}}(\nu_{h_i}) = \frac{1}{b} \mathbb{E}_{\theta \sim U(S^{d-1})} \left[ \text{erf} \left( \frac{\theta}{\sqrt{2}} \right) \theta \right]$$

where $U(S^{d-1})$ is the uniform distribution on $S^{d-1}$. In this paper, expectation over $U(S^{d-1})$ is approximated by the Monte Carlo method with $s$ number of samples. Therefore, PER results in simple modification of the backward pass as in Alg. 1.

Encouraging activations to follow the standard normal distribution can be motivated by the natural gradient [Amari 1998]. The natural gradient is the steepest descent direction in a Riemannian manifold, and it is also the direction that maximizes the probability of not increasing generalization error [Roux et al. 2008]. The natural gradient is obtained by multiplying the inverse Fisher information matrix to the gradient. In Raiko et al. [2012] and Desjardins et al. [2015], under the independence assumption between forward and backward passes and activations between different layers, the Fisher information matrix is a block diagonal matrix each of which block is given by:

$$F_l = \mathbb{E}_{(x,y) \sim (x,y)} \left[ \frac{\partial L}{\partial \text{vec}(W_l)} \frac{\partial L}{\partial \text{vec}(W_l)}^T \right] = \mathbb{E}_x \left[ h^{l-1} h^{l-1}^T \right] \mathbb{E}_{(x,y)} \left[ \frac{\partial L}{\partial a^l} \frac{\partial L}{\partial a^l}^T \right]$$

where vec$(W_l)$ is vectorized $W_l$, $h^{l-1} = f^{l-1}(x)$, and $a^l = W_l f^{l-1}(x) + b^l$ for $x \sim x$.

Since computing the inverse Fisher information matrix is too expensive to perform every iterations, previous studies put efforts into developing reparameterization techniques, activation functions, and

4
regularization losses to make $F^l$ close to $I$, thereby making the gradient close to the natural gradient. For instance, making zero mean and unit variance activations (LeCun et al., 1998; Schraudolph, 1998; Wiesler et al., 2014; Glorot & Bengio, 2010; Raiko et al., 2012) and decorrelated activations (Huang et al., 2018; Cogswell et al., 2015; Xiong et al., 2016) make $E\left[h^{-1}h^{-1T}\right] \approx I$, and these techniques result in faster training and improved generalization performance. In this perspective, it is expected that PER will enjoy the same advantages by matching $\nu_{hl}$ to $N(0, I)$.

3.1 Comparison to Controlling Activations in $L^p$ Space

In this subsection, we theoretically compare PER with existing methods that control activations in $L^p$ space. $L^p(\mathbb{R}^d_0)$ is the space of measurable functions whose $p$-th power of absolute value is Lebesgue integrable, and norm of $f \in L^p(\mathbb{R}^d_0)$ is given by:

$$\|f\|_p = \left(\int_{\mathbb{R}^d_0} |f(x)|^p dP_x(x)\right)^{1/p} < \infty \quad (8)$$

where $P_x$ is the unknown probability distribution generating training samples $\{x_i\}_{i=1}^n$. Since we have no access to $P_x$, it is approximated by the empirical measure of mini-batch samples.

The $L^p$ norm is widely used in the literature for regularization and normalization of neural networks. For instance, activation norm regularization (Merity et al., 2017) penalizes $L^p$ norm of activations. As another example, BN and its $p$-th order generalization use $L^p$ norm such that the norm of the centralized activation, or pre-activation, is bounded:

$$\psi(h_{ij}^l) = \gamma_j^l \xi(h_{ij}^l) + \beta_j^l, \quad \xi(h_{ij}^l) = \frac{h_{ij}^l - \bar{\mu}_j}{\left(\sum_k \frac{1}{b} h_{kj}^l - \bar{\mu}_j\right)^{1/p}} \quad (9)$$

where $h_{ij}^l$ is $j$-th unit of $h_i^l$, $\bar{\mu}_j = \frac{1}{b} \sum_k h_{kj}^l$ is the sample mean, $\beta_j^l$ is a learnable shift parameter, and $\gamma_j^l$ is a learnable scale parameters. Herein, we have $\|\xi \circ f_j^l\|_p = 1$ for any unit $j$ and any empirical measure, thus $\|\psi\|_p \leq \|\gamma_j^l \xi \circ f_j^l\|_p + \|\beta_j^l\|_p = |\gamma_j^l| + |\beta_j^l|$.

PER differs from $L^p$ norm-based approaches in two aspects. First, PER can be considered as $L^p$ norm with adaptive order in the projected space because it is very similar to the Pseudo-Huber loss in one-dimensional space (Fig. 3). Herein, the Pseudo-Huber loss is a smooth approximation of the Huber loss (Huber, 1964). Therefore, PER smoothly changes its behavior between $L^1$ and $L^2$ norms, making the regularization loss sensitive to small values and insensitive to outliers with large values. However, the previous approaches use predetermined order $p$, which makes the norm to change insensitively in the near-zero region when $p \leq 1$ or to explode in large value region when $p > 1$.

Second, PER captures the interaction between hidden units by projection vectors, unlike $L^p$ norm. To see this, let $\|f^l\|_p^p = \frac{1}{b} \sum_{i,j} |h_{ij}^l|^p = \frac{1}{b} \sum_{i,j} \langle h_{ij}^l, e_j \rangle^p$ where $\{e_j\}_{j=1}^d$ is the natural basis of

\[ \text{Figure 3: Illustration of PER and its gradient in } \mathbb{R}. \text{ Herein, PER is shifted by } c \text{ so that } L_{per}(0) - c = 0. \text{ The Huber loss is defined as } h(x) = |x| - 0.5 \text{ in } |x| > 1 \text{ and } h(x) = x^2/2 \text{ in } |x| \leq 1 \text{ and the Pseudo-Huber loss is defined as } g(x) = \sqrt{1 + x^2} - 1. \]
Table 1: Top-1 error rates of ResNets on CIFAR-10. Lower is better. All numbers are rounded to two decimal places. Boldface indicates the minimum error. * and ** are results from Zhang et al. (2019) and He et al. (2016), respectively.

<table>
<thead>
<tr>
<th>Model</th>
<th>Method</th>
<th>Test error</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-56</td>
<td>Vanilla</td>
<td>7.21</td>
</tr>
<tr>
<td></td>
<td>BN</td>
<td>6.95</td>
</tr>
<tr>
<td></td>
<td>PER</td>
<td><strong>6.72</strong></td>
</tr>
<tr>
<td>ResNet-110</td>
<td>Vanilla</td>
<td>6.90 (7.24*)</td>
</tr>
<tr>
<td></td>
<td>BN</td>
<td>6.62 (6.61**)</td>
</tr>
<tr>
<td></td>
<td>PER</td>
<td><strong>6.19</strong></td>
</tr>
</tbody>
</table>

Table 2: Top-1 error rates of 11-layer CNNs on tiny ImageNet. Lower is better. All numbers are rounded to two decimal places. Boldface indicates the minimum error. Numbers in parentheses represent results in Littwin & Wolf (2018).

<table>
<thead>
<tr>
<th>Method</th>
<th>Test error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vanilla</td>
<td>37.45 (39.22)</td>
</tr>
<tr>
<td>BN</td>
<td>39.22 (40.02)</td>
</tr>
<tr>
<td>VCL</td>
<td>(37.30)</td>
</tr>
<tr>
<td>PER</td>
<td><strong>36.74</strong></td>
</tr>
</tbody>
</table>

That is, the norm computes the regularization loss, or the normalizer, of activations with the natural basis as a projection vector. However, PER uses general projection vectors \( \theta \sim U(S^{d-1}) \), capturing the interaction between hidden units when computing the regularization loss. These two differences make PER more delicate criterion for regularizing activations in deep neural networks than \( L_p \) norm, as we will show in the next section.

4 EXPERIMENTS

This section illustrates the effectiveness of PER through experiments on different benchmark tasks with various datasets and architectures. We compare PER with BN normalizing the first and second moments and VCL regularizing the fourth moments. PER is also compared with \( L_1 \) and \( L_2 \) activation norm regularizations that behave similarly in some regions of the projected space. We then analyze the computational complexity PER and the impact of PER on the distribution of activations. Throughout all experiments, we use 256 number of slices and the same regularization coefficient for the regularization losses computed in each layer.

4.1 IMAGE CLASSIFICATION IN CIFAR-10, CIFAR-100, AND TINY IMAGENET

We evaluate PER in image classification task in CIFAR (Krizhevsky et al., 2009) and a subset of ImageNet (Russakovsky et al., 2015), called tiny ImageNet. We first evaluate PER with ResNet (He et al., 2016) in CIFAR-10 and compare it with BN and a vanilla network initialized by fixup initialization (Zhang et al., 2019). We match the experimental details in training under BN with He et al. (2016) and under PER and vanilla with Zhang et al. (2019), and we obtain similar performances presented in the papers. Herein, we search the regularization coefficient over \{ 3e-4, 1e-4, 3e-5, 1e-5 \}. Table 1 presents results of CIFAR-10 experiments with ResNet-56 and ResNet-110. PER outperforms BN as well as vanilla networks in both architectures. Especially, PER improves the test errors by 0.49 % and 0.71 % in ResNet-56 and ResNet-110 without BN, respectively.

We also performed experiments on an 11-layer convolutional neural network (11-layer CNN) examined in VCL (Littwin & Wolf, 2018). This architecture is originally proposed in Clevert et al. (2015). Following Littwin & Wolf (2018), we perform experiments on 11-layer CNNs with ELU, ReLU, and Leaky ReLU activations, and match experimental details in Littwin & Wolf (2018) except that we used 10x less learning rate for bias parameters and additional scalar bias after ReLU and Leaky ReLU based on Zhang et al. (2019). By doing so, we obtain similar results presented in Littwin & Wolf (2018). Then, a search space of the regularization coefficient is \{ 3e-4, 1e-4, 3e-5, 1e-5 \}. For ReLU and Leaky ReLU in CIFAR-100, however, we additionally search \{ 3e-6, 1e-6, 3e-7, 1e-7 \} because of divergence of training with PER in these setting. As shown in Table 3, PER shows the best performances on four out of six experiments. In other cases, PER gives compatible performances to BN or VCL, giving 0.16 % less than the best performances.

Following Littwin & Wolf (2018), PER is also evaluated on tiny ImageNet. In this experiment, the number of convolutional filters in each layer is doubled. Due to the limited time and resources, we
Table 3: Top-1 error rates of 11-layer CNNs on CIFAR-10 and CIFAR-100. Lower is better. All numbers are rounded to two decimal places. Boldface indicates the minimum error. Numbers in parentheses represent results in Littwin & Wolf (2018).

<table>
<thead>
<tr>
<th>Activation Method</th>
<th>CIFAR-10</th>
<th>CIFAR-100</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReLU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vanilla</td>
<td>8.43 (8.36)</td>
<td>29.45 (32.80)</td>
</tr>
<tr>
<td>BN</td>
<td>7.53 (7.78)</td>
<td><strong>29.13</strong> (29.10)</td>
</tr>
<tr>
<td>VCL</td>
<td>7.80 (7.80)</td>
<td>30.30 (30.30)</td>
</tr>
<tr>
<td>PER</td>
<td><strong>7.21</strong></td>
<td>29.29</td>
</tr>
<tr>
<td>LeakyReLU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vanilla</td>
<td>6.73 (6.70)</td>
<td>26.50 (26.80)</td>
</tr>
<tr>
<td>BN</td>
<td>6.38 (7.08)</td>
<td>26.83 (27.20)</td>
</tr>
<tr>
<td>VCL</td>
<td>6.45 (6.45)</td>
<td>26.30 (26.30)</td>
</tr>
<tr>
<td>PER</td>
<td><strong>6.29</strong></td>
<td><strong>25.50</strong></td>
</tr>
<tr>
<td>ELU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vanilla</td>
<td>6.74 (6.98)</td>
<td>27.53 (28.70)</td>
</tr>
<tr>
<td>BN</td>
<td>6.69 (6.63)</td>
<td>26.60 (26.90)</td>
</tr>
<tr>
<td>VCL</td>
<td><strong>6.26</strong> (6.15)</td>
<td>25.86 (25.60)</td>
</tr>
<tr>
<td>PER</td>
<td>6.42</td>
<td><strong>25.73</strong></td>
</tr>
</tbody>
</table>

conduct experiments only with ELU that gives good performances for PER, BN, and VCL in CIFAR. As shown in Table 2, PER is also effective in the larger model in the larger image classification dataset.

4.2 LANGUAGE MODELING IN PTB AND WikiText2

We evaluate PER in word-level language modeling task in PTB (Mikolov et al., 2010) and WikiText2 (Merity et al., 2016). We apply PER to LSTM with two layers having 650 hidden units with and without reuse embedding (RE) proposed in Inan et al. (2016) and Press & Wolf (2016), and variational dropout (VD) proposed in Gal & Ghahramani (2016). We used the same configurations with Merity et al. (2017) and failed to reproduce the results in Merity et al. (2017). Especially, when we rescale gradient when its norm exceeds 10, we observed divergence or bad performance (almost 2x perplexity compared to the published result). Therefore, we rescale gradient with norm over 0.25 instead of 10 based on the default hyperparameter of the PyTorch word-level language model that is also mentioned in Merity et al. (2017). We also train the networks for 60 epochs instead of 80 epochs since validation perplexity is not improved after 60 epochs in most cases. In this task, PER is compared with recurrent BN (RBN) because BN is not directly applicable to LSTM (Cooijmans et al., 2016). We also compare PER with $L^1$ and $L^2$ activation norm regularizations. Herein, the search space of regularization coefficients of PER, $L^1$ regularization, and $L^2$ regularization is {3e-4, 1e-4, 3e-5}. For $L^1$ and $L^2$ penalties in PTB, we search additional coefficients over {1e-5, 3e-6, 1e-6, 3e-6, 1e-7, 3e-7, 1e-7} because the searched coefficients seem to constrain the capacity.

We list in Table 4 the perplexities of methods on PTB and WikiText2. While all regularization techniques show regularization effects by giving improved test perplexity, PER gives the best test perplexity except LSTM and RE-VD-LSTM in the PTB dataset wherein PER is the second-best method. We also note that naively applying RBN often reduces performance. For instance, RBN increases test perplexity of VD-LSTM by about 5 in PTB and WikiText2.

4.3 ANALYSIS

In this subsection, we analyze the computational complexity of PER and its impact on closeness to the standard normal distribution in the 11-layer CNN.

---

1 Available in [https://github.com/pytorch/examples/tree/master/word_language_model](https://github.com/pytorch/examples/tree/master/word_language_model)
Table 4: Validation and test perplexities on PTB and WikiText2. Lower is better. All numbers are rounded to one decimal place. Boldface indicates minimum perplexity.

<table>
<thead>
<tr>
<th>Model</th>
<th>Method</th>
<th>PTB Valid</th>
<th>PTB Test</th>
<th>WikiText2 Valid</th>
<th>WikiText2 Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM</td>
<td>Vanilla</td>
<td>123.2</td>
<td>122.0</td>
<td>138.9</td>
<td>132.7</td>
</tr>
<tr>
<td></td>
<td>$L^1$ penalty</td>
<td>119.6</td>
<td><strong>114.1</strong></td>
<td>137.7</td>
<td>130.0</td>
</tr>
<tr>
<td></td>
<td>$L^2$ penalty</td>
<td>120.5</td>
<td>115.2</td>
<td>136.0</td>
<td>131.1</td>
</tr>
<tr>
<td></td>
<td>RBN</td>
<td><strong>118.2</strong></td>
<td>115.1</td>
<td>156.2</td>
<td>148.3</td>
</tr>
<tr>
<td></td>
<td>PER</td>
<td>118.5</td>
<td>114.5</td>
<td><strong>134.2</strong></td>
<td><strong>129.6</strong></td>
</tr>
<tr>
<td>RE-LSTM</td>
<td>Vanilla</td>
<td>114.1</td>
<td>112.2</td>
<td>129.2</td>
<td>123.2</td>
</tr>
<tr>
<td></td>
<td>$L^1$ penalty</td>
<td>112.2</td>
<td>108.5</td>
<td>128.6</td>
<td>122.7</td>
</tr>
<tr>
<td></td>
<td>$L^2$ penalty</td>
<td>116.6</td>
<td><strong>108.2</strong></td>
<td>126.5</td>
<td>123.3</td>
</tr>
<tr>
<td></td>
<td>RBN</td>
<td>113.6</td>
<td>110.4</td>
<td>138.1</td>
<td>131.6</td>
</tr>
<tr>
<td></td>
<td>PER</td>
<td><strong>110.0</strong></td>
<td>108.5</td>
<td><strong>123.2</strong></td>
<td><strong>117.4</strong></td>
</tr>
<tr>
<td>VD-LSTM</td>
<td>Vanilla</td>
<td>84.9</td>
<td>81.1</td>
<td>99.6</td>
<td>94.5</td>
</tr>
<tr>
<td></td>
<td>$L^1$ penalty</td>
<td>84.9</td>
<td>81.5</td>
<td>98.2</td>
<td>92.9</td>
</tr>
<tr>
<td></td>
<td>$L^2$ penalty</td>
<td>84.5</td>
<td>81.2</td>
<td>98.8</td>
<td>94.2</td>
</tr>
<tr>
<td></td>
<td>RBN</td>
<td>89.7</td>
<td>86.4</td>
<td>104.3</td>
<td>99.4</td>
</tr>
<tr>
<td></td>
<td>PER</td>
<td><strong>84.1</strong></td>
<td><strong>80.7</strong></td>
<td><strong>98.1</strong></td>
<td><strong>92.6</strong></td>
</tr>
<tr>
<td>RE-VD-LSTM</td>
<td>Vanilla</td>
<td>78.9</td>
<td>75.7</td>
<td>91.4</td>
<td>86.4</td>
</tr>
<tr>
<td></td>
<td>$L^1$ penalty</td>
<td>78.3</td>
<td>75.1</td>
<td>90.5</td>
<td>86.1</td>
</tr>
<tr>
<td></td>
<td>$L^2$ penalty</td>
<td>79.2</td>
<td>75.8</td>
<td><strong>90.3</strong></td>
<td>86.1</td>
</tr>
<tr>
<td></td>
<td>RBN</td>
<td>83.7</td>
<td>80.5</td>
<td>95.5</td>
<td>90.5</td>
</tr>
<tr>
<td></td>
<td>PER</td>
<td><strong>78.1</strong></td>
<td><strong>74.9</strong></td>
<td>90.6</td>
<td><strong>85.9</strong></td>
</tr>
</tbody>
</table>

4.3.1 Computational complexity

PER has no additional parameters. However, BN and VCL require additional parameters for each channel and each location and channel in every layer, respectively; that is, 2.5K and 350K number of parameters are introduced in BN and VCL in the 11-layer CNN, respectively. In terms of time complexity, PER has the complexity of $O(bdl)$ for projection operation in each layer $l$. On the other hand, BN and VCL have $O(bdl)$ complexities. In our benchmarking, each training iteration takes 0.071 seconds for a vanilla network, 0.083 seconds for BN, 0.087 for VCL, and 0.093 seconds for PER on a single NVIDIA TITAN X. Even though PER requires slightly more training time than BN and VCL, this disadvantage can be mitigated by computation of PER is only required in training and PER does not have additional parameters.

4.3.2 Closeness to the standard normal distribution

To examine the effect of PER on the closeness to $\mathcal{N}(0, I)$, we analyze the distribution of activations in 11-layer CNN in different perspectives. We first analyze the distribution of a single activation $h^l_j$ for some unit $j$ and layer $l$ (Fig. 4). We observe that changes in probability distributions between two consecutive epochs are small under BN because BN bound the $L^2$ norm of activations into learned parameters. On the contrary, activation distributions under vanilla and PER are jiggled between two consecutive epochs. However, PER prevents the variance explosion and pushes the mean to zero. As shown in Fig. 4, while variances of $\nu_{h_j^l}$ under both PER and Vanilla are very high at the beginning of training, the variance keeps moving towards one under PER during training. Similarly, PER recovers biased means of $\nu_{h_j^l}$ at the early stage of learning.

To precisely evaluate closeness to the standard normal distribution, we also analyze $SW_1(\mathcal{N}(0, I), \nu_{h_j^l})$ at each epoch (Fig. 5). Herein, the sliced Wasserstein distance is computed by approximating the Gaussian measure using the empirical measure of samples drawn from $\mathcal{N}(0, I)$ as in Rabin et al. (2011). As similar to the previous result, while BN $\beta_j = 0$ and $\gamma_j = 1$ at initial state gives small $SW_1(\mathcal{N}(0, I), \nu_{h_j^l})$ in early stage of training, PER also can effectively control...
Figure 4: Evolution of distributions of $\nu_{h^3_i}$, $\nu_{h^6_j}$, and $\nu_{h^9_k}$ for fixed randomly drawn $i, j, k$ on training set. (a)-(c) represent values (0.25, 0.5, 0.75) quantiles under PER, vanilla, and BN. (d) and (e) represent the sample mean and the sample variance of activations. Variance is clipped at 5 for better visualization.

Figure 5: Closeness to $\mathcal{N}(0, I)$ in the Wasserstein probability distribution space.

the distribution without such normalization. This confirms that PER prevents the distribution of activation to be drifted away from the target distribution.

5 CONCLUSION

We proposed the regularization loss that minimizes the upper bound of the 1-Wasserstein distance between the standard normal distribution and the distribution of activations. In image classification and language modeling experiments, PER gives marginal but consistent improvements over methods based on sample statistics (BN and VCL) as well as $L^1$ and $L^2$ activation regularization methods. The analysis of changes in activations’ distribution during training verifies that PER can stabilize the probability distribution of activations without normalization. Considering that the regularization loss can be easily applied to a wide range of tasks without changing architectures or training strategies
unlike BN, we believe that the results indicate the valuable potential of regularizing networks in the probability distribution space as a future direction of research.

The idea of regularizing activations with the metric in probability distribution space can be extended to many useful applications. For instance, one can utilize task-specific prior when determining a target distribution, e.g., the Laplace distribution for making sparse activation. The empirical distribution of activations computed by a pretrained network can also be used as a target distribution to prevent catastrophic forgetting. In this case, the activation distribution can be regularized so that it does not drift away from the activation distribution learned in the previous task as different from previous approaches constrains the changes in the parameter $l_2$ space (Kirkpatrick et al., 2017) or the function $L^2$ space (Benjamin et al., 2018).

REFERENCES


