Utilizing Class Information for Deep Network Representation Shaping

Daeyoung Choi,[*] **Wonjong Rhee**[*]
Department of Transdisciplinary Studies
Seoul National University
Seoul, 08826, South Korea
{choid, wrhee}@snu.ac.kr

Abstract

Statistical characteristics of deep network representations, such as sparsity and correlation, are known to be relevant to the performance and interpretability of deep learning. When a statistical characteristic is desired, often an adequate regularizer can be designed and applied during the training phase. Typically, such a regularizer aims to manipulate a statistical characteristic over all classes together. For classification tasks, however, it might be advantageous to enforce the desired characteristic per class such that different classes can be better distinguished. Motivated by the idea, we design two class-wise regularizers that explicitly utilize class information: class-wise Covariance Regularizer (cw-CR) and class-wise Variance Regularizer (cw-VR). cw-CR targets to reduce the covariance of representations calculated from the same class samples for encouraging feature independence. cw-VR is similar, but variance instead of covariance is targeted to improve feature compactness. For the sake of completeness, their counterparts without using class information, Covariance Regularizer (CR) and Variance Regularizer (VR), are considered together. The four regularizers are conceptually simple and computationally very efficient, and the visualization shows that the regularizers indeed perform distinct representation shaping. In terms of classification performance, significant improvements over the baseline and L1/L2 weight regularization methods were found for 21 out of 22 tasks over popular benchmark datasets. In particular, cw-VR achieved the best performance for 13 tasks including ResNet-32/110.

Introduction

For deep learning, a variety of regularization techniques have been developed by focusing on the *weight parameters*. A classic example is the use of L2 (Hoerl and Kennard 1970) and L1 (Tibshirani 1996) weight regularizers. They have been popular because they are easy to use, computationally light, and often result in performance enhancements. Another example is the parameter sharing technique that enforces the same weight values as in the Convolutional Neural Networks (CNNs). Regularization techniques that focus on the *representation* (the activations of the units in a deep network), however, have been less popular even though the performance of deep learning is known to depend on the learned
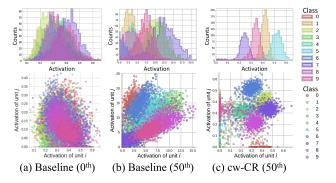
---
[*]Authors contributed equally.

Figure 1: A single unit's activation histogram (upper three plots) and two randomly chosen units' activation scatter plots (lower three plots) for MNIST. For a 6-layer Multilayer Perceptron (MLP), the fifth layer's representation vectors calculated using 10,000 test samples were used to generate the plots. For the baseline model, a substantial overlap among different classes can be observed at the time of initialization as shown in (a). Even after 50 epochs of training, still, a substantial overlap can be observed as shown in (b). When class information is used to regularize the representation shapes, the overlap is significantly reduced as shown in (c). Note that a slight correlation between each pair of classes can be observed in the scatter plot of (b), but not in that of (c) due to the use of cw-CR. The figures are best viewed in color.

representation heavily.

For representation shaping (regularization), some of the promising methods for performance and interpretability include (Glorot, Bordes, and Bengio 2011; Cogswell et al. 2015; Liao et al. 2016). (Glorot, Bordes, and Bengio 2011) considers increasing representational sparsity, (Cogswell et al. 2015) focuses on reducing covariance among hidden units, and (Liao et al. 2016) forces parsimonious representations using k-means style clustering. While all of them are effective representation regularizers, none of them explicitly use class information for the regularization. A few recent works (Wen et al. 2016b; Belharbi et al. 2017; Yang et al. 2018) do utilize class information, and their approaches are based on *hidden layer activation vectors*. The method of (Belharbi et al. 2017) is computationally expen-

sive because pair-wise dissimilarities need to be calculated among the same class samples in each mini-batch.

In this work, two computationally light representation regularizers, cw-CR (class-wise Covariance Regularizer) and cw-VR (class-wise Variance Regularizer), that utilize class information are introduced and studied. We came up with the design ideas by observing typical histograms and scatter plots of deep networks as shown in Figure 1. In Figure 1 (b), different classes substantially overlap even after the training is complete. If we directly use class information in regularization, as opposed to using it only for cross-entropy cost calculation, we can specifically reduce overlaps or pursue a desired representation characteristic. An example of cw-CR reducing class-wise covariance is shown in Figure 1 (c), and later we will show that cw-VR can notably reduce class-wise variance resulting in minimal overlaps. The two class-wise regularizers are very simple and computationally efficient, and therefore can be easily used as L1 or L2 weight regularizers that are very popular.

Our Contributions

The contributions of this work can be summarized as follows.

Introduction of three new representation regularizers We introduce two representation regularizers that utilize class information. cw-CR and cw-VR reduce per-class covariance and variance, respectively. In this work, their penalty loss functions are defined, and their gradients are analyzed and interpreted. Also, we investigate VR that is cw-VR's all-class counterpart. Intuitively, reducing the variance of each unit's activations does not make sense unless it is applied per class, but we have tried VR for the sake of completeness and found that VR is useful for performance enhancement. cw-CR's all-class counterpart, CR, is analyzed as well, but CR turns out to be the same as DeCov that was already studied in-depth in (Cogswell et al. 2015).

Performance improvement with the new representation regularizers Rather than trying to find a single case of beating the state-of-the-art record, we performed an extensive set of experiments on the most popular datasets (MNIST, CIFAR-10, CIFAR-100) and architectures (MLP, CNN). Additionally, ResNet (He et al. 2016) was tested as an example of a sophisticated network, and an image reconstruction task using autoencoder was tested as an example of a different type of task. We have tested a variety of scenarios with different optimizers, number of classes, network size, and data size. The results show that our representation regularizers outperform the baseline (no regularizer) and L1/L2 weight regularizers for almost all the scenarios that we have tested. More importantly, class-wise regularizers (cw-CR, cw-VR) usually outperformed their all-class counterparts (CR, VR). Typically cw-VR was the best performing regularizer and achieved the best performance for the autoencoder task, too.

Effects of representation regularization Through visualizations and quantitative analyses, we show that the new representation regularizers indeed shape representations in the ways that we have intended. The quantitative analysis of representation characteristics, however, indicates that each regularizer affects multiple representation characteristics together and therefore the regularizers cannot be used to control a single representation characteristic without at least mildly affecting some other representation characteristics.

Related Works

Regularization for Deep Learning

The classic regularizers apply L2 (Hoerl and Kennard 1970) and L1 (Tibshirani 1996) penalties to the *weights* of models, and they are widely used for Deep Neural Networks (DNNs) as well. (Wen et al. 2016a) extended L1 regularizers by using group lasso to regularize the structures of DNN (i.e., filters, channels, filter shapes, and layer depth). (Srivastava et al. 2014) devised dropout that randomly applies activation masking over the units. While dropout is applied in a multiplicative manner, (Glorot, Bordes, and Bengio 2011) used L1 penalty regularization on the activations to encourage sparse representations. XCov proposed by (Cheung et al. 2014) minimizes the covariance between autoencoding units and label encoding units of the same layer such that representations can be disentangled. Batch normalization (BN) proposed by (Ioffe and Szegedy 2015) exploits mini-batch statistics to normalize activations. It was developed to accelerate training speed by preventing internal covariate shift, but it was also found to be a useful regularizer. In line with batch normalization, weight normalization, developed by (Salimans and Kingma 2016), uses mini-batch statistics to normalize weight vectors. Layer normalization proposed by (Ba, Kiros, and Hinton 2016) is a RNN version of batch normalization, where they compute the mean and variance used for normalization from all of the summed inputs to the units in a layer on a single training case. There are many other publications on regularization techniques for deep learning, but we still do not fully understand how they really affect the performance. Recent work by (Zhang et al. 2016) shows that the traditional concept of controlling generalization error by regularizing the effective capacity does not apply to the modern DNNs.

Penalty Regularization on Representations

Some of the existing regularization methods explicitly shape representations by adopting a penalty regularization term. DeCov (Cogswell et al. 2015) is a penalty regularizer that minimizes the off-diagonals of a layer's representation covariance matrix. DeCov reduces co-adaptation of a layer's units by encouraging the units to be decorrelated. In this work, it is called as CR (Covariance Regularizer) for consistent naming. A recent work (Liao et al. 2016) used a clustering based regularization that encourages parsimonious representations. In their work, similar representations in sample, spatial, and channel dimensions are clustered and used for regularization such that similar representations are encouraged to become even more similar. While their work can be applied to unsupervised as well as supervised tasks, our work utilizes a much simpler and computationally efficient method of directly using class labels during training to

avoid k-means like clustering.

Class-wise Learning

True class information has been rarely used directly for regularization methods. Traditionally, the class information has been used only for evaluating the correctness of predictions and the relevant cost function terms. Some of the recent works, however, have adopted the class-wise concept in more sophisticated ways. In those works, class information is used as a switch or for emphasizing the discriminative aspects over different classes. As an example, (Li, Pan, and Chu 2008) proposed a kernel learning method using class information to model the manifold structure. They modify locality preserving projection to be class dependent. (Jiang, Lin, and Davis 2011) added label consistent regularizers for learning a discriminative dictionary. (Wen et al. 2016b) developed a regularizer called center loss that reduces the activation vector distance between representations and their corresponding class centers for face recognition tasks. (Yang et al. 2018) designed a loss function named prototype loss that improves representation's intra-class compactness for enhancing the robustness of CNN. Another recent work by (Belharbi et al. 2017) directly uses class labels to encourage similar representations per class as in our work, but it is computationally heavy as explained earlier. Besides the pair-wise computation, two optimizers are used for handling the supervised loss term and the hint term separately. Class information is used for autoencoder tasks as well. (Shi et al. 2016) implicitly reduced the intra-class variation of reconstructed samples by minimizing pair-wise distances among same class samples. Like the strategies listed above, our cw-VR and cw-CR use class-wise information to control the statistical characteristics of representations. However, our methods are simple because only one optimizer is used, and computationally efficient because pair-wise computation is not required.

Class-wise Representation Regularizers: cw-CR and cw-VR

In this section, we first present basic statistics of representations. Then, three representation regularizers, cw-CR, cw-VR, and VR are introduced with their penalty loss functions and gradients. Interpretations of the loss functions and gradients are provided as well.

Basic Statistics of Representations

For the layer l, the output activation vector of the layer is defined as $\mathbf{z}_l = \max(\mathbf{W}_l^\top \mathbf{z}_{l-1} + \mathbf{b}_l, 0)$ using Rectified Linear Unit (ReLU) activation function. Because we will be focusing on the layer l for most of the explanations, we drop the layer index. Then, z_i is the ith element of $\mathbf{z}$ (i.e. activation of ith unit).

To use statistical properties of representations, we define mean of unit i, μ_i, and covariance between unit i and unit j,

$c_{i,j}$, using the N samples in each mini-batch.

$$\mu_i = \frac{1}{N} \sum_n z_{i,n} \tag{1}$$

$$c_{i,j} = \frac{1}{N} \sum_n (z_{i,n} - \mu_i)(z_{j,n} - \mu_j) \tag{2}$$

Here, $z_{i,n}$ is the activation of unit i for n^{th} sample in the mini-batch. From equation (2), variance of i unit can be written as the following.

$$v_i = c_{i,i} \tag{3}$$

When class-wise statistics need to be considered, we choose a single label k from K labels and evaluate mean, covariance, and variance using only the data samples with true label k in the mini-batch.

$$\mu_i^k = \frac{1}{|S_k|} \sum_{n \in S_k} z_{i,n} \tag{4}$$

$$c_{i,j}^k = \frac{1}{|S_k|} \sum_{n \in S_k} (z_{i,n} - \mu_i^k)(z_{j,n} - \mu_j^k) \tag{5}$$

$$v_i^k = c_{i,i}^k \tag{6}$$

Here, S_k is the set containing indexes of the samples whose true label is k, and $|S_k|$ is the cardinality of the set S_k.

cw-CR

cw-CR uses off-diagonal terms of the mini-batch covariance matrix of activations per class as the penalty term: $\Omega_{cw\text{-}CR} = \sum_k \sum_{i \neq j} (c_{i,j}^k)^2$. This term is added to the original cost function J, and the total cost function $\widetilde{J}$ can be denoted as

$$\widetilde{J} = J + \lambda \Omega_{cw\text{-}CR}(\mathbf{z}), \tag{7}$$

where λ is the penalty loss weight ($\lambda \in [0, \infty)$). The penalty loss weight balances between the original cost function J and the penalty loss term Ω. When λ is equal to zero, $\widetilde{J}$ is the same as J, and cw-CR does not influence the network. When λ is a positive number, the network is regularized by cw-CR, and the performance is affected. In practice, we have observed that deep networks with too large λ cannot be trained at all.

cw-VR

A very intuitive way of enforcing distinguished representations per class is to maximize the inter-class distances in the representation space. Because inter-class needs to be maximized, the corresponding penalty term can be inverted or multiplied by -1 before it is minimized with the original cost function. We tried such approaches, but the optimization became unstable (failed to converge). An alternative way is to reduce intra-class (same-class) variance. By applying this idea, the penalty loss term of cw-VR can be formulated as $\Omega_{cw\text{-}VR} = \sum_k \sum_i v_i^k$.

With the design of cw-VR, we naturally invented VR that is the all-class counterpart of cw-VR. VR minimizes the activation variance of each unit, and it is mostly the same as

Table 1: Penalty loss functions and gradients of the representation regularizers. All the penalty loss functions are normalized with the number of units (I) and the number of classes (K) such that the value of λ can have a consistent meaning. CR and cw-CR are standardized using the number of distinct covariance combinations.

Penalty loss function		Gradient	
Ω_{CR}	$= \dfrac{2}{I(I-1)} \sum_{i \neq j} (c_{i,j})^2$	$\dfrac{\partial \Omega_{CR}}{\partial z_{i,n}}$	$= \dfrac{4}{NI(I-1)} \sum_{j \neq i} c_{i,j}(z_{j,n} - \mu_j)$
$\Omega_{cw\text{-}CR}$	$= \dfrac{2}{KI(I-1)} \sum_{k} \sum_{i \neq j} (c_{i,j}^k)^2$	$\dfrac{\partial \Omega_{cw\text{-}CR}}{\partial z_{i,n}}$	$= \dfrac{4}{KI(I-1)|S_k|} \sum_{j \neq i} c_{i,j}^k(z_{j,n} - \mu_j^k), n \in S_k$
Ω_{VR}	$= \dfrac{1}{I} \sum_{i} v_i$	$\dfrac{\partial \Omega_{VR}}{\partial z_{i,n}}$	$= \dfrac{2}{NI}(z_{i,n} - \mu_i)$
$\Omega_{cw\text{-}VR}$	$= \dfrac{1}{KI} \sum_{k} \sum_{i} v_i^k$	$\dfrac{\partial \Omega_{cw\text{-}VR}}{\partial z_{i,n}}$	$= \dfrac{2}{KI|S_k|}(z_{i,n} - \mu_i^k), n \in S_k$

cw-VR except for not using the class information. We expected VR to hurt the performance of deep networks because it encourages all classes to have similar representation in each unit. VR, however, turned out to be effective and useful for performance enhancement. We provide a possible explanation in the Experiments section.

Penalty Loss Functions and Gradients

The penalty loss functions of cw-CR and cw-VR are similar to CR and VR, respectively, except that the values are calculated for each class using the mini-batch samples with the same class label. Also, gradients of CR and cw-CR are related to those of VR and cw-VR as shown in Table 1. We investigate more details of the equations in the following.

Interpretation of the gradients Among the gradient equations shown in Table 1, the easiest to understand is VR's gradient. It contains the term $z_{i,n} - \mu_i$, indicating that the representation $z_{i,n}$ of each sample n is encouraged to become closer to the mean activation μ_i. In this way, each unit's variance can be reduced. For cw-VR, the equation contains $z_{i,n} - \mu_i^k$ instead of $z_{i,n} - \mu_i$. Therefore the representation $z_{i,n}$ of a class k sample is encouraged to become closer to the *class* mean activation μ_i^k. Clearly, the variance reduction is applied per class by cw-VR.

For CR, the equation is less straightforward. As explained in (Cogswell et al. 2015), a possible interpretation is that the covariance term $c_{i,j}$ is encouraged to be reduced where $z_{j,n} - \mu_j$ acts as the weight. But, another possible interpretation is that $z_{j,n}$ is encouraged to become closer to μ_j just as in the case of VR, where $c_{i,j}$ acts as the weight. Note that VR's mechanism is straightforward where each unit's variance is directly addressed in the gradient equation of activation i, but CR's mechanism is slightly complicated where all variances over all activations of j ($j = 1, ..., I$, where $j \neq i$) are collectively addressed through the summation terms over all j ($j = 1, ..., I$, where $j \neq i$). Thus, one can interpret CR as a hybrid regularizer that wants either or both of covariance and variance to be reduced. This can be the reason why the visualizations of CR and VR are similar as will be shown in Figure 2 later.

For cw-CR, it can be interpreted similarly. As in the re-

lationship between VR and cw-VR, cw-CR is the class-wise counterpart of CR and it can be confirmed in the gradient equation: cw-CR has $c_{i,j}^k(z_{j,n} - \mu_j^k)$ instead of $c_{i,j}(z_{j,n} - \mu_j)$. As in our explanation of CR, cw-CR can also be interpreted as trying to reduce either or both of covariance and variance. The visualizations of cw-CR and cw-VR turn out to be similar as well.

The interpretations can be summarized as follows. VR and cw-VR aim to reduce activation variance whereas CR and cw-CR additionally aim to reduce covariance. CR and VR do not distinguish among different classes, but cw-CR and cw-VR explicitly perform representation shaping per class.

Activation squashing effect There is another important effect that is not necessarily obvious from the gradient formulations. For L1W (L1 weight regularization) and L2W (L2 weight regularization), the gradients contain the weight terms, and therefore the weights are explicitly encouraged to become smaller. Similarly, our representation regularizers include the activation terms $z_{i,n}$ and therefore the activations are explicitly encouraged to become smaller (when activations become close to zero, the mean terms become close to zero as well). Thus, a simple way to reduce the penalty loss is to scale the activations to small values instead of satisfying the balance between the terms in the gradient equations. This means that there is a chance for the learning algorithm to squash activations just so that the representation regularization term can be ignored. As we will see later in the next section, indeed activation squashing happens when our regularizers are applied. Nonetheless, we will also show that the desired statistical properties are sufficiently manifested anyway. One might be able to prevent activation squashing with another regularization technique, but such an experiment was not in the scope of this work.

Experiments

In this section, we investigate performance improvements of the four representation regularizers, where baseline, L1W, L2W, CR, cw-CR, VR, and cw-VR are evaluated for image classification and reconstruction tasks. When a regularizer (including L1W and L2W) was used for an evalu-

Table 2: Error performance (%) for CIFAR-10 CNN model.

Regularizer	Optimizer	
	Adam	Momentum
Baseline	26.64 ± 0.16	25.78 ± 0.37
L1W	26.46 ± 0.39	25.73 ± 0.40
L2W	25.71 ± 0.98	26.35 ± 0.54
CR	24.96 ± 0.63	26.72 ± 0.61
cw-CR	22.99 ± 0.58	25.93 ± 0.59
VR	$\mathbf{21.44 \pm 0.88}$	25.01 ± 0.41
cw-VR	21.58 ± 0.21	$\mathbf{24.42 \pm 0.31}$

Table 3: Error performance (%) for CIFAR-100 CNN model.

Regularizer	Number of Classes		
	16	64	100
Baseline	45.75 ± 0.73	58.02 ± 0.40	61.26 ± 0.52
L1W	45.08 ± 1.53	58.08 ± 1.18	60.97 ± 0.64
L2W	45.28 ± 1.59	57.47 ± 0.66	60.23 ± 0.31
CR	44.55 ± 1.10	56.76 ± 0.86	59.88 ± 0.50
cw-CR	43.50 ± 1.21	54.24 ± 0.64	57.03 ± 0.73
VR	42.33 ± 1.03	54.32 ± 0.40	57.68 ± 0.94
cw-VR	$\mathbf{41.38 \pm 0.53}$	$\mathbf{54.23 \pm 1.06}$	$\mathbf{56.75 \pm 0.64}$

ation scenario, the penalty loss weight λ was determined as one of {0.001, 0.01, 0.1, 1, 10, 100} using 10,000 validation samples. Once the λ was determined, performance evaluation was repeated five times. Code is made available at https://github.com/snu-adsl/class_wise_regularizer.

Image Classification Task

Three popular datasets (MNIST, CIFAR-10, and CIFAR-100) were used as benchmarks. An MLP model was used for MNIST, and a CNN model was used for CIFAR-10/100. The details of the architecture hyperparameters can be found in Section A of the supplementary materials. All the regularizers were applied to the fifth layer of the 6-layer MLP model and the fully connected layer of the CNN model, and the reason will be explained in the Layer Dependency section. For L1W and L2W, we applied regularization to all the layers as well for comparison, but the performance results were comparable to when applied to the fifth layer. Mini-batch size was increased to 500 for CIFAR-100 such that class-wise operations can be appropriately performed but was kept at the default value of 100 for MNIST and CIFAR-10. We have tested a total of 20 scenarios where the choice of an optimizer, number of classes, network size, or data size was varied.

The results for two CIFAR-10 CNN scenarios are shown in Table 2 and three CIFAR-100 CNN scenarios are shown in Table 3. The rest of the scenarios including full cases of MNIST MLP can be found in Section B of the supplementary materials. In the Table 2 and Table 3, it can be seen that cw-VR achieves the best performance in 4 out of 5 cases and class-wise regularizers perform better than their all-class counterparts except for one case. For the scenarios shown in Table 3, we initially guessed that the performance of class-wise regularizers would be sensitive to the number

Table 4: Error performance (%) for ResNet-32/110 (CIFAR-10). For ResNet-32, average of two experiments is shown. For ResNet-110, we experimented five times and 'best (mean±std)' is reported as in (He et al. 2016).

Model & Regularizer	He et al.	Ours
ResNet-32	7.51	7.39
ResNet-32 + CR		7.27
ResNet-32 + cw-CR		7.21
ResNet-32 + VR		7.22
ResNet-32 + cw-VR		**7.17**
ResNet-110	6.43 (6.61±0.16)	6.12 (6.31±0.14)
ResNet-110 + CR		6.17 (6.26±0.05)
ResNet-110 + cw-CR		6.10 (6.18±0.10)
ResNet-110 + VR		6.10 (6.17±0.05)
ResNet-110 + cw-VR		**6.00** (6.18±0.15)

of classes, but cw-VR performed well for all three cases. As for the 20 scenarios that were tested, the best performing one was cw-VR for 11 cases, VR for 5 cases, cw-CR for 2 cases, and CR for 1 case. L1W and L2W were never the best performing one, and the baseline (no regularization) performed the best for only one case.

As mentioned earlier, in general, VR did not hurt performance compared to the baseline. There are two possible explanations. First, representation characteristics other than variance are affected together by VR (see Table 5 in the next section), and VR might have indirectly created a positive effect. Second, the cross-entropy term limits how much VR performs variance reduction, and the overall effects might be more complicated than a simple variance reduction.

To test a sophisticated and advanced DNN architecture, we tried the four representation regularizers on ResNet-32/110. ResNet is known as one of the best performing deep networks for CIFAR-10, and we applied the four representation regularizers to the output layer without modifying the network's architecture or hyperparameters. The results are shown in Table 4. All four turned out to have positive effects where cw-VR showed the best performance again.

Image Reconstruction Task

In order to test a completely different type of task, we examined an image reconstruction task where a deep autoencoder are used. Class information is used for representation regularization only. A 6-hidden layer autoencoder with a standard L2 objective function was used. Representation regularizers were only applied to the third layer because the representations of the layer are considered as latent variables. The other experiment settings are the same as the image classification tasks in the previous subsection. The reconstruction error of the baseline is 1.44×10^{-2} and become reduced to 1.19×10^{-2} when cw-VR is applied. Result details can be found in Section B of the supplementary materials. As in the classification tasks, class-wise regularizers performed better than their all-class counterparts.

Representation Characteristics

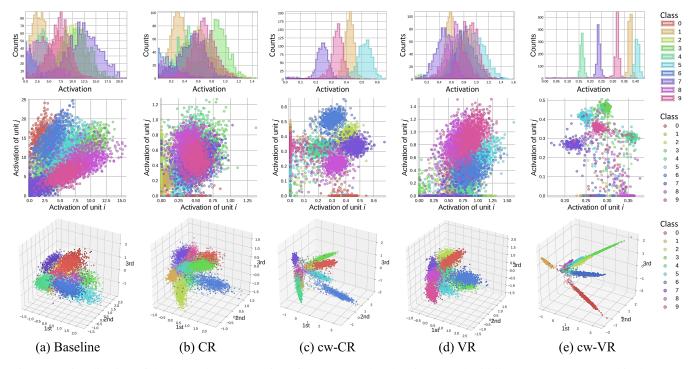In this section, we investigate representation characteristics when the regularizers are applied.

Figure 2: Visualization of the learned representations for MNIST. The plots in top and middle rows were generated in the same way as in the Figure 1. The plots in the bottom row show the top three principle components of the representations.

(a) Baseline   (b) CR   (c) cw-CR   (d) VR   (e) cw-VR

Table 5: Quantitative evaluations of representation characteristics.

Regularizer	Test error (%)	ACTIVATION_AMPLITUDE	COVARIANCE (CR)	CORRELATION (CR)	CW_CORRELATION (cw-CR)	VARIANCE (VR)	N_CW_VARIANCE (cw-VR)
Baseline	2.85 ± 0.11	4.93	2.08	0.27	0.21	9.05	1.33
L1W	2.85 ± 0.06	4.53	1.95	0.28	0.22	7.78	1.33
L2W	3.02 ± 0.40	4.76	2.23	0.29	0.21	8.38	1.36
CR	2.50 ± 0.05	*0.50*	0.01	**0.19**	0.15	0.04	1.37
cw-CR	2.49 ± 0.10	*0.63*	0.02	0.31	**0.19**	0.06	0.95
VR	2.65 ± 0.11	*1.35*	0.15	0.26	0.17	**0.58**	1.52
cw-VR	$\mathbf{2.42 \pm 0.06}$	*0.63*	0.02	0.36	0.25	0.05	**0.74**

Visualization

In Figure 2, the 50th epoch plots are shown for the baseline and four representation regularizers. L1W and L2W are excluded because their plots are very similar to those of the baseline. Principle Component Analysis (PCA) was also performed over the learned representations, and the plots in the bottom row show the top three principal components of the representations (before ReLU). The first thing that can be noticed is that the representation characteristics are quite different depending on which regularizer is used. Apparently, the regularizers are effective at affecting representation characteristics. In the first row, it can be seen that cw-VR minimizes the activation overlaps among different classes as intended. Because the gradient equation of cw-CR is related to that of cw-VR, cw-CR also shows reduced overlaps. CR and VR still show substantial overlaps because class information was not used by them. In the second row, a linear correlation can be observed in the scatter plot of the baseline, but such a linear correlation is mostly removed for CR as expected. For VR, still, linear correlations can be observed. For cw-CR and

cw-VR, it is difficult to judge because many points do not belong to the main clusters and their effects on correlation are difficult to guess. As we will see in the following quantitative analysis section, in fact, correlation was not reduced for cw-CR and cw-VR. In the third row, it can be seen that the cw-VR has the least overlaps when the first three principal components are considered. Interestingly, a needle-like shape can be observed for each class in the cw-VR's plot. The plots using learned representations after ReLU are included in Section C of the supplementary materials. Overall, cw-VR shows the most distinct shapes compared to the baseline.

Quantitative Analysis

For the same MNIST task that was used to plot Figure 1 and Figure 2, the quantitative values of representation characteristics were evaluated, and the results are shown in Table 5. Each is calculated using only positive activations and is the average of representation statistics. For example, ACTI-VATION_AMPLITUDE is the mean of positive activations in

3401

a layer. In the third column (ACTIVATION_AMPLITUDE), it can be confirmed that indeed the four representation regularizers cause activation squashing. Nonetheless, the error performance is improved as shown in the second column. For CR, covariance is supposed to be reduced. In the fourth column (COVARIANCE), it can be confirmed that the covariance of CR is much smaller than that of the baseline. The small value, however, is mostly due to the activation squashing. In the fifth column (CORRELATION), the normalized version of covariance is shown. The correlation of CR is confirmed to be smaller than that of the baseline, but the reduction rate is much smaller compared to the covariance that was affected by the activation squashing. In any case, CR indeed reduces correlation among hidden units. For cw-CR, class-wise correlation (CW_CORRELATION) is expected to be small, and it is confirmed in the sixth column. The value 0.19, however, is larger than CR's 0.15 or VR's 0.17. This is an example where not only cw-CR but also other representation regularizers end up reducing CW_CORRELATION because the regularizers' gradient equations are related. For VR, variance should be reduced. In the seventh column (VARIANCE), the variance of VR is indeed much smaller than that of the baseline, but again other representation regularizers have even smaller values because their activation squashing is more severe than that of VR. For cw-VR, class-wise variance is supposed to be small. Normalized class-wise variance is shown in the last column (N_CW_VARIANCE), and it is confirmed that cw-VR is capable of reducing N_CW_VARIANCE. (Normalization was performed by mapping activation range of each hidden unit to [0,10] such that activation squashing effect can be removed.)

Layer Dependency

In the previous sections, we have consistently applied the representation regularizers to the upper layers that are closer to the output layer. This is because we have found that it is better to target the upper layers, and two exemplary results are shown in Figure 3. In Figure 3 (a), the performance improvement becomes larger as the representation regularization targets upper layers. In fact, the best performance is observed when the output layer is regularized. In Figure 3 (b), similar patterns can be seen over the convolutional layers, but the performance degrades when applied to fully connected or output layers. This phenomenon is probably relevant to how representations are developed in deep networks. Because the lower layers often represent many simpler concepts, regularizing the shapes of representations can be harmful. For the upper layers, a smaller number of more complex concepts are represented and therefore controlling representation characteristics (e.g., reduction of activation overlaps) might have a better chance to improve the performance.

Discussion and Conclusion

A well-known representation regularizer is L1 representation regularizer (L1R) whose penalty loss function can be written as $\Omega_{L1R} = \frac{1}{NI} \sum_n \sum_i |z_{i,n}|$. L1R is known to increase representational sparsity. CR and VR have second-
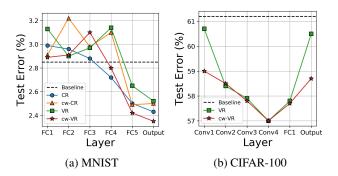

Figure 3: Layer dependency of representation regularizers. The x-axis indicates layers where regularizers are applied. CR and cw-CR are excluded in (b) due to the high computational burden of applying them to the convolutional layers. The result of CIFAR-10 can be found in Section D of the supplementary materials.

order terms in their penalty loss functions, but L1R does not. As a consequence, L1R's class-wise counterpart turns out to have the same penalty function as L1R's (this is trivial to prove). So, one might say that L1R is also a class-wise representation regularizer just like cw-CR and cw-VR. When it is used, however, there is no need for the true class information. For instance, when true label information is not available for an autoencoder problem, one might use L1R and still have a chance to obtain the benefits of class-wise regularization. In our study, we have not included L1R such that we can better focus on the difference between all-class and class-wise regularizers. When cw-VR was directly compared with L1R in terms of performance, we have found that cw-VR performs better than L1R for 12 out of the 21 test scenarios (ResNet-110 and an autoencoder were not tested). Overall, however, it looks like both L1R and cw-VR are very effective representation regularizers for improving performance of deep networks.

Dropout and batch normalization are very popular regularizers, but they are fundamentally different because they are not 'penalty cost function' regularizers. Instead, they are implemented by directly affecting the feedforward calculations during training. Dropout has been shown to have similar effects as ensemble and data-augmentation through its noisy training procedure, and such benefits are not obtainable with a penalty regularizer. On the other hand, there is a common belief that 'dropout reduces co-adaptation (or pairwise correlation).' Reducing correlation is something that can be done by penalty regularizers as we have shown in this work. When we applied the same quantitative analysis on the test scenarios while using dropout, however, we have found that dropout does not really reduce the correlation. This indicates that the belief might be an incorrect myth. Batch normalization has been known to have a stabilization effect because it can adjust covariate shift even when the network is in the early stage of training. Thus a higher learning rate can be used for faster training. Such an effect is not something that can be achieved with a penalty

regularizer. But when dropout and batch normalization were directly compared with the two representation regularizers cw-VR and L1R in terms of performance, we have found that at least one of cw-VR and L1R outperforms both of dropout and batch normalization for 16 out of the 20 test cases (ResNet-32/110 and an autoencoder were not tested). Despite the performance results for our benchmark scenarios, it is important to recognize that dropout and batch normalization might be able to play completely different roles that cannot be addressed by the penalty regularizers. When such additional roles are not important for a task as in our test scenarios, there is a very high chance of penalty regularizers outperforming dropout and batch normalization.

Performance improvement through representation regularizers, especially by utilizing class information, has been addressed in this work and other previous works. The underlying mechanism for the improvement, however, is still unclear. Recently, (Choi et al. 2018) showed that some of the statistical properties of representations cannot be the direct cause of performance improvement. The representation regularizers might have tuning effects instead.

With the enormous efforts of the research community, deep learning is becoming better understood, and regularization techniques are evolving with the in-depth understandings. In this work, we have addressed the fundamentals of using class information for penalty representation regularization. The results indicate that class-wise representation regularizers are very efficient and quite effective, and they should be considered as important and high-potential configurations for learning of deep networks.

Acknowledgments

References

Ba, J. L.; Kiros, J. R.; and Hinton, G. E. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Belharbi, S.; Chatelain, C.; Herault, R.; and Adam, S. 2017. Neural networks regularization through invariant features learning. *arXiv preprint arXiv:1709.01867*.

Cheung, B.; Livezey, J. A.; Bansal, A. K.; and Olshausen, B. A. 2014. Discovering hidden factors of variation in deep networks. *arXiv preprint arXiv:1412.6583*.

Choi, D.; Lee, K.; Shin, C.; and Rhee, W. 2018. On the statistical and information-theoretic characteristics of deep network representations. *arXiv preprint arXiv:1811.03666*.

Cogswell, M.; Ahmed, F.; Girshick, R.; Zitnick, L.; and Batra, D. 2015. Reducing overfitting in deep networks by decorrelating representations. *arXiv preprint arXiv:1511.06068*.

Glorot, X.; Bordes, A.; and Bengio, Y. 2011. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 315–323.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hoerl, A. E., and Kennard, R. W. 1970. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics* 12(1):55–67.

Ioffe, S., and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, 448–456.

Jiang, Z.; Lin, Z.; and Davis, L. S. 2011. Learning a discriminative dictionary for sparse coding via label consistent k-svd. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, 1697–1704. IEEE.

Li, J.-B.; Pan, J.-S.; and Chu, S.-C. 2008. Kernel classwise locality preserving projection. *Information Sciences* 178(7):1825–1835.

Liao, R.; Schwing, A.; Zemel, R.; and Urtasun, R. 2016. Learning deep parsimonious representations. In *Advances in Neural Information Processing Systems*, 5076–5084.

Salimans, T., and Kingma, D. P. 2016. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems*, 901–909.

Shi, H.; Zhu, X.; Lei, Z.; Liao, S.; and Li, S. Z. 2016. Learning discriminative features with class encoder. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 46–52.

Srivastava, N.; Hinton, G. E.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research* 15(1):1929–1958.

Tibshirani, R. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)* 267–288.

Wen, W.; Wu, C.; Wang, Y.; Chen, Y.; and Li, H. 2016a. Learning structured sparsity in deep neural networks. In *Advances in Neural Information Processing Systems*, 2074–2082.

Wen, Y.; Zhang, K.; Li, Z.; and Qiao, Y. 2016b. A discriminative feature learning approach for deep face recognition. In *European Conference on Computer Vision*, 499–515. Springer.

Yang, H.-M.; Zhang, X.-Y.; Yin, F.; and Liu, C.-L. 2018. Robust classification with convolutional prototype learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3474–3482.

Zhang, C.; Bengio, S.; Hardt, M.; Recht, B.; and Vinyals, O. 2016. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*.