Ph.D. Dissertation of Engineering

Deep Network Regularization with Representation Shaping

February 2019

Graduate School of
Convergence Science and Technology
Seoul National University
Program in Digital Contents and Information

Daeyoung Choi

Deep Network Regularization with Representation Shaping

Advisor Wonjong Rhee

Submitting a Ph.D. Dissertation of Engineering

February 2019

Graduate School of
Convergence Science and Technology
Seoul National University
Program in Digital Contents and Information

Daeyoung Choi

Confirming the Ph.D. Dissertation written by
Daeyoung Choi
February 2019

Chair	Kyogu Lee	(Seal) [illegible]
Vice Chair	Nojun Kwak	(Seal) [illegible]
Examiner	Wonjong Rhee	(Seal) [illegible]
Examiner	Bongwon Suh	(Seal) [illegible]
Examiner	Yongjae Lee	(Seal) [illegible]

Abstract

The statistical characteristics of learned representations such as correlation and representational sparsity are known to be relevant to the performance of deep learning methods. Also, learning meaningful and useful data representations by using regularization methods has been one of the central concerns in deep learning. In this dissertation, deep network regularization using representation shaping are studied. Roughly, the following questions are answered: what are the common statistical characteristics of representations that high-performing networks share? Do the characteristics have a causal relationship with performance? To answer the questions, five representation regularizers are proposed: class-wise Covariance Regularizer (cw-CR), Variance Regularizer (VR), class-wise Variance Regularizer (cw-VR), Rank Regularizer (RR), and class-wise Rank Regularizer (cw-RR). Significant performance improvements were found for a variety of tasks over popular benchmark datasets with the regularizers. The visualization of learned representations shows that the regularizers used in this work indeed perform distinct representation shaping. Then, with a variety of representation regularizers, a few statistical characteristics of learned representations including covariance, correlation, sparsity, dead unit, and rank are investigated. Our theoretical analysis and experimental results indicate that all the statistical characteristics considered in this work fail to show any general or causal pattern for improving performance. Mutual information $I(\mathbf{z};\mathbf{x})$ and $I(\mathbf{z};\mathbf{y})$ are examined as well, and it is shown that regularizers can affect $I(\mathbf{z};\mathbf{x})$ and thus indirectly influence the performance. Finally, two practical ways of using representation regularizers are presented to address the usefulness of representation regularizers: using a set of representation regularizers as a performance tuning tool and enhancing network compression with representation regularizers.

Keywords: deep learning, representation learning, deep neural network, regularization
Student number: 2015-30722

Table of Contents

List of Tables

List of Figures

Chapter 1. Introduction

Despite the recent success of deep learning in a wide range of applications, in-depth understanding of deep learning is still under its early phase of research, and it has become a critical and timely topic. Toward a better understanding of deep learning, a variety of theoretical and empirical approaches have been proposed, and some of them have made a meaningful progress. Statistical learning theory, approximation theory, information theory, optimization, and deep representations are some of the topics that are being actively discussed in the research community. Based on the recent theoretical studies of deep learning, this dissertation focuses on the deep representations. In this chapter, we begin this dissertation with the research background and motivation followed by a summary of the contributions.

1.1. Background and Motivation

Theoretical understanding of deep learning is essential in research, and the properties of high-performing networks obtained from theoretical analysis can be beneficial for designing new network architectures. Many works have been performed to understand deep networks, but a deep network is still considered as a black-box model. This is because a deep network typically has a few million parameters and nonlinear activation functions thus making it highly nonconvex. Furthermore, it is difficult to describe a deep network because a real-world dataset almost always cannot be written in a clean mathematical form.

A deep network consists of multiple hidden layers including hidden units (neurons) between input and output layers, and nonlinear activation functions are used to capture the complicated relationship between the input and output. After the architecture and hyperparameters of a deep network are defined, the network is trained by applying a backpropagation algorithm. The trained network is a deterministic function of the input, and the input data flow from the input layer to the output layer through hidden layers. Therefore, the trained network can be considered as a Markov chain that consists of consecutive hidden representations of the input data. That is, the network encodes and decodes the input data and hidden representations. By doing this, the hidden layers automatically capture the underlying explanatory factors of data related to a task, which is the main reason why deep learning methods often outperform the other machine learning algorithms even after applying hand-crafted feature engineering.

It is well known that a larger network with more hidden layers and units often performs better than a smaller network. This is because the expressivity of deep network increases by adding more hidden layers and units. As a result, the network can capture task-relevant information from input data more easily and efficiently and form a better representation via the advantages of distributed and deep representations. Thus, deep networks are designed to have more than a few million parameters for obtaining good performance in practice. For example, ResNet-110 model, one of the state-of-the-art network architectures, has 1.7 million parameters (He et al. 2016).

The potential drawback of the large network is that overfitting could occur because the number of parameters significantly exceeds the number of training data. Surprisingly, overfitting rarely happens for the large network. The ResNet-110 model is trained using only 50,000 training samples but still has a small generalization error, which can be estimated as the difference between training error and test error. This small generalization error is difficult to explain with statistical learning theory that says a much smaller number of training data samples compared to the number of model parameters often leads to overfitting. It has been commonly believed that overfitting rarely occurs in deep networks because stochastic gradient descent implicitly regularizes the networks, and explicit regularization methods such as L2 weight decay and dropout (Srivastava et al. 2014) decrease the excess capacity of the networks. However, recent experimental work by Zhang et al. (2016) revealed that deep networks having small generalization error have large enough capacity to memorize random labels and randomly generated data. Furthermore, the performance gain by using explicit regularization methods is relatively very small compared to the performance obtained by the stochastic gradient decent alone. Therefore, it can be concluded that regularizers are not the main reason for the excellent generalization of deep networks and do not seem to play the same roles like those of traditional machine learning regularizers that limit the complexity of learning models.

Zhang et al. (2016) fueled the research in deep learning generalization, and many researchers have actively investigated the topic recently (Arpit et al. 2017; Krueger et al. 2017; Hoffer, Hubara, and Soudry 2017; Wu, Zhu, and E 2017; Dziugaite and Roy 2017; Dinh et al. 2017). Some research focuses on the theoretical generalization bound, and information-theoretic approaches (Shwartz-Ziv and Tishby 2017; Achille and Soatto 2018a; Saxe et al. 2018) have become popular in the deep learning theory field as well. These works have mostly focused on the reason for the excellent generalization performance that might have been caused by the implicit regularization via the stochastic gradient descent, but the roles of explicit regularization in deep networks have not been sufficiently studied yet. As mentioned above, training a deep network is a process to learn hidden representation. Each regularizer builds a representation that has different characteristics, so the characteristics of learned representations can have a connection with the performance of deep networks. Also, we might be able to use the characteristics to design high-performing networks if they exist. To this end, we first need to understand representation characteristics in-depth. In this dissertation, we study representation regularization of deep networks using representation shaping and mainly focus on design and the relationship between representation characteristics and performance.

In practice, developing a way to improve a task performance or build an efficient network based on an understanding of deep representation characteristics has become one of the important topics. As mentioned above, deep networks have numerous weight parameters, and thus there are many ways to build different networks with a comparable number of parameters. The types of layers, the number of hidden layers and units are the typical hyperparameters in deep learning, and the choice of hyperparameters, of course, can significantly affect the performance of deep learning models. Besides these architecture hyperparameters, there exist a variety of regularization methods that can be applied to deep networks and that can be considered as hyperparameters. If we consider multiple representation regularizers that manipulate each of representation characteristic, the regularizers can be used as hyperparameters. Plus, their combinations and penalty loss weights can significantly affect performance, too. Another practical way of using representation regularization is the network compression, and it can be used to build an efficient network. As mentioned above, deep networks are usually highly overparameterized, so unimportant parameters can be removed with a small or even no performance loss. By applying regularization methods that alter representation characteristics of deep networks, corresponding weight parameters can be compressed more efficiently. In this dissertation, motivated by the need for improving performance and building efficient networks, we propose two practical ways of using the regularizers as a tuning tool and network compression method.

1.2. Contributions

The main contributions of this dissertation are summarized as follows.

Chapter 3

Introduction of three new representation regularizers We introduce two representation regularizers that utilize class information. Class-wise Covariance Regularizer (cw-CR) and class-wise Variance Regularizer (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 Variance Regularizer (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, Covariance Regularizer (CR), is analyzed as well, but CR turns out to be the same as DeCov that was already studied in-depth (Cogswell et al. 2016).

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-32/110 (He et al. 2016) were tested as an example of a sophisticated network, and an image reconstruction task using deep 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 cw-VR 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.

Chapter 4

Identical Output Networks and Covariance and correlation of deep representations We show that for a deep network $\mathcal{N}_{\mathcal{A}}$, there exist infinitely many Identical

Output Networks (IONs) whose representation characteristics such as covariance and correlation are completely different from $\mathcal{N}_{\mathcal{A}}$'s but whose output for any given input $\mathbf{x}$ is the same as $\mathcal{N}_{\mathcal{A}}$'s. The stronger part of the results holds only for linear layers, but the empirical results suggest that the findings are applicable to ReLU layers as well. The existence of IONs implies that some of the representation characteristics should not be quoted as the reason for superior performance.

Sparsity, dead unit, and rank of deep representations We consider sparsity, dead unit, and rank to show that only loose relationships exist among the three representation characteristics. A higher sparsity or additional dead units do not imply a better or worse performance when the rank of representation is fixed. In particular, we develop Rank Regularizer (RR) that can control the *stable rank* of the representation and use the regularizer to empirically show that the number of independent factors in the data generation process does not support why one should use regularizers that encourage sparsity or lower rank.

Information-theoretic characteristics Mutual information $I(\mathbf{z};\mathbf{x})$ and $I(\mathbf{z};\mathbf{y})$ are investigated. Unlike the statistical characteristics of learned representation, mutual information directly measures the amount of useful and useless information using the joint density functions. When twelve different regularizers were tried on an MNIST image classification task, the resulting $I(\mathbf{z};\mathbf{x})$ and classification performance showed a strong correlation.

Chapter 5

Tuning deep network performance by using representation regularizers We compared the effects of twelve different regularizers on differently conditioned learning tasks (tasks with a fixed dataset, less training data samples, a smaller or larger layer width, different optimizers, or fewer target labels). We tested over MNIST, CIFAR-10, and CIFAR-100 datasets, and tried several architectures. The investigation results show that none of the twelve regularizers consistently outperforms for any conditioned learning task. While some of the regularizers perform very well, the well-performing regularizers are changed even by making a single change in the task constraint. Some of the existing works that compared against only a few regularizers to derive or imply a general conclusion might need to be revisited for a deeper investigation.

Enhancing network compression by using representation regularizers We showed that network compression could be enhanced by applying cw-VR to representations and RR to weight parameters. First, since some of the representation regularizers such as cw-VR and VR make representation compact, it can be expected that corresponding weight matrices are also compact, meaning that they have low rank. Therefore, they can be better compressed than those networks without any regularization. Second, RR can be applied to weight matrices to make the matrices lower rank, and thus resulting in better compression. Our results show that deep networks became better compressed by applying regularizations for both cases.

Chapter 2. Generalization, Regularization, and Representation in Deep Learning

In this work, we aim for an in-depth understanding of deep network regularization using representation shaping. To this end, we briefly review recent literature related to generalization, regularization, and representation in deep learning, and explain the relationship between the three and this work. We begin this chapter by describing the settings and terminologies of a deep network.

2.1. Deep Networks

Applying appropriate network architecture for datasets and tasks is known to be one of the most important factors contributing to the success of deep learning tasks. In applications such as image recognition (Krizhevsky, Sutskever, and Hinton 2012; Farabet et al. 2013; Tompson et al. 2014; Szegedy et al. 2015) and speech recognition (Mikolov et al. 2011; Hinton et al. 2012; Sainath et al. 2013), specific architectures such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) worked very well and led to the success of deep learning. However, the basis of all these network architectures is a Multilayer Perceptron (MLP) also called deep feedforward networks.

A deep network consists of multiple hidden layers, and a single processing layer is composed of multiple hidden units. A single parameter of a deep network is defined as a pair of units in two consecutive layers. That is, the connection from one unit (i) to another unit (j) is defined as $w_{i,j}$, and bias is defined as b. Data used to train the deep network is expressed as a pair of input $\mathbf{x}$ and label $\mathbf{y}$ for supervised learning. It is assumed that this data is selected from distribution $\mathcal{D}$ to i.i.d. In this work, we mainly focus on supervised learning, which is one of the machine learning problems. The cost function (loss function) of a supervised deep network has the form $\mathcal{L}((\mathbf{w}, \mathbf{b}), \mathbf{x}, \mathbf{y})$, which measures how well the true label fits for the input point $\mathbf{x}$. The cost function is defined differently depending on the task. For example, cross-entropy is usually used in classification tasks, and Root Mean Squared Error (RMSE) is often used in regression problems.

A deep network differs from other machine learning models in that the input $\mathbf{x}$ corresponding to a feature is usually not the result of hand-craft feature engineering. Often, raw data is fed into the network, and the representation (activation vector) $\mathbf{z}$ of each layer is formed as task-relevant information. Therefore, forming informative $\mathbf{z}$ well is closely linked to the high performance of deep learning tasks. The characteristics of representation $\mathbf{z}$ can be manipulated via representation shaping to improve task performance. Also, nonlinear activation functions play an important role because it helps a deep network learns a complicated relationship between the input and output. An activation function is applied to pre-activations element-wisely so affects the shape of representations. Among a range of activation functions, Rectified Linear Unit (ReLU) often provides better performance than others such as sigmoid and tanh activation functions. In this work, we only consider ReLU activation function.

2.2. Generalization

Machine learning has been defined in a variety of ways depending on fields that use machine learning. One of the widely used definitions is Tom M. Mitchell's (Michalski, Carbonell, and Mitchell 2013), "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E." In practice, experience E means that a computer program learns regular patterns in training data and then generalizes them for unseen data. Therefore, there can exist multiple models that perfectly fit the training data, but only some of them generalize well. Among the models, finding one that generalizes better than others is a key in machine learning.

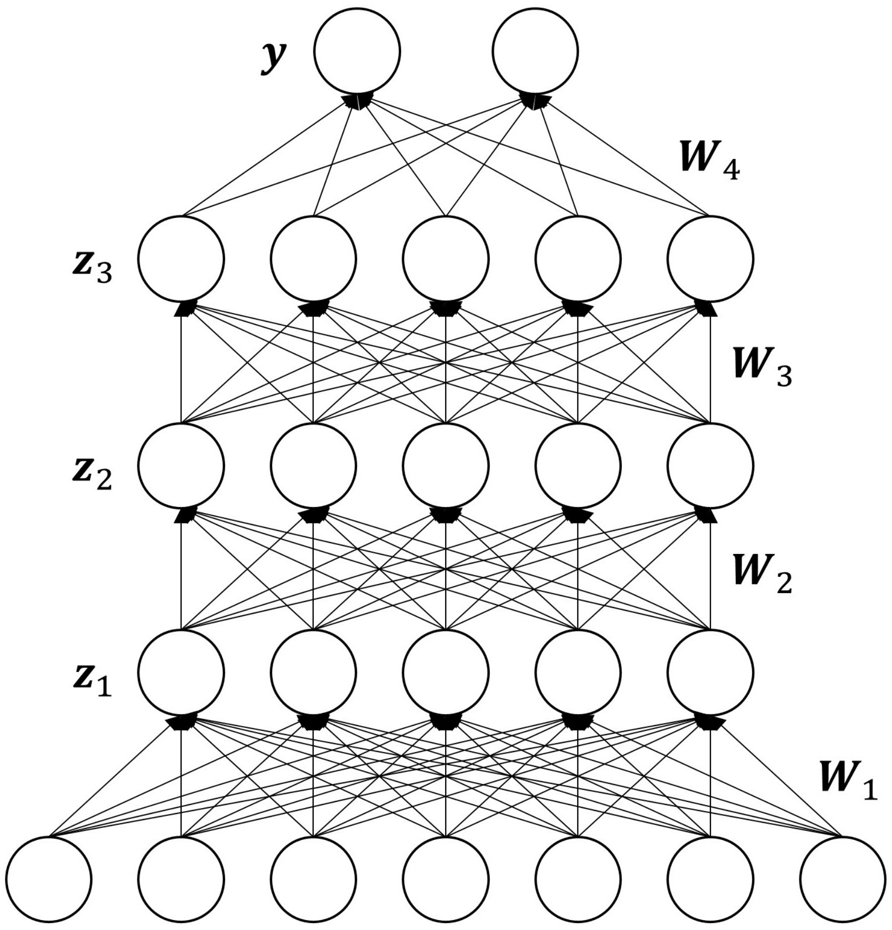


Figure 2.1. An illustration of a multilayer perceptron. This network is for solving a binary classification task. The network includes one input layer with seven scalar values and four hidden layers containing five units each, and the last hidden layer indicates the output layer. Note that typical deep networks are much larger than this example depending on input data types and network capacity required for a task. Each edge is a weight parameter describing the relationship between two units of two consecutive layers. All the edges between two layers form a weight matrix $\mathbf{W}$ that maps from a layer to the next layer. The bias (intercepter) parameters associated with each layer are omitted.

However, in deep learning, there often exist a large number of networks that generalize comparably well. This is because there are a lot of local minima in the weight space. An intuitive example is weight space symmetry. Another example is the scaling of incoming weight and rescaling of its outgoing weight. We will define Identical Output Network (ION) to generalize these cases to linear layers and show that ReLU networks' IONs are somewhat limited in Chapter 4. The number of weight parameters of a deep network is usually far more significant than the number of data samples used for training the network. However, the network with excessive capacity still generalizes well. That is, the network rarely leads to overfitting, which is against statistical learning theory. To better understand the generalization of deep learning, we first briefly review the relationship between model capacity and generalization. Then, we explain recent research on generalization in deep learning.

2.2.1. Capacity, Overfitting, and Generalization

In machine learning, a model is trained with some given training data by minimizing empirical risk. This model training can be considered as an optimization process to find model parameters for fitting the given training data. Therefore, training error does not tell us how well it will work for new and previously unobserved data. The goal of machine learning is to minimize generalization error. The expected value of the error on unseen input data is called *generalization error*. We often estimate the generalization error by measuring the difference between the training error and test error because we almost always do not know the true joint probability distribution of input and output data. Test data are assumed to be selected from the same distribution as that of training data with i.i.d assumption but are gathered separately from the training data. When the test data is not available, we estimate test error by measuring validation error.

Since both training and test data are assumed to be collected from the same distribution, their expected errors are the same as well for some fixed model parameters. Apparently, model parameters are not fixed before training time; instead, model parameters are learned using training data during training time. As mentioned above, since training is an optimization process using given training data, several models that perfectly fit the training data or that have small training error exist. For example, multiple machine learning algorithms have the same zero training error, and even the same algorithms with different hyperparameters can have the same zero error. This can be possible if the algorithms have enough capacity to fit all the training data and noise perfectly, which means that they may not perform well on the test data as shown in Figure 2.2. This challenge is called *overfitting*, which occurs when generalization error is too large. (On the contrary, underfitting occurs when a model is too small to fit the training data so does not obtain a sufficiently low training error.) Now, the question is how to choose a model that has a small generalization error. We explain this in the next section.

2.2.2. Generalization in Deep Learning

As mentioned in the previous subsection, classical learning theory relates generalization ability with the capacity of hypotheses (Vapnik 1999). Deep hypothesis spaces are more advantageous over shallow hypothesis spaces so can have better ability to fit a broad range of functions and generalizability (Pascanu, Montufar, and Bengio 2014; Montufar et al. 2014; Telgarsky 2016). However, it seems that the theory on model capacity control does not apply to deep learning. Real-life deep learning generalizes well even if it has a large number of parameters. Therefore, the capacity of a deep network can be considered to be larger than other machine learning algorithms. Considering that the number of samples is usually much smaller than the number of parameters, it is astonishing that overfitting rarely occurs. Long-term belief on this mystery is that stochastic gradient descent implicitly reduces the excess capacity of the network. Another is that explicit regularization methods such as weight decay and dropout help reduce the capacity. However, recently, Zhang et al. (2016) empirically showed that the capacity of deep learning models is large enough to memorize the entire training

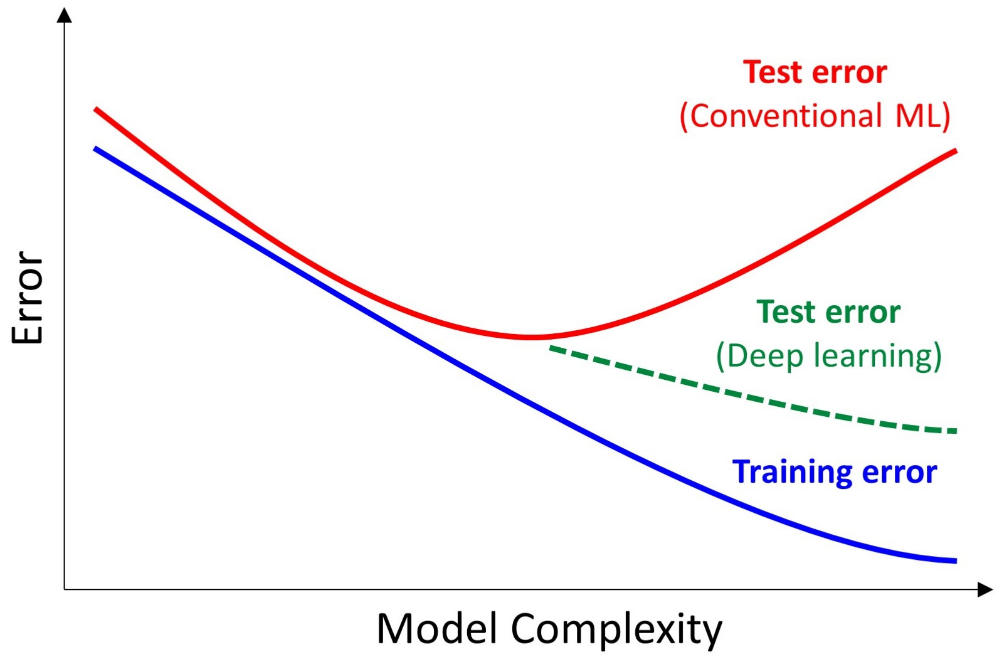


Figure 2.2. The relationship between model complexity and error. The training error monotonically decreases as model complexity increases (blue line). The test error decreases like the training error does but begins to increase as model complexity becomes too high (red line). It is therefore important to find the lowest possible test error by cross-validation techniques. This may not be true for real-life deep learning. Oftentimes, the test error of deep networks monotonically reduces while with increasing the number of weight parameters, which is considered as model complexity in deep networks (green line) .

samples, and performance improvement using explicit regularization methods such as L2 weight decay and dropout seems a tuning effect. Arpit et al. (2017) further empirically investigated memorization and generalization in deep networks.

To better understand the generalization of deep learning, researchers try to find a correlation between a common property of high-performing networks and generalization performance. Neyshabur et al. (2017) examined some of the complexity measures such as norm, robustness, and sharpness for explaining generalization of deep learning. They concluded that a single measure cannot fully explain the generalization of a deep network. In this work, we seek properties that high-performing networks share. However, we more focus on the properties of deep representations than those of weight parameters. We investigate the causal relationship between the statistical characteristics of deep representations and performance in great detail in Chapter 4.

2.3. Regularization

In the previous section, we briefly introduced the concept of capacity, overfitting, and generalization. In this section, we elucidate regularization that is a strategy to prevent overfitting and reduce generalization error. We explain regularization methods for deep learning and also discuss the roles of deep network regularization.

2.3.1. Capacity Control and Regularization

Among models with comparable training performance, some of them may overfit to the training data depending on their capacity, which leads to poor generalization performance. As presented in the previous section, we want to build a model that performs well on unseen data such that better generalization is achieved. Generally, overfitting can be avoided by controlling model capacity, which can be done in many ways.

A method to prevent overfitting can be categorized into two strategies. First, the function class can be restricted to minimize empirical risk. It is reasonable to limit the class of functions to consider because there are too many models that perfectly fit training data. In statistical learning theory, the amount of overfitting can be lessened by using more training data or applying less complex algorithms. Therefore, simple models tend to be preferred for preventing overfitting. More importantly, choosing proper hypothesis space which closely matches the underlying complexity of the learning task is required. For example, the degree of polynomials can be controlled to fit true joint probability distribution in linear regression appropriately. Another example is to choose a proper number of hidden layers and units to capture true underlying explanatory factors for a learning task in deep learning.

Capacity can be controlled by explicitly penalizing the model as well. In machine learning, extra information is added to a learning task to choose a particular model that fits well, which is closely related to the next method. The second method is to alter the learning criterion (cost or objective function), which is called *regularization* in general. Adding a regularization term to control model complexity is very popular in both traditional machine learning and deep learning. Regularization is related to the no free lunch theorem which tells us that no single machine learning algorithm universally performs well on all tasks (Wolpert and Macready 1997). Therefore, even though the algorithm used is the same (the function class is restricted), one with different hyperparameters or capacity can be preferred. Among the two strategies for preventing overfitting, we focus on regularization in this work.

For regularization, a model with large capacity can be chosen first, and a regularizer can then be applied to the model to reduce its capacity. A typical way to regularize the model is to add penalty terms such as Ridge regression (Hoerl and Kennard 1970) and the Lasso (Tibshirani 1996). More specifically, Ridge regression adds the L2-norm loss to the objective function (J). Then, the modified objective function ($\widetilde{J}$) can be written as follows.

$$\widetilde{J}(\mathbf{w}) = J(\mathbf{w}) + \lambda \|\mathbf{w}\|^2 \,, \tag{2.1}$$

where $\boldsymbol{w}$ is a parameter vector, and λ is a regularization parameter that balances between the two terms. By tuning λ, one can choose the proper trade-off between fitting and complexity. When λ is chosen to be large, the model capacity becomes small. As a result, training performance may degrade a little, but generalization performance can improve greatly.

Of course, L2 norm regularization is one choice among a range of regularization methods. One definition of regularization by Goodfellow, Bengio, and Courville (2016) is “any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error.” Their definition is not limited to the capacity control of the machine learning algorithm. Instead, they focus on prior knowledge or preference for learning models and generalization performance improvement by expressing prior knowledge or preference. By this definition, even multitask learning and adversarial training can be considered as regularization methods.

In this work, we mainly focus on how to regularize deep networks by adding penalty terms on representations. This approach is different from conventional penalty methods on weight parameters; this approach indirectly imposes a complicated penalty on weight parameters via representation regularization. We will review details of deep network regularization using representations in the next section. Now, we take a closer look at regularization for deep learning in the next subsection.

2.3.2. Regularization for Deep Learning

When deep learning methods are used, we hope that our deep learning model closely approximates the true target function. However, it is easily seen that we usually do not know the true target function or true data-generating process because the domains of deep learning applications such as image, audio, and language are too complicated. We, therefore, often choose an overly complex model and regularize it to have a proper capacity (neither underfitting nor overfitting happens).

From the viewpoint of standard regularization, the complexity of deep networks can be controlled in various ways in order to avoid overfitting. As in the example provided in the previous subsection, a very simple and widely used way is adding the L2 loss of deep network weights to the cost function and choosing proper coefficient λ to control the complexity of the network. By doing this, validation performance of the deep network is improved, and it has been believed that the improvement is a result of controlling the complexity of the network. However, Zhang et al. (2016) showed that the regularized network performs better than the original network for real data but still has enough capacity to fit random label data perfectly, meaning zero training error. Therefore, we cannot conclude that the capacity of the network is reduced, and less complexity provides a performance improvement. Furthermore, the performance improvement by the weight decay was reasonably small, so they argued that explicit regularization methods such as weight decay, data augmentation, and dropout are more like tuning methods unlike regularization methods of machine learning.

The examples of deep learning regularization are quite broad. Deep learning regularization includes standard regularization such as L1 and L2 penalty loss on parameters called weight decay, and parameter sharing and tying are often implemented as a deep network architecture like CNNs. Early stopping and data augmentation are popular techniques to improve generalization performance. As in the definition shown in the previous subsection, even noise injection and an ensemble of networks can be a regularization method if generalization performance is improved by the method. In this work, among all possible regularization methods, we focus on some popular ones in practice. Some related works are as follows.

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 networks as well. Wen et al. (2016) extended L1 regularizers by using group lasso to regularize the structures of a deep network (i.e., filters, channels, filter shapes, and layer depth). Srivastava et al. (2014) devised dropout that randomly applies activation masking over the units. By doing this, co-adaptation among hidden units is reduced, and overfitting can be lessened. 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. (2015) 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 an 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 deep network regularization techniques, but we still do not fully understand how they really affect performance. Regularization methods using deep network representations are reviewed in the next section.

2.4. Representation

It is well known that the success of deep learning stems from the ability of deep learning algorithms to construct task-relevant data representations automatically. To further benefit from the ability, many researchers have actively studied representation learning which is one of the central topics in the field of machine learning. In this section, we briefly introduce representation learning, representation shaping, and then review studies related to representation shaping in detail.

2.4.1. Representation Learning

Extracting useful information from data is a key to building high performing supervised classifiers and other predictors. In the same way, discovering underlying explanatory factors in data is important for building generative models. In traditional machine learning, these are usually done by hand using human domain knowledge, which is called feature engineering. Feature engineering is often beneficial to achieve high performance but often quite time and effort consuming. More importantly, it has become harder to capture underlying explanatory factors with hand-craft feature engineering in a variety of complex sensory data used in artificial intelligence applications such as images, audio signals, and languages.

On the contrary, representation learning allows a system to capture useful information from the data automatically thus removes the need for explicit manual feature engineering. Representation learning methods learn data representation (features) and utilize them to perform a machine learning task. However, the objective of representation learning might be less clear than that of supervised learning. A few conventional examples are as follows. Principle Component Analysis (PCA) and Independent Component Analysis (ICA) are popular techniques that capture low-dimensional features and independent factors in data. A variety of clustering methods including k-mean clustering can discover the underlying structure of data so can be considered as representation learning methods.

Deep learning methods such as multilayer perceptrons, autoencoders, and other forms of neural networks can be collectively considered as representation learning methods. Deep learning methods can automatically capture task-relevant information in the data by forming hidden layer representation so significantly reduce the effort needed for feature engineering. Distributed, disentangled, and deep nature of deep network representations are known to be a primary reason why deep learning methods form a good representation that is one making learning task easier. For example, CNNs capture from low to high-level concepts of images by forming multiple convolutional layers. An autoencoder discovers underlying disentangled factors in the data by multiple nonlinear transformations of the input. Generative Adversarial Networks (GANs) can even generate new data samples by controlling independent factors.

In practice, it is important to learn a good representation by applying new architectures, optimization techniques, and regularization methods. One possible way is to shape representation by using penalty regularization on representations. Learned representations can become helpful for the task by manipulating their statistical characteristics such as correlation and representational sparsity. In the next subsection, we introduce a few representation shaping strategies and class-wise regularization methods to improve task performance and interpretability of deep representations.

2.4.2. Representation Shaping

In statistics, the shape of a distribution can be described by either descriptive language like the descriptive term ‘bell-shaped’ or quantitative measures such as modality and kurtosis (Mood 1950). In deep learning, our interest is how activations of a hidden unit are encoded. In other words, the distribution of hidden unit activations is considered important because it may tell us how a deep network efficiently encodes and decodes input data to capture task-relevant information. The extension to high dimensional distribution (hidden layer activations) is, of course, more important but is often difficult to understand intuitively and visually.

In statistics, describing the shape of the distribution itself is often considered more important than changing its shape. However, in deep learning, it would be interesting to alter the shape of the distribution such that statistical characteristics of representations are manipulated and uncover the relationship between its shape and task performance. To the best of our knowledge, there is no formal definition of altering the shape of distribution in statistics (it seems that a definition is not needed). In deep learning, generally speaking, representation shaping can be defined as forcing constraints on representations to have some desired properties using prior knowledge on the data and task. For example, we can shape representations to have multi-modal features or a high kurtosis value. Representation shaping can be divided into two approaches. First, a particular operation to shape representation such as dropout and batch normalization can be added to a layer or unit in deep networks. The second approach is to penalize hidden activations by adding a penalty term like the L1 penalty on representation to the cost function to encourage sparser representations. In this work, we mainly focus on the second approach. In this subsection, we introduce a few representation shaping strategies and class-wise shaping methods as well.

Penalty Regularization on Representations

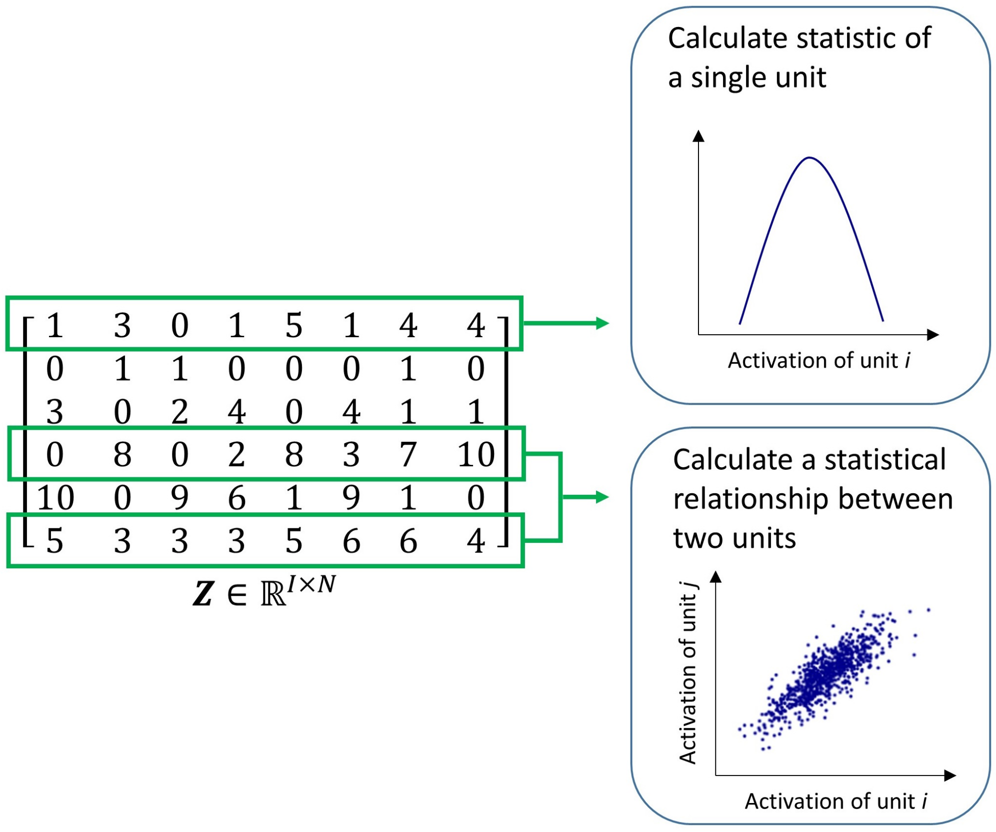
Like placing a penalty on weights, a penalty can be placed on representations, the activations of hidden units in a deep network. For example, L1 penalty regularization on activations is widely used to enforce representational sparsity. Of course, L1 penalty on representations is only one choice of representation regularizers, and a representation penalty term can be in any form that reflects a preference for a model. Figure 2.3 illustrates an example of a hidden layer activation matrix in a deep network. Two examples of statistical properties that can be shaped are as follows. We first consider the distribution of a single hidden unit. Activation distribution can be altered to increase kurtosis for the stability of the activation distribution just like the purpose of batch normalization. This regularization, of course, can hurt classification performance when a loss weight is too large. When considering two hidden units' distributions together, cross-covariance of activations of two units can be regularized for linearly independent representations. This shaping aims to discover underlying independent factors, so may be beneficial for density estimation tasks.

The objective function ($\widetilde{J}$) with a representation regularization term can be written as follows.

$$\widetilde{J}(\mathbf{w}) = J(\mathbf{w}) + \lambda\Omega(\mathbf{z}), \tag{2.2}$$

where $\mathbf{z}$ is an activation vector, λ is a regularization parameter that balances between two terms, and Ω is a representation penalty function. This equation has the same form as that of weight regularization except that it has $\mathbf{z}$ instead of $\mathbf{w}$ in the penalty term. Therefore, weight parameters are indirectly penalized by the representation regularization term.

Some of the existing regularization methods explicitly shape representations by adopting a penalty regularization term. DeCov by Cogswell et al. (2016) is a penalty regularizer that minimizes the off-diagonals of a layer's representation covariance matrix. DeCov reduces the co-adaptation of a layer's units by encouraging the units to



Calculate statistic of a single unit
Activation of unit *i*
Calculate a statistical relationship between two units
Activation of unit *j*
Activation of unit *i*

$$\mathbf{Z} \in \mathbb{R}^{I \times N} = \begin{bmatrix} 1 & 3 & 0 & 1 & 5 & 1 & 4 & 4 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 3 & 0 & 2 & 4 & 0 & 4 & 1 & 1 \\ 0 & 8 & 0 & 2 & 8 & 3 & 7 & 10 \\ 10 & 0 & 9 & 6 & 1 & 9 & 1 & 0 \\ 5 & 3 & 3 & 3 & 5 & 6 & 6 & 4 \end{bmatrix}$$

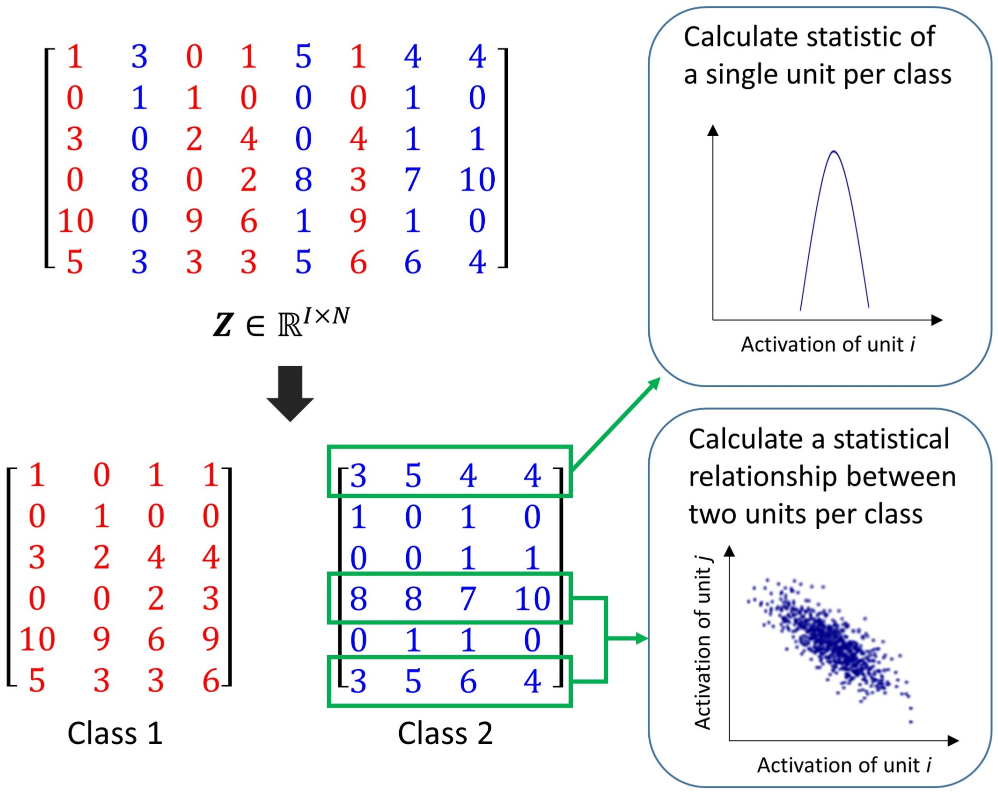
Figure 2.3. An example of a hidden layer activation matrix (representation) in a deep network. The network has six units (I) in the hidden layer and a mini-batch size of eight (N) in this example. A statistic of a single unit and a statistical relationship between two units can be computed by using the hidden layer activation matrix. Calculated statistics of representations can be added to the objective function of deep network such that representations can be regularized to manipulate the statistics.

be decorrelated. In this work, it is called CR (Covariance Regularizer) for consistency. A recent work by 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 both supervised and unsupervised tasks, our work utilizes a much simpler and computationally efficient method of directly using class labels during training to avoid k-means like clustering. Littwin and Wolf (2018) proposed a new regularization term called Variance Consistency Loss (VCL) that is used for stabilizing the variance of the activations so that the variance of each mini-batch is close to each other. As a result of applying their VCL, learned representations have several different modes in a single hidden unit, and performance is often improved. However, they have not shown that the modes correspond to each class, so why their regularization is helpful for improving performance seems unclear.

Representation Regularization Using Class Information

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 for each class such that different classes can be better distinguished. Surprisingly, true class information has not been commonly used directly for regularization methods. Traditionally, 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.

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 a 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. Besides the pair-wise computation, two optimizers are used



$\begin{bmatrix} 1 & 3 & 0 & 1 & 5 & 1 & 4 & 4 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 3 & 0 & 2 & 4 & 0 & 4 & 1 & 1 \\ 0 & 8 & 0 & 2 & 8 & 3 & 7 & 10 \\ 10 & 0 & 9 & 6 & 1 & 9 & 1 & 0 \\ 5 & 3 & 3 & 3 & 5 & 6 & 6 & 4 \end{bmatrix}$
$\boldsymbol{Z} \in \mathbb{R}^{I \times N}$
$\begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 3 & 2 & 4 & 4 \\ 0 & 0 & 2 & 3 \\ 10 & 9 & 6 & 9 \\ 5 & 3 & 3 & 6 \end{bmatrix}$
Class 1
$\begin{bmatrix} 3 & 5 & 4 & 4 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 8 & 8 & 7 & 10 \\ 0 & 1 & 1 & 0 \\ 3 & 5 & 6 & 4 \end{bmatrix}$
Class 2
Calculate statistic of a single unit per class
Activation of unit *i*
Calculate a statistical relationship between two units per class
Activation of unit *j*
Activation of unit *i*

Figure 2.4. An example of two hidden layer activation matrices separated by a class label in a deep network. An activation matrix $\mathbf{Z}$ is the same as the matrix in Figure 2.3. Each color refers to a different class. In this example, there are two classes, and $\mathbf{Z}$ is divided into two activation matrices. As mentioned in the caption of Figure 2.3, a statistic of a single unit and a statistical relationship between two units can be calculated using $\mathbf{Z}$, but they are computed class-wise in this example. Calculated statistics of representations per class can be added to the objective function of the deep network so that representations can be regularized to manipulate the class-wise statistics.

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. In this work, like the strategies listed above, our cw-VR and cw-CR use class information to control the statistical characteristics of representations. We first consider the distribution of a single hidden unit where class information is used. Variances of activations per class can be regularized (cw-VR). This regularization aims to encourage more separable representation per class, which may lead to better classification performance. Class-wise regularization can be applied to a pair of hidden units as well to decorrelate representations (cw-CR). Even though our methods are similar to methods above in terms of using class information, our methods are different in some points. Our methods are simple because they use only one optimizer and are computationally efficient because they require only neuron-wise calculations while not requiring pair-wise computations.

Chapter 3. Representation Regularizer Design with Class Information

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 representation heavily.

For representation shaping (regularization), some of the promising methods for performance and interpretability include (Cogswell et al. 2016; Glorot, Bordes, and Bengio 2011; Liao et al. 2016). Glorot, Bordes, and Bengio (2011) consider increasing representational sparsity, Cogswell et al. (2016) focus on reducing covariance among hidden units, and Liao et al. (2016) force 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; Yang et al. 2018; Belharbi et al. 2017) do utilize class information, but their approaches are based on hidden layer activation vectors. Among them, the method by Belharbi et al. (2017) is computationally expensive 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

3.1. In Figure 3.1(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 3.1(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.

3.1. 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.

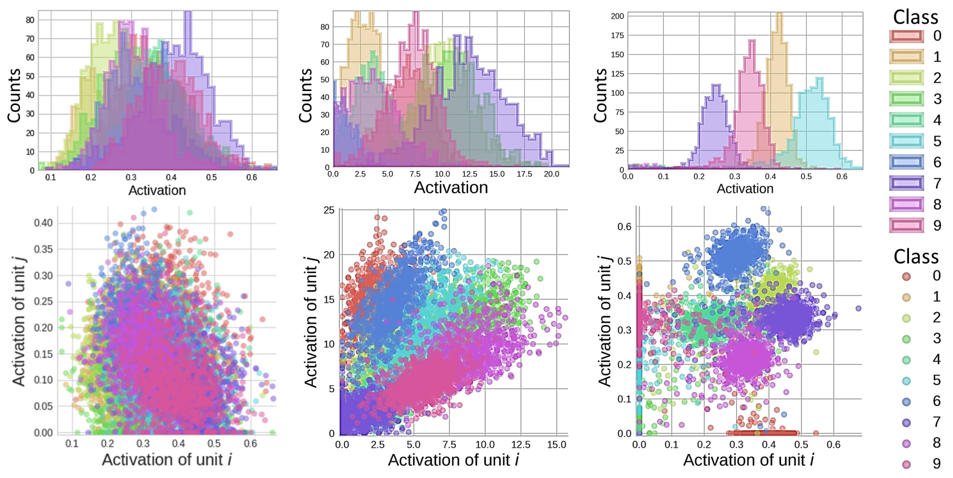
3.1.1. Basic Statistics of Representations

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.

$$\mu_i = \frac{1}{N}\sum_n z_{i,n} \tag{3.1}$$

$$c_{i,j} = \frac{1}{N}\sum_n (z_{i,n} - \mu_i)(z_{j,n} - \mu_j) \tag{3.2}$$



(a) Baseline (0th) (b) Baseline (50th) (c) cw-CR (50th)

Figure 3.1. 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.

Table 3.1. 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
$\Omega_{CR} = \frac{2}{I(I-1)} \sum_{i \neq j} (c_{i,j})^2$	$\frac{\partial \Omega_{CR}}{\partial z_{i,n}} = \frac{4}{NI(I-1)} \sum_{j \neq i} c_{i,j}(z_{j,n} - \mu_j)$
$\Omega_{cw\text{-}CR} = \frac{2}{KI(I-1)} \sum_k \sum_{i \neq j} (c^k_{i,j})^2$	$\frac{\partial \Omega_{cw\text{-}CR}}{\partial z_{i,n}} = \frac{4}{KI(I-1)\lvert S_k \rvert} \sum_{j \neq i} c^k_{i,j}(z_{j,n} - \mu^k_j), n \in S_k$
$\Omega_{VR} = \frac{1}{I} \sum_i v_i$	$\frac{\partial \Omega_{VR}}{\partial z_{i,n}} = \frac{2}{NI}(z_{i,n} - \mu_i)$
$\Omega_{cw\text{-}VR} = \frac{1}{KI} \sum_k \sum_i v^k_i$	$\frac{\partial \Omega_{cw\text{-}VR}}{\partial z_{i,n}} = \frac{2}{KI\lvert S_k \rvert}(z_{i,n} - \mu^k_i), n \in S_k$

3.1.2. cw-CR

Here, $z_{i,n}$ is the activation of unit i for nth sample in the mini-batch. From equation (3.2), variance of i unit can be written as the following.

$$v_i = c_{i,i} \tag{3.3}$$

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.

$$\mu^k_i = \frac{1}{\lvert S_k \rvert} \sum_{n \in S_k} z_{i,n} \tag{3.4}$$

$$c^k_{i,j} = \frac{1}{\lvert S_k \rvert} \sum_{n \in S_k} (z_{i,n} - \mu^k_i)(z_{j,n} - \mu^k_j) \tag{3.5}$$

$$v^k_i = c^k_{i,i} \tag{3.6}$$

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

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

$$\widetilde{J} = J + \lambda \Omega_{cw\text{-}CR}(\mathbf{z}), \tag{3.7}$$

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.

3.1.3. 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).

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 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 useful for performance enhancement. We provide a possible explanation in the Experiments section.

3.1.4. 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 3.1. We investigate more details of the equations in the following.

Interpretation of the gradients

Among the gradient equations shown in Table 3.1, 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. (2016), 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. However, 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 3.2 later.

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

3.2. 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 evaluation 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.

Table 3.2. 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	**21.44 ± 0.88**	25.01 ± 0.41
cw-VR	21.58 ± 0.21	**24.42 ± 0.31**

Table 3.3. 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	**41.38 ± 0.53**	**54.23 ± 1.06**	**56.75 ± 0.64**

3.2.1. 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 Chapter 5. All the regularizers were applied to the fifth layer of the 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 3.2 and three CIFAR-100 CNN scenarios are shown in Table 3.3. The rest of the scenarios including full cases of MNIST MLP can be found in Chapter 5. In the Table 3.2 and Table 3.3, 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 3.3, we initially guessed that the performance of class-wise regularizers would be sensitive to the number 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 3.6 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 deep network 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 hyper-

Table 3.4. Error performance (%) for ResNet-32/110 (CIFAR-10). We perform ResNet-110 experiment five times and report 'best (mean±std)' 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)

Table 3.5. Mean squared error of deep autoencoder.

Regularizer	Mean Squared Error
Baseline	$1.44 \times 10^{-2} \pm 3.36 \times 10^{-4}$
CR	$1.29 \times 10^{-2} \pm 2.44 \times 10^{-4}$
cw-CR	$1.22 \times 10^{-2} \pm 3.63 \times 10^{-4}$
VR	$1.29 \times 10^{-2} \pm 5.16 \times 10^{-4}$
cw-VR	$\mathbf{1.19 \times 10^{-2} \pm 2.48 \times 10^{-4}}$

parameters. The results are shown in Table 3.4. All four turned out to have positive effects where cw-VR showed the best performance again.

3.2.2. Image Reconstruction Task

To test a completely different type of task, we examined an image reconstruction task where autoencoders 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 Table 3.5. As in the classification tasks, class-wise regularizers performed better than their all-class counterparts.

3.3. Analysis of Representation Characteristics

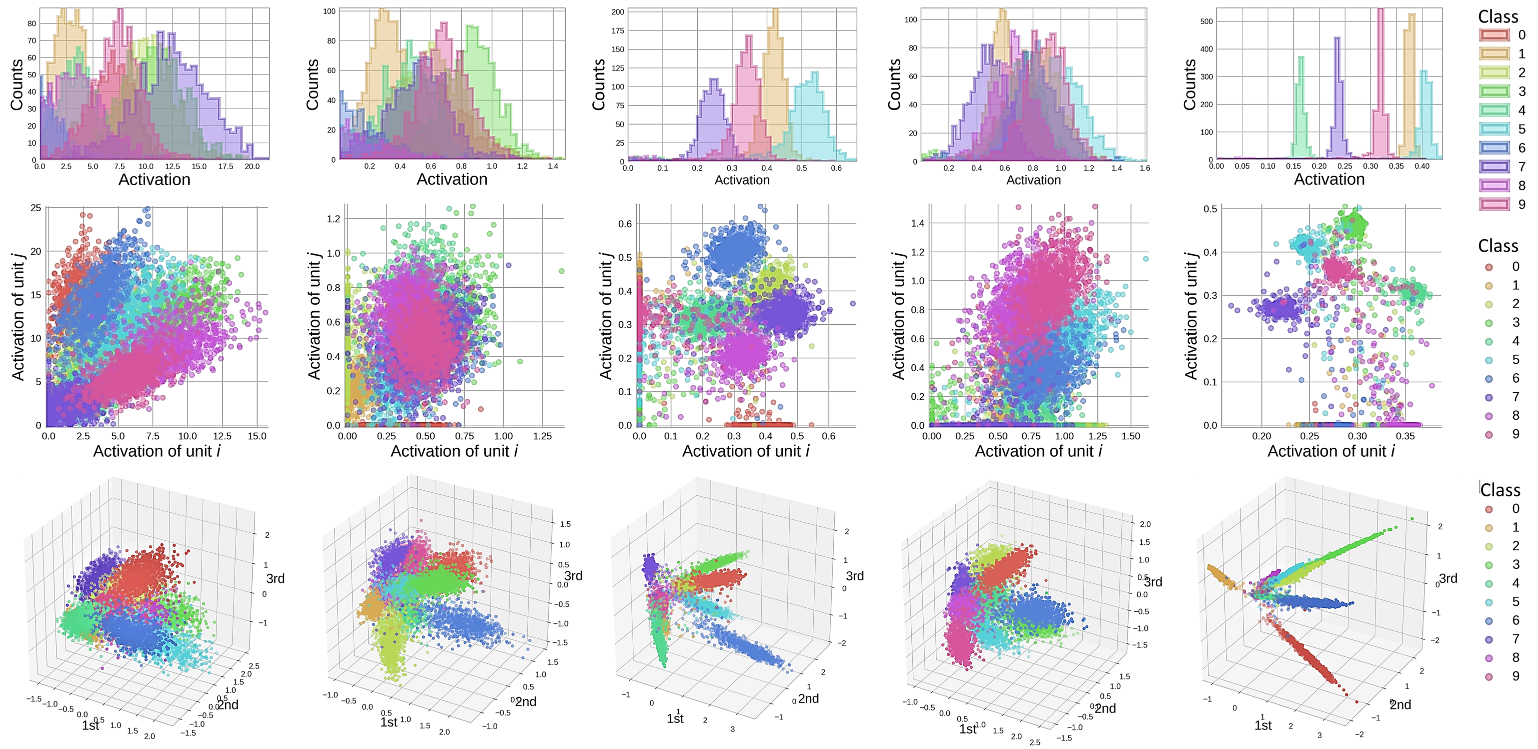
In this section, we investigate representation characteristics when the regularizers are applied.

3.3.1. Visualization

In Figure 3.2, the 50^{th} epoch plots of Figure 3.1 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, the 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 Appendix A. Overall, cw-VR shows the most distinct shapes compared to the baseline.

3.3.2. Quantitative Analysis

For the same MNIST task that was used to plot Figure 3.1 and Figure 3.2, the quantitative values of representation characteristics were evaluated and the results are shown in Table 3.6. Each is calculated using only positive activations and is the average of representation statistics. For example, ACTIVATION_AMPLITUDE is the mean of positive activations in 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



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

Figure 3.2. Visualization of the learned representations for MNIST. The plots in top and middle rows were generated in the same way as in the Figure 3.1. The plots in the bottom row show the top three principle components of the representations.

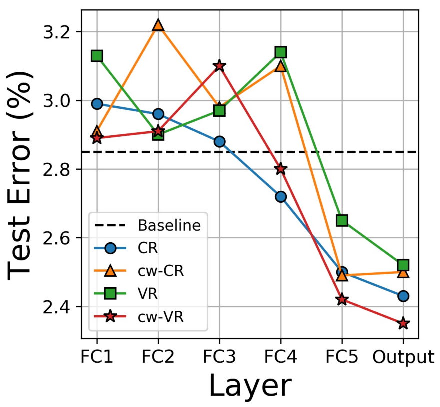
Table 3.6. Quantitative evaluations of representation characteristics.

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	**2.42 ± 0.06**	*0.63*	0.02	0.36	0.25	0.05	**0.74**

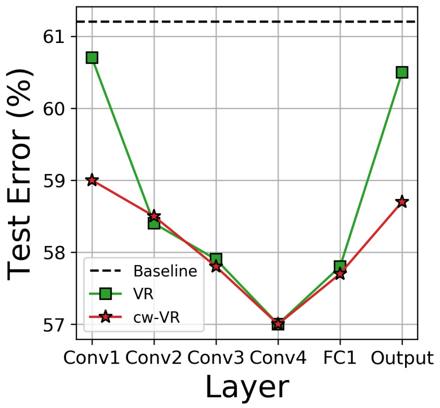
an example where not only cw-CR but also other representation regularizers end up reducing CW_CORRELATION because the regularizers' gradient equations are related. For VR, the variance should be reduced. In the seventh column (VARIANCE), the variance of VR is indeed much smaller than that of the baseline, but again other representation regularizers have even smaller values because their activation squashing is more severe than that of VR. For cw-VR, a 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.)

3.4. 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 3.3. In Figure 3.3 (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 3.3 (b) and 3.4, 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



(a) MNIST



(b) CIFAR-100

Figure 3.3. Layer dependency of representation regularizers on MNIST MLP and CIFAR-10 CNN models. 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.

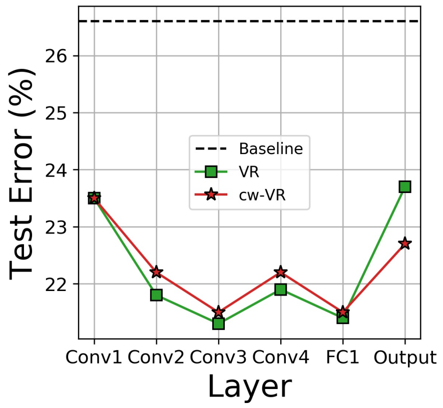


Figure 3.4. Layer dependency of representation regularizers on CIFAR-10 CNN model. The x-axis indicates layers where regularizers are applied. CR and cw-CR are excluded because of the high computational burden of applying them to the convolutional layers.

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.

Chapter 4. Representation Characteristics and Their Relationship with Performance

A learned representation can significantly affect the performance of deep networks, and the representation's distributed and deep natures are the essential elements for the success of deep learning (Bengio, Courville, and Vincent 2013). As a consequence, deep networks have a greater *expressiveness* compared to the other machine learning algorithms (Hinton 1986) or shallow networks (Montufar et al. 2014; Telgarsky 2015; Eldan and Shamir 2016; Raghu et al. 2017). Besides the distributed and deep natures that have been intensively studied, a hidden layer's representation characteristics are considered to be important as well. Nonetheless, a relatively smaller number of studies have been completed on the topic, and the goal of this work is to understand the representation characteristics better. Therefore, *the meaning of representation in this work is restricted to the activation vector of a single hidden layer* and a *unit* refers to a neuron of the hidden layer.

A few previous studies considered manipulating *statistical characteristics* of representations such as reducing covariance among hidden units (Cogswell et al. 2016; Xiong et al. 2016), encouraging representational sparsity (Glorot, Bordes, and Bengio 2011), or forcing parsimonious representations via clustering (Liao et al. 2016). In some of the similar works, a popular argument has been that the representation regularization reduces the generalization error via altering a representation characteristic. This argument, however, has not been rigorously studied. Another popular argument has been the reduction of effective capacity via regularization. This argument has been recently examined by Zhang et al. (2016) where they empirically show that explicit regularization methods like L2 weight decay and dropout do not sufficiently limit the effective capacity of deep networks.

Since a novel information-theoretic analysis method was proposed for deep learning

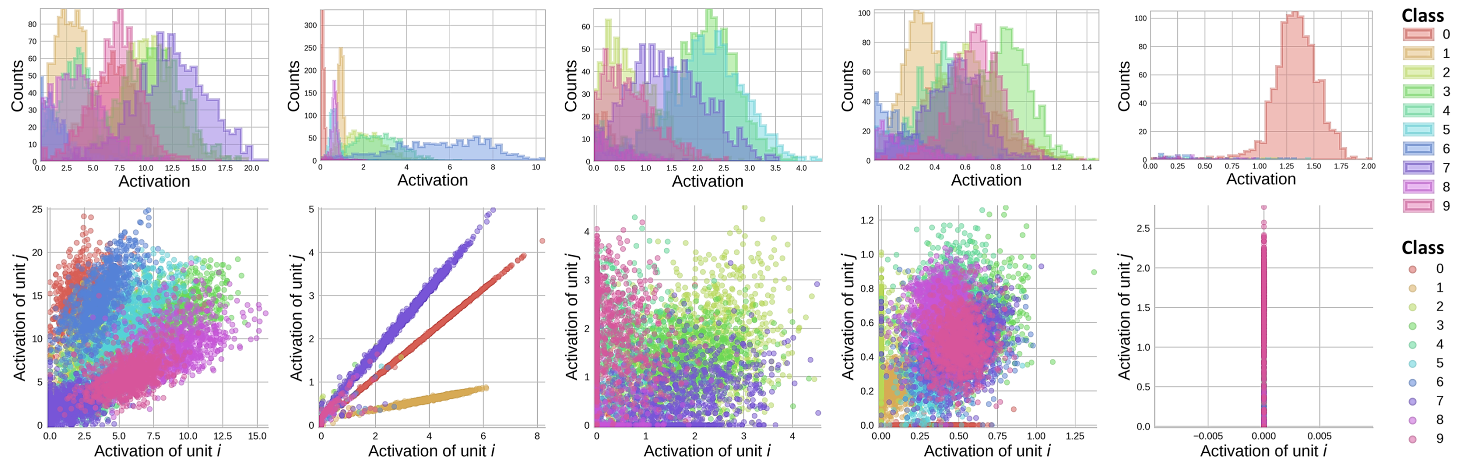
Table 4.1. Representation characteristics.

Characteristic	Symbol	Expression
Activation amplitude	$\bar{\lvert z \rvert}$	$\mathbb{E}_i[\lvert\, \mathbf{z}_{l,i} \,\rvert]$
Covariance	$\bar{c}$	$\mathbb{E}_{i \neq j}[c_{i,j}]$, where $c_{i,j} \triangleq \{\mathbf{C}_l\}_{i,j} = \mathbb{E}[(\mathbf{z}_{l,i} - \mu_{z_{l,i}})(\mathbf{z}_{l,j} - \mu_{z_{l,i}})]$
Correlation	$\bar{\rho}$	$\mathbb{E}_{i \neq j}[\rho_{i,j}]$, where $\rho_{i,j} \triangleq \{\mathbf{C}_l\}_{i,j} / \sigma_{z_{l,i}} \sigma_{z_{l,j}} = \mathbb{E}[(\mathbf{z}_{l,i} - \mu_{z_{l,i}})(\mathbf{z}_{l,j} - \mu_{z_{l,i}})] / \sigma_{z_{l,i}} \sigma_{z_{l,j}}$
Sparsity	P_s	$\mathbb{E}_{i,n}[\mathbb{1}(z^n_{l,i})]$, where $\mathbb{1}$ is an indicator function whose output is 1 only when $z^n_{l,i} = 0$
Dead unit	P_d	$\mathbb{E}_i[\mathbb{1}(z_{l,i})]$, where $\mathbb{1}$ is an indicator function whose output is 1 only when $z^n_{l,i} = 0$ for all $n = 1, .., N$
Rank	r	$rank(\mathbf{C}_l)$; numerical evaluations are approximated as the stable rank $\lVert \mathbf{C}_l \rVert_F^2 / \lVert \mathbf{C}_l \rVert_2^2$
Mutual information	$I_{\mathbf{x}}$	$I(\mathbf{z}_l; \mathbf{x})$
Mutual information	$I_{\mathbf{y}}$	$I(\mathbf{z}_l; \mathbf{y})$

(Shwartz-Ziv and Tishby 2017; Tishby and Zaslavsky 2015), *information-theoretic* characteristics of representation have become an important research topic. In their work, mutual information $I(\mathbf{z}_l; \mathbf{x})$ and $I(\mathbf{z}_l; \mathbf{y})$ are used to address the learning dynamics and generalization of deep learning, where $\mathbf{z}_l$ is the hidden layer l's representation, $\mathbf{x}$ is the input, and $\mathbf{y}$ is the output. It is further discussed that a good representation is the one that contains a minimal amount of information from the input while containing a sufficient amount of information from the output. In Achille and Soatto (2018a), the Information Bottleneck Lagrangian (Tishby, Pereira, and Bialek 1999) is decomposed into the sum of a cross-entropy term and a regularization term. The regularization term turns out to be $I(\mathbf{z}_l; \mathbf{x})$ that needs to be minimized. Some of the recent works will be additionally addressed in Section 4.5.

4.1. Representation Characteristics

In this section, we briefly address the most popular statistical characteristics and information-theoretic characteristics of representations. Consider a neural network $\mathcal{N}_{\mathcal{A}}$ whose architecture $\mathcal{A}$ is fixed and the weights for the l^{th} layer are given by $\{\mathbf{W}_l\}$ and $\{\mathbf{b}_l\}$ after training. Notation-wise, we simply write $\mathcal{N}_{\mathcal{A}} = (\mathbf{W}, \mathbf{b})$ to define a network and $\mathbf{y}$ or $\mathcal{N}_{\mathcal{A}}(\mathbf{x})$ to refer to its deterministic output for a given input $\mathbf{x}$. The



(a) Baseline (b) Dropout (c) BN (d) CR (e) L1R

Figure 4.1. Visualization of the learned representations for a 6-layer MLP trained with MNIST dataset. A single unit's activation histogram (upper plots) and two randomly chosen units' activation scatter plots (lower plots) are shown for the fifth layer's representation, where each color corresponds to a different class. The plots were generated using 10,000 test samples of MNIST dataset.**(Upper)** It can be seen that the baseline has a large class-wise variance and inter-class overlaps, and BN and CR (originally known as DeCov show similar properties. Dropout looks completely different where activation values are more spread out for the active classes. L1R (L1 Representation regularizer) typically allow only one or two classes to be activated per unit. **(Lower)** While the baseline shows modest linearity, the others show quite different representation characteristics depending on the choice of regularizer. Dropout shows an extremely high class-wise correlation, but BN shows very low correlation. CR shows almost no correlation. Since L1R increases sparsity on representation, a class is activated over at most one of the two randomly chosen units.

layer index l is omitted when the meaning is obvious. The l^{th} layer's activation vector for the given input $\mathbf{x}$ is noted as $\mathbf{z}_l(\mathbf{x})$ or simply $\mathbf{z}_l$, and the i^{th} element of $\mathbf{z}_l$ is noted as $z_{l,i}$. The mean, variance, and standard deviation of $z_{l,i}$ are defined as $\mu_{z_{l,i}}$, $v_{z_{l,i}}$, and $\sigma_{z_{l,i}}$, respectively. The covariance of $\mathbf{z}_l$ is defined as $\mathbf{C}_l$. Then, the basic representation characteristics can be summarized as in Table 4.1. Six of them are statistical characteristics, and the last two are information-theoretic characteristics.

Previous studies on statistical characteristics are often based on regularizers. Srivastava et al. (2014) address dropout for preventing co-adaptation among hidden units by randomly putting zeros in a layer's activation vector. Ioffe and Szegedy (2015) explain batch normalization (BN) that reduces internal covariate shift via normalizing activations of each unit to speed up network training. Cogswell et al. (2016) suggest DeCov that utilizes a penalty loss function to reduce activation covariance among hidden units. Choi and Rhee (2018) consider an extension to class-wise regularization and provides four representation regularizers. Glorot, Bordes, and Bengio (2011) explain L1 representation regularization, called L1R in this work, that applies L1 penalty on activations. These representation regularization methods have distinct effects on representation characteristics, and examples of the learned representations are shown in Figure 4.1.

Because the true distribution of data is not accessible, the numerical results in the following sections are evaluated using the empirical distribution of the test dataset. Then, the expectations in Table 4.1 are with respect to the empirical distribution. For instance, $\mathbf{C}_l$ is calculated as the covariance matrix of N activation vectors $\{\mathbf{z}_l^1, ..., \mathbf{z}_l^N\}$ where $\mathbf{z}_l^n$ corresponds to the activation vector for the n'th test data sample, $\mathbf{x}^n$. Rank can be calculated by examining $\mathbf{C}_l$, but often there are tiny eigenvalues that hinder a proper assessment of the rank. Therefore, we evaluate *stable rank* instead, and it will be explained further in Section 4.4. Two information-theoretic characteristics, $I(\mathbf{z}_l; \mathbf{x})$ and $I(\mathbf{z}_l; \mathbf{y})$, are estimated using upper and lower bounds (Kolchinsky and Tracey 2017; Kolchinsky, Tracey, and Wolpert 2017). Further details are provided in Section

4.5. Rectified Linear Unit (ReLU) is the only activation function that is considered in this work. When ReLU is used, ACTIVATION AMPLITUDE, COVARIANCE, and CORRELATION are calculated using only the positive activation values such that the numerical evaluations can provide meaningful insights on what is happening to the non-zero representation values.

4.2. Experimental Results of Representation Regularization

We investigate the statistical characteristics of the learned representations when different regularizers are applied. We used the same network and dataset as the ones used for generating Figure 4.1. All the regularizers were applied only to the fifth layer, and the representation characteristics were calculated using the fifth layer as well. The penalty loss functions and their description are summarized in Table 4.2, and typical evaluation results of statistical characteristics are shown in Table 4.3. We can confirm that the statistical characteristics targeted by each representation regularizer are indeed manipulated as expected **(Bold)**. In particular, Rank Regularizer (RR) and class-wise Rank Regularizer (cw-RR) designed in this work to regularize the stable rank work as expected. The two weight regularizers (L1W: L1 Weight Regularizer, L2W: L2 Weight Regularizer) have similar characteristic values as the baseline's, and this can be taken for granted because the regularizers do not directly regularize representations. A few conventional beliefs mentioned in this work are quantitatively confirmed or disproved as well. A large number of dead units is known to be harmful because they do not contribute toward improving the performance of deep networks. Our result shows even 39% of DEAD UNIT caused by L1R does not hurt the performance, which is in line with our analysis in Section 4.4. For dropout, COVARIANCE is reduced as in Cogswell et al. (2016), but CORRELATION is actually increased compared to the baseline. In fact, COVARIANCE is reduced simply because ACTIVATION AMPLITUDE is reduced as mentioned in Section 4.3, and the correlation between two active units is actually made larger by applying dropout. Therefore, it cannot be said that the relationship between

Table 4.2. Penalty loss functions of representation regularizers. All the penalty loss functions are normalized with the number of units and the number of classes when they are used in our experiments such that the value of λ can have a consistent meaning. CR and cw-CR are standardized using the number of distinct covariance combinations. Note that the normalization terms are excluded in this table. We put a superscript k to define a class-wise statistic that is calculated using only class k's samples out of a total of K labels in the mini-batch. Class-wise mean, covariance, and variance are defined in Chapter 3.

Penalty loss function	Description on regularization term
$\Omega_{CR} = \sum_{i \neq j} (c_{i,j})^2$	*Covariance* of representations calculated from all-class samples.
$\Omega_{cw\text{-}CR} = \sum_{k} \sum_{i \neq j} (c^k_{i,j})^2$	*Covariance* of representations calculated from **the same class samples**.
$\Omega_{VR} = \sum_{i} v_i$	*Variance* of representations calculated from all-class samples.
$\Omega_{cw\text{-}VR} = \sum_{k} \sum_{i} v^k_i$	*Variance* of representations calculated from **the same class samples**.
$\Omega_{L1R} = \sum_{n} \sum_{i} \lvert z^n_i \rvert$	*Absolute amplitude* of representations calculated from all-class samples.
$\Omega_{RR} = \frac{\lVert \mathbf{Z} \rVert_F^2}{\lVert \mathbf{Z} \rVert_2^2}$	*Stable rank* of representations calculated from all-class samples.
$\Omega_{cw\text{-}RR} = \sum_{k} \frac{\left\lVert \mathbf{Z}^k \right\rVert_F^2}{\left\lVert \mathbf{Z}^k \right\rVert_2^2}$	*Stable rank* of representations calculated from **the same class samples**.

a pair of neurons becomes weaker by applying dropout. This is in contrary to the 'reduction of co-adaptation' idea. Note that we have excluded the inactive neurons for the evaluations. If the inactive ones are included with their zero values, the covariance and correlation values will be different.

Table 4.3. Statistical characteristics of learned representations.

Regularizer	Test Error (%)	ACTIVATION AMPLITUDE	COVARIANCE	CORRELATION	SPARSITY	DEAD UNIT	RANK
Baseline	2.85	4.93	2.08	0.27	0.34	0.13	2.41
L1W	2.85	4.53	1.95	0.28	0.29	0.01	2.32
L2W	3.02	4.76	2.23	0.29	0.34	0.09	2.26
Dropout	2.70	2.72	0.87	*0.42*	0.58	0.06	2.75
BN	2.81	1.35	0.24	0.28	0.52	0.00	5.14
CR	2.50	0.50	**0.01**	**0.19**	0.40	0.03	7.12
cw-CR	2.49	0.63	0.02	0.31	0.51	0.07	3.60
VR	2.65	1.35	0.15	0.26	0.40	0.08	3.92
cw-VR	2.42	0.63	0.02	0.36	0.53	0.06	3.90
L1R	2.35	1.29	0.03	0.40	**0.97**	*0.39*	5.94
RR	2.81	7.23	226.2	0.90	0.43	0.18	**1.00**
cw-RR	2.57	10.31	96.3	0.91	0.31	0.22	**1.00**

4.3. Scaling, Permutation, Covariance, and Correlation

After training is completed for a deep network $\mathcal{N}_{\mathcal{A}}$, the output of the network becomes a deterministic function of the input $\mathbf{x}$. Without an activation function, i.e. a linear layer, $\mathbf{z}_l = \mathbf{W}_l^T \mathbf{z}_{l-1} + \mathbf{b}_l$. When ReLU is applied to layer l, the activation vector becomes $\mathbf{z}_l = \text{ReLU}(\mathbf{W}_l^T \mathbf{z}_{l-1} + \mathbf{b}_l) = \max(\mathbf{W}_l^T \mathbf{z}_{l-1} + \mathbf{b}_l, 0)$. In this section, we investigate the most flexible affine transformation that can be applied to a layer's representation $\mathbf{z}_l$ without influencing the output $\mathcal{N}_{\mathcal{A}}(\mathbf{x})$ for any arbitrary input vector $\mathbf{x}$. While complicated transformations over multiple layers can be explored, we limit our focus to manipulating only the weights of layer l and layer $l+1$ for the analytical tractability. Because scaling and permutation are well known results, covariance and correlation are the main focus of this section.

4.3.1. Identical Output Network (ION)

We first consider a linear layer l. For a linear layer, it turns out that any affine transformation can be applied as long as the transformation does not cause an information loss.

Theorem 1. ***(ION for a linear layer)*** *For a deep network $\mathcal{N}_\mathcal{A} = (\mathbf{W}, \mathbf{b})$ whose layer l is linear, there exists $\widetilde{\mathcal{N}}_\mathcal{A} = (\widetilde{\mathbf{W}}, \widetilde{\mathbf{b}})$ that satisfy the following conditions:*

$$\forall \mathbf{x},\ \mathcal{N}_\mathcal{A}(\mathbf{x}) = \widetilde{\mathcal{N}}_\mathcal{A}(\mathbf{x}); \tag{4.1}$$

$$\forall \mathbf{x},\ \widetilde{\mathbf{z}}_l = \mathbf{Q}(\mathbf{z}_l - \mathbf{m}), \tag{4.2}$$

where $\mathbf{Q}$ is any nonsingular square matrix of a proper size and $\mathbf{m}$ is any vector of a proper size.

The first condition says that the two networks generate identical outputs for any input $\mathbf{x}$. The second condition says that $\mathbf{z}_l$ can be affinely transformed using any nonsingular matrix $\mathbf{Q}$. The proof is straightforward and can be found in Appendix B.

While simple, Theorem 1 has significant implications on the representation characteristics of $\mathbf{z}_l$. Let's inspect covariance and correlation (normalized version of covariance) first. If $\mathcal{N}_\mathcal{A}$ is a network that is globally optimal for a task and has at least one linear layer l, then $\mathcal{N}_\mathcal{A}$'s covariance $\mathbf{C}_l$ can be whitened to have $\widetilde{\mathbf{C}}_l = \mathbf{I}$ by choosing $\mathbf{m}$ as the expected mean and $\mathbf{Q}$ as a whitening matrix. The resulting network $\widetilde{\mathcal{N}}_\mathcal{A}$ will have zero correlation between any pair of units in layer l, but will be globally optimal, too. In fact, there are infinitely many globally optimal networks with different covariance characteristics, and one can easily construct an ION with an arbitrary covariance matrix $\widetilde{\mathbf{C}}_l$ as long as its rank is the same as $\mathbf{C}_l$'s rank. With this result, it becomes unclear why one should pursue a lower correlation when training a deep network. Unless regularization for a low correlation somehow helps optimization to reach a better performing network, there seems to be no reason to pursue low (or high) correlation.

For dead neurons, a similar claim can be made. If globally optimal $\mathcal{N}_\mathcal{A}$ has no dead neurons in layer l and $\mathbf{C}_l$ is not full rank, one can make an affine transformation to align the null spaces of $\mathbf{C}_l$ to some of the neurons. Then, the resulting network $\widetilde{\mathcal{N}}_\mathcal{A}$ will be still globally optimal, but with some dead neurons in layer l. For higher layers of classification tasks, typically the rank of $\mathbf{C}_l$ is close to the number of classes. For

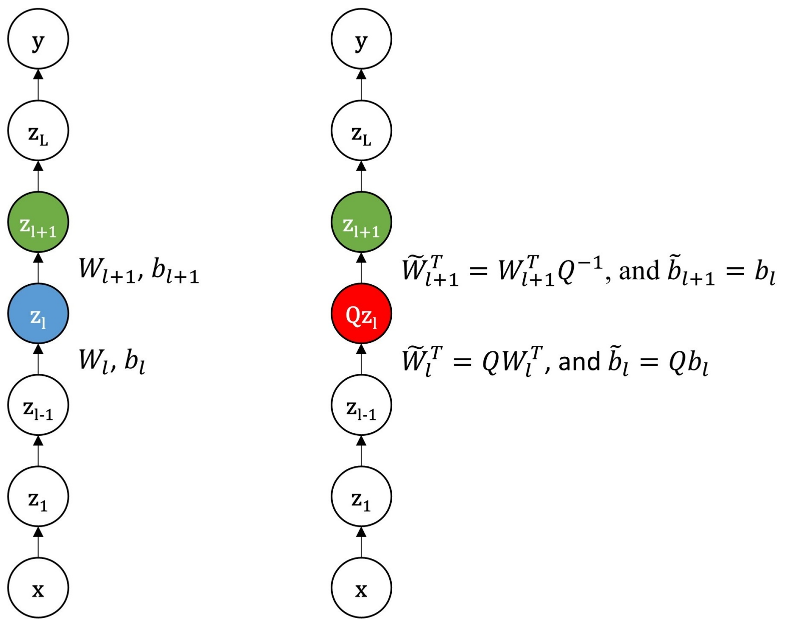

y
z_L
z_{l+1}
W_{l+1}, b_{l+1}
z_l
W_l, b_l
z_{l-1}
z_1
x
(a) Original network
y
z_L
z_{l+1}
$\widetilde{W}_{l+1}^T = W_{l+1}^T Q^{-1}$, and $\tilde{b}_{l+1} = b_l$
Qz_l
$\widetilde{W}_l^T = QW_l^T$, and $\tilde{b}_l = Qb_l$
z_{l-1}
z_1
x
(b) ION (transformed network)

Figure 4.2. Illustration of an ION. The representation $\mathbf{z}_l$ (blue circle) of the original network in (a) is different from $\mathbf{Q}\,\mathbf{z}_l$ (red circle) of the ION in (b). However, their upper layer representations $\mathbf{z}_{l+1}$ (green circle) are the same, so the outputs ($\mathbf{y}$) are identical to each other.

classification tasks with only 2∼10 classes, it is possible to construct an ION that has as many dead neurons as the size of $\mathbf{C}_l$'s null space. This can be done without negatively affecting the performance, and the wisdom of 'reduce the number of dead neurons' becomes dubious.

For scaling and permutation, their influences are rather insignificant. As for the scaling that can affect activation amplitude, it often has no effect on the network's performance. For instance, scaled activation amplitude can affect the probability of classification tasks when softmax is in the last layer, but the class with the highest probability remains the same anyway. When representation regularizers are used, often activation amplitude is squashed to reduce the cost of the representation penalty function, but the network can still perform well. As reported in Choi and Rhee (2018), such an activation squashing can make covariance much smaller, but the effect is removed when a correlation value is calculated. As for the permutation, it is considered to be meaningless because the index number itself is not important.

Before discussing further, a similar result is developed for ReLU layers. The resulting $\mathbf{Q}$, however, is much more limited. The proof can be found in Appendix B as well.

Theorem 2. ***(ION for a ReLU layer)*** *For a deep network $\mathcal{N}_{\mathcal{A}} = (\mathbf{W},\, \mathbf{b})$ whose activation function of layer l is ReLU, there exists $\widetilde{\mathcal{N}}_{\mathcal{A}} = (\widetilde{\mathbf{W}},\, \widetilde{\mathbf{b}})$ that satisfy the following conditions:*

$$\forall\, \mathbf{x},\ \mathcal{N}_{\mathcal{A}}(\mathbf{x}) = \widetilde{\mathcal{N}}_{\mathcal{A}}(\mathbf{x}); \tag{4.3}$$

$$\forall\, \mathbf{x},\ \widetilde{\mathbf{z}}_l = \mathbf{Q}\,\mathbf{z}_l, \tag{4.4}$$

where $\mathbf{Q}$ is any permuted positive diagonal matrix of a proper size. Furthermore, it can be shown that any $\mathbf{Q}$ that satisfy the above two conditions must be a permuted positive diagonal matrix.

Using a permuted positive diagonal matrix $\mathbf{Q}$, covariance can be affected by independently scaling activation amplitudes of layer l's units. As explained above, such scaling is canceled out when calculating correlation and therefore a linear transformation cannot affect correlation while keeping the output identical. There are a few possibilities for overcoming the limitations of Theorem 2, and they are discussed in the following subsection.

For rank and mutual information, the invertible affine transformation has no effect. They are discussed in the following sections.

4.3.2. Possible Extensions for ION

We discuss three possible extensions for ReLU's ION.

Table 4.4. Comparison of statistical characteristics for linear and ReLU layers. A 7-layer MLP was used with MNIST dataset, and only the sixth layer was linear, and the others were ReLU layers. Statistical characteristics of the first layer (ReLU), fifth layer (ReLU), and the sixth layer (linear) are compared. It can be seen that representation characteristics of fifth and sixth are very similar because they are both located in the upper part of the network. Note that the characteristics of the sixth layer were calculated only using positive activation values for a fair comparison.

Regularizer	ACTIVATION AMPLITUDE	COVARIANCE	CORRELATION	SPARSITY	DEAD UNIT	RANK
Baseline (1st)	1.16	0.08	0.14	0.00	0.00	2.22
Baseline (5th)	2.22	0.45	0.25	0.32	0.07	2.27
Baseline (6th)	2.75	0.62	0.25	0.47	0.00	2.78
Dropout (1st)	0.76	0.04	0.27	0.86	0.00	3.50
Dropout (5th)	2.19	1.23	0.70	0.53	0.00	2.19
Dropout (6th)	1.85	0.97	0.53	0.48	0.01	1.52
BN (1st)	0.80	0.03	0.10	0.49	0.00	4.07
BN (5th)	1.01	0.10	0.19	0.51	0.00	4.59
BN (6th)	1.39	0.19	0.23	0.48	0.00	4.26

Insertion of a linear layer One way to overcome the limitations of $\mathbf{Q}$ is to insert an extra linear layer near the target ReLU layer and to consider its implications. When representation characteristics are analyzed or interpreted, researchers do not care much about the layer's activation function, regularization, etc. The activation vector's representation characteristics are the essential components for understanding and assessing the representations. Therefore, one can apply the insights from Theorem 1 when the extra linear layer shows similar statistical properties as the ReLU layer. In Table 4.4, statistical properties of immediately neighboring ReLU and linear layers are compared. Compared to the representation characteristics of the first ReLU layer, the 5th ReLU layer and the inserted sixth linear layer show very similar characteristics. Then the correlation and dead unit characteristics are not so important as the consequence of Theorem 1, and the same might be conjectured for the 5th ReLU layer.

Comparable Performance Network (CPN) According to Theorem 2, only permuted positive diagonal matrices can form IONs. If we ignore the result and apply an affine transformation in the same way as in the ION of a linear layer, the resulting network $\widetilde{\mathcal{N}}_{\mathcal{A}}$ will not form an ION, but instead, we might be able to find a Comparable Performance Network (CPN) that achieves a comparable performance while showing different representation characteristics. We tried this brute-force method, and two sample results along with the baseline and a positive diagonal matrix case are shown in Table 4.5. In the first row where $\mathbf{Q}$ is identity, the values are for the original network $\mathcal{N}_{\mathcal{A}}$. In the next row, 'Random positive diagonal,' uniformly random values between 0 and 1 ($U(0,1)$) were used as the diagonal values. Note that this choice of $\mathbf{Q}$ satisfies Theorem 2, and therefore the error performance remains the same while affecting activation amplitude and covariance only. In the 'Random with ones in diagonal,' $\mathbf{Q}$ was chosen as a matrix of random values selected from $U(0,1)$ with its diagonals replaced with ones. We randomly generated 100 of such random matrices and selected the one that resulted in a higher correlation while showing a comparable performance. Despite the very high correlation of 0.80, the selected network $\widetilde{\mathcal{N}}_{\mathcal{A}}$ can perform comparably

Table 4.5. Statistical characteristics of representations transformed by CPNs. The original network is 6-layer MLP on the MNIST dataset, and the 5th layer representations were transformed. To improve the performance, the weights to the output layer were fine tuned after applying $\mathbf{Q}$.

Q matrix	Test Error (%)	ACTIVATION AMPLITUDE	COVARIANCE	CORRELATION	SPARSITY
Identity	2.54	6.79	4.29	0.28	0.36
Random positive diagonal	2.54	**3.37**	**1.04**	0.28	0.36
Random with ones in diagonal	2.76	158.88	723.55	**0.80**	**0.00**
Whitening	5.48	1.22	0.96	**0.09**	**0.49**

well. In the last row, we applied a whitening filter where $\mathbf{Q}$ and $\mathbf{m}$ were calculated while ignoring ReLU. The resulting network does not end up with zero correlation because the whitening is not perfect in the presence of ReLU. However, the correlation is considerably reduced to 0.09 from 0.28 while achieving a slightly worse error rate of 5.48%.

To find the examples in Table 4.5, all we had to do was to construct a meaningful matrix $\mathbf{Q}$ or to try 100 random matrices and choose one. The fact that it is an almost painless job to find a CPN also implies that the relevant representation characteristics might not be essential for achieving high performance.

Non-affine transformations over multiple layers In the ION derivations, we have considered only an affine transformation applied to the layers l and $l+1$ only. If we remove the constraints and borrow the results from expressivity of DNN and universal approximation theorem, it might be possible to derive more powerful and general results. In the extreme case, one can divide a deep network $\mathcal{N}_{\mathcal{A}}$ into two parts: $\mathcal{N}_{\mathcal{A}_{\text{lower}}}$ and $\mathcal{N}_{\mathcal{A}_{\text{upper}}}$. Then, $\mathcal{N}_{\mathcal{A}_{\text{lower}}}(\mathbf{x}) = \mathbf{z}_l$ and $\mathcal{N}_{\mathcal{A}}(\mathbf{x}) = \mathcal{N}_{\mathcal{A}_{\text{upper}}}(\mathcal{N}_{\mathcal{A}_{\text{lower}}}(\mathbf{x}))$. In theory, there exist $\widetilde{\mathcal{N}}_{\mathcal{A}_{\text{lower}}}$ and $\widetilde{\mathcal{N}}_{\mathcal{A}_{\text{upper}}}$ that can result in $\mathcal{N}_{\mathcal{A}}(\mathbf{x}) = \widetilde{\mathcal{N}}_{\mathcal{A}_{\text{upper}}}(\widetilde{\mathcal{N}}_{\mathcal{A}_{\text{lower}}}(\mathbf{x}))$ while allowing $\widetilde{\mathbf{z}}_l$ to have a completely different characteristics compared to $\mathbf{z}_l$. Such $\widetilde{\mathcal{N}}_{\mathcal{A}_{\text{lower}}}$ and $\widetilde{\mathcal{N}}_{\mathcal{A}_{\text{upper}}}$, however, might be infeasibly large or fail to learn in the way we desire. Therefore, it might be more practical to consider a reasonable extension of Theorem 1 and Theorem 2.

4.4. Sparsity, Dead Unit, and Rank

Sparsity and dead unit have been considered as important representation characteristics. Rank of $\mathbf{C}_l$, however, has received much less attention so far. In this section, we investigate the three and show that rank might be the most fundamental characteristic.

4.4.1. Analytical Relationship

In Table 4.1, sparsity is defined as $P_s = \mathbb{E}_{i,n}[\mathbb{1}(z^n_{l,i})]$. This can be interpreted as the probability of $z^n_{l,i}$ (unit i's activation for n'th test sample $\mathbf{x}^n$) being zero, because $\mathbb{1}(z^n_{l,i}) = 1$ when $z^n_{l,i} = 0$ and $\mathbb{1}(z^n_{l,i}) = 0$ when $z^n_{l,i} \neq 0$. Similarly, dead unit is defined as $P_d = \mathbb{E}_i[\mathbb{1}(z_{l,i})]$ and it can be interpreted as the probability of $z_{l,i}$ (unit i's activation) being always zero or at least for all M test samples. Because $\mathbb{1}(z_{l,i}) \leq \mathbb{1}(z^n_{l,i})$ for any pair of (i, n), $P_d \leq P_s$ can be shown by taking expectations on both sides. The rank r in Table 4.1 is defined as the rank of $\mathbf{C}_l$. For layer l with M units, this means that r out of M linearly independent dimensions are used by the codewords $\{\mathbf{z}^1_l, ..., \mathbf{z}^N_l\}$ and that the other $M - r$ dimensions form a null space of $\mathbf{C}_l$. When dead units are considered, MP_d units need to be constant zero by the definition of the dead unit and it implies that at least MP_d dimensions need to be included in the null space. Therefore, $MP_d \leq M - r$. These results can be summarized as below.

$$P_d \leq P_s \tag{4.5}$$

$$MP_d \leq M - r \tag{4.6}$$

Between sparsity P_s and rank r, there is no clear relationship. The codeword $\mathbf{z}_l^n$ for a test sample $\mathbf{x}^n$ can be very sparse, and yet the set of codewords $\{\mathbf{z}_l^1, ..., \mathbf{z}_l^N\}$ collectively might use all of the M dimensions. Conversely, rank r can be very small and yet P_s can also be very small when $\{z_{l,1}^n, ..., z_{l,M}^n\}$ are strongly correlated and the basis vectors are not sparse over the M units.

From the viewpoint of signal processing or information theory, sparsity is a property that is related to individual signals or individual codewords while rank is a property that is related to the total number of dimensions used by the set of signals or the entire codebook. Therefore, sparsity is not directly responsible for the efficiency of the signals or codebook while rank is directly responsible for the efficiency. From the viewpoint of deep learning, rank can be associated with the maximum number of latent factors that are independent. As for the dead unit, we know from equation 4.6 that it is upper bounded as a function of rank. When the bound is met, the value of P_d is merely an artifact of how the representation vectors $\{\mathbf{z}_l\}$ are aligned with the eigenvectors of $\mathbf{C}_l$. If each neuron is aligned to an eigenmode of $\mathbf{C}_l$, then $P_d = 1 - r/M$ will be achieved. From our experience, however, such a perfect alignment never happens when using the backpropagation based learning process. This has been true even when L1R or other advanced representation regularizers were applied. ION, however, can easily meet the requirement for a linear layer.

Motivated by the above discussion, we have designed a rank regularizer and examined common wisdom that says 'most of the data generation processes have a small number of independent factors and therefore increasing sparsity of representation can be helpful.' For instance, see Bengio, Courville, and Vincent (2013). We first explain the design of rank regularizer.

4.4.2. Rank Regularizer

In deep learning, a low-rank approximation of convolutional filters (Jaderberg, Vedaldi, and Zisserman 2014; Lebedev et al. 2015; Tai et al. 2016) and weight matrices (Xue, Li, and Gong 2013; Xue et al. 2014; Nakkiran et al. 2015; Masana et al. 2017; Alvarez and Salzmann 2017) has been widely used for network compression and fast network training. Some of the works applied a singular value decomposition to weight matrices after network training ends but not to representations. In this work, as L1 representation regularizer was designed to encourage a higher sparsity by adding a penalty loss term $\Omega_{L1R} = \sum_n \sum_i |z_{l,i}^n|$, Rank Regularizer (RR) is designed to encourage a lower rank of representations and used during network training. Because the usual definition of rank can be very sensitive to the tiny singular values, we use *stable rank* of activation matrix $\mathbf{Z} = [\mathbf{z}_l^1, \ldots, \mathbf{z}_l^{N_{MB}}]^T$ as a surrogate. Note that N_{MB} instead of N activation vectors are used for each mini-batch. Stable rank of $\mathbf{Z}$ is defined as

$$\Omega_{RR} = \frac{\|\mathbf{Z}\|_F^2}{\|\mathbf{Z}\|_2^2} = \frac{\sum_i s_i^2}{\max_i s_i^2}, \tag{4.7}$$

where $\|\mathbf{Z}\|_F$ is the Frobenius norm, $\|\mathbf{Z}\|_2$ is the spectral norm, and $\{s_i\}$ are the singular values of $\mathbf{Z}$. From $\frac{\sum_i s_i^2}{\max_i s_i^2}$, it can be clearly seen that stable rank is upper bounded by the usual rank that counts strictly positive singular values. Because the spectral norm is based on a singular value decomposition, calculating stable rank's derivative for every mini-batch is a computationally heavy operation. To reduce the computational burden, we introduce an approximation using a special case of Holder's inequality.

$$\begin{align}
\Omega_{RR} = \frac{\|\mathbf{Z}\|_F^2}{\|\mathbf{Z}\|_2^2} &= \frac{\text{trace}(\mathbf{Z}^T\mathbf{Z})}{\|\mathbf{Z}\|_2^2} \tag{4.8}\\
&\geq \frac{\text{trace}(\mathbf{Z}^T\mathbf{Z})}{\|\mathbf{Z}\|_1\|\mathbf{Z}\|_\infty} = \frac{\sum_{i,n}(z_i^n)^2}{(\max_i \sum_{n=1}^{N_{MB}} |z_i^n|)(\max_n \sum_{i=1}^{M} |z_i^n|)} \tag{4.9}
\end{align}$$

The inequality $\|\mathbf{Z}\|_2 \leq \sqrt{\|\mathbf{Z}\|_1\|\mathbf{Z}\|_\infty}$ was used where $\|\mathbf{Z}\|_1$ is the maximum absolute column sum of the matrix $\mathbf{Z}$ (sum of all activation values of unit i) and $\|\mathbf{Z}\|_\infty$ is the maximum absolute row sum of the matrix $\mathbf{Z}$ (sum of all activation values of sample n). Then the gradient of approximation Ω_{RR} can be written as below.

$$\frac{\partial \Omega_{RR}}{\partial z_i^n} \simeq \frac{\frac{\partial \|\mathbf{Z}\|_F^2}{\partial z_i^n}}{\|\mathbf{Z}\|_1 \|\mathbf{Z}\|_\infty} - \frac{\|\mathbf{Z}\|_F^2 \cdot (\frac{\partial \|\mathbf{Z}\|_1}{\partial z_i^n} \cdot \|\mathbf{Z}\|_\infty + \|\mathbf{Z}\|_1 \cdot \frac{\partial \|\mathbf{Z}\|_\infty}{\partial z_i^n})}{\|\mathbf{Z}\|_1^2 \|\mathbf{Z}\|_\infty^2}, \tag{4.10}$$

$$\text{where } \frac{\partial \|\mathbf{Z}\|_F^2}{\partial z_i^n} = 2 z_i^n,$$

$$\frac{\partial \|\mathbf{Z}\|_1}{\partial z_i^n} = \mathbb{1}_{(i=i^*)} \cdot \text{sign}(z_i^n), \tag{4.11}$$

$$\frac{\partial \|\mathbf{Z}\|_\infty}{\partial z_i^n} = \mathbb{1}_{(n=n^*)} \cdot \text{sign}(z_i^n),$$

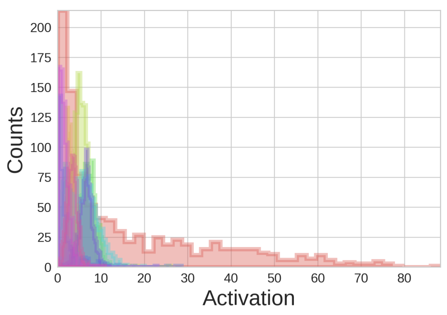
$$i^* = \underset{1 \le i \le M}{\arg\max} \sum_{n=1}^{N_{MB}} |z_i^n|, \text{ and} \tag{4.12}$$

$$n^* = \underset{1 \le n \le N_{MB}}{\arg\max} \sum_{i=1}^{M} |z_i^n|.$$

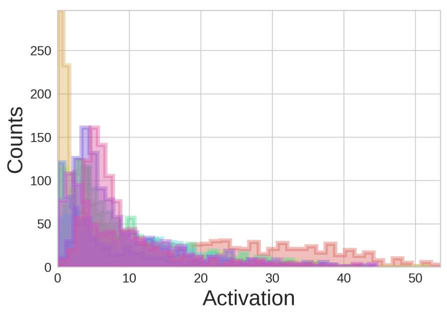
The extension of RR to its class-wise counterpart, cw-RR, is obvious. The activation histograms and scatter plots of randomly chosen units are shown in Figure 4.3 and 4.4, respectively. One can observe that two hidden units are highly correlated in both RR and cw-RR, which is consistent with their CORRELATION in Table 4.3. Surprisingly, the classification performance of cw-RR is comparable to that of CR having very low CORRELATION and that of the baseline. The result is against the conventional wisdom that correlated representations hurt the performance of deep networks. When the performance of cw-RR leading to large correlation is compared with that of the baseline in a range of condition tasks, no systematic patterns to confirm one is better than the other can be found as shown in Table 5.2, 5.3, and 5.4 of Chapter 5.

4.4.3. A Controlled Experiment on Data Generation Process

We have designed two datasets where the number of independent factors is fully controlled. The first dataset is a synthetic 10-class classification dataset that was created using Python scikit-learn library (Pedregosa et al. 2011). The number of independent Gaussian factors, *d*, was controlled to be 10, 50, 100, 250, and 500, and the independent factors were mixed using a randomly generated $1000 \times d$ rotation matrix. The second

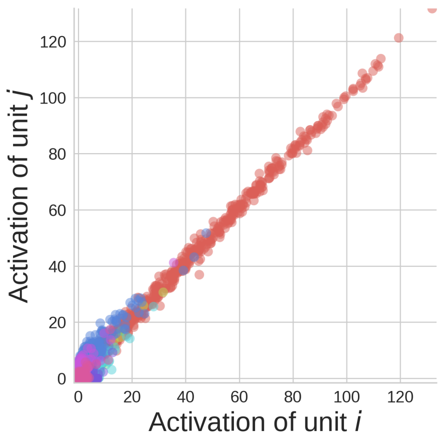


(a) RR

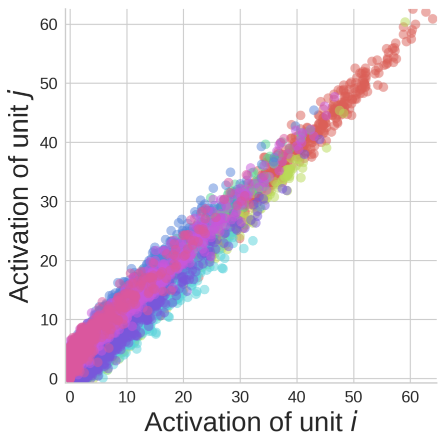


(b) cw-RR

Figure 4.3. Activation histogram of a unit (RR and cw-RR) 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.



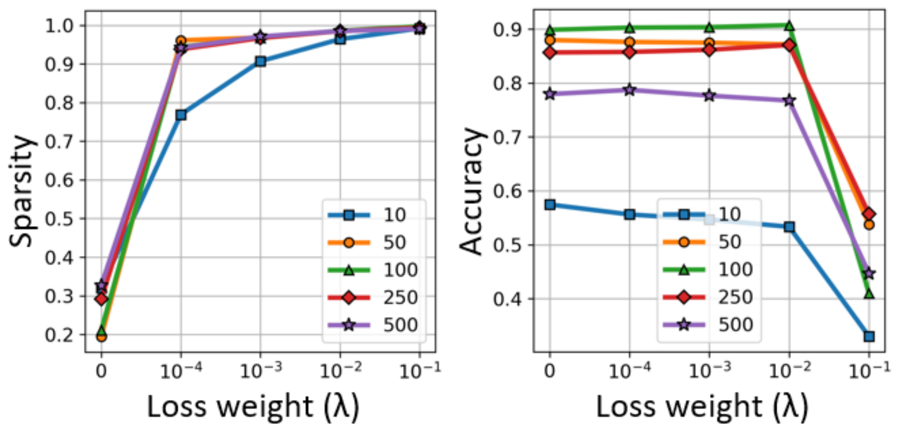
(a) RR



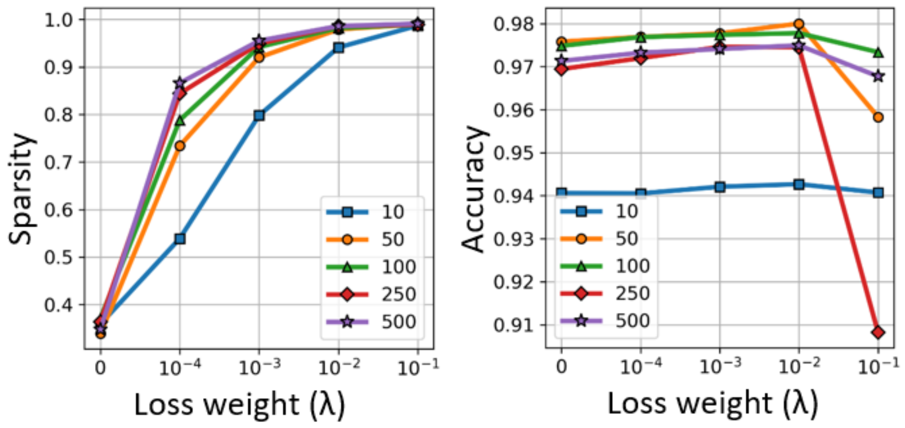
(b) cw-RR

Figure 4.4. Scatter plot of two units (RR and cw-RR) 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. Note that a large correlation between each pair of classes can be observed in the scatter plot of both (a) and (b).

dataset is a PCA-controlled MNIST data that was created by including only the top 10, 50, 100, 250, and 500 dimensions of MNIST's PCA dimensions.

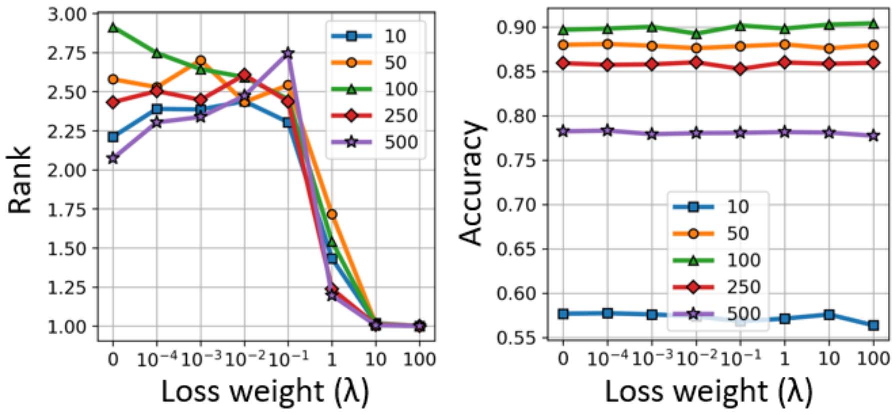


(a) Synthetic data

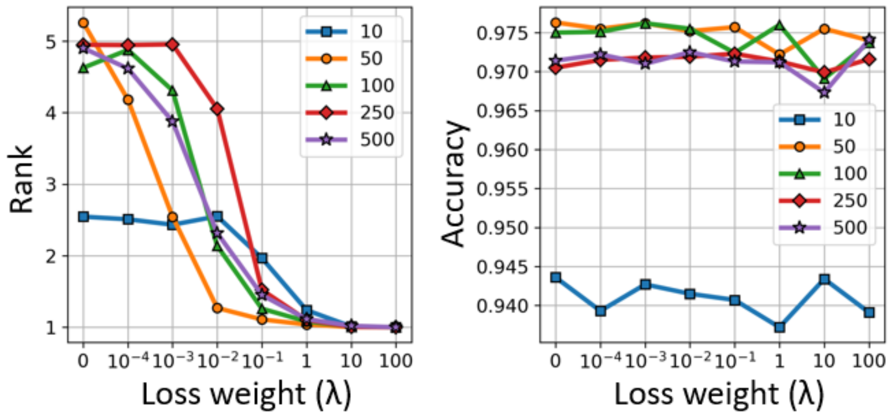


(b) PCA-controlled MNIST data

Figure 4.5. Effect of L1R (L1 Representation Regularizer). Representation sparsity (P_s) and accuracy are shown as a function of L1R's loss weight. Each line corresponds to a different number of independent factors. While sparsity is well controlled, test accuracy does not show any meaningful dependency on the number of independent factors used in the data generation process.



(a) Synthetic data



(b) PCA-controlled MNIST data

Figure 4.6. Effect of RR (Rank Regularizer). Representation rank (r) and accuracy are shown as a function of RR's loss weight. Each line corresponds to a different number of independent factors. While rank is well controlled, test accuracy does not show any meaningful dependency on the number of independent factors used in the data generation process.

For the two datasets, we have chosen $\mathcal{N}_{\mathcal{A}}$ to be the same 6-layer MLP as before and repeatedly performed training while applying either L1R or RR with different loss weights. The results are shown in Figure 4.5 and Figure 4.6. The sparsity and rank plots show that indeed sparsity is increased and rank is reduced by increasing the loss weight. The accuracy performance, however, does not show any meaningful dependency on d. For instance, even when $d = 10$, and there were only ten independent factors in the data generation process, strongly applying L1R or RR did not result in improved performance. On the contrary, the accuracy often suffered when loss weight was increased.

According to the discussion in subsection 4.4.1, it is not surprising that the level of learned representation's sparsity does not affect the accuracy performance. Perhaps it is more surprising that even the level of learned representation's rank does not affect the accuracy performance. We move on to the analysis of mutual information for a further discussion on this issue.

4.5. Mutual Information

So far, we have investigated popular statistical characteristics of representation $\mathbf{z}_l$ where none of the statistical characteristics showed a strong and apparent relationship to a deep network's performance. In this section, we examine two information-theoretic characteristics: $I(\mathbf{z}_l;\mathbf{x})$ and $I(\mathbf{z}_l;\mathbf{y})$. In the original and pioneering work of Shwartz-Ziv and Tishby (2017), the two characteristics of $\mathbf{z}_l$ were used to explain the concept of information bottleneck on deep networks. Basically, the work shows that the task-relevant information should be maximized via $I(\mathbf{z}_l;\mathbf{y})$ while the task-irrelevant information should be minimized via $I(\mathbf{z}_l;\mathbf{x})$. A further development was made in Achille and Soatto (2018a), where the Information Bottleneck Lagrangian $\mathcal{L}(p(\mathbf{z}_l\,|\,\mathbf{x})) = H(\mathbf{y}\,|\,\mathbf{z}_l) + \beta I(\mathbf{z}_l;\mathbf{x})$ was explained - the first term is the usual cross entropy cost function, the second term is a penalty term on $I(\mathbf{z}_l;\mathbf{x})$, and β is a parameter for controlling a tradeoff between sufficiency (the first term) and minimality (the second

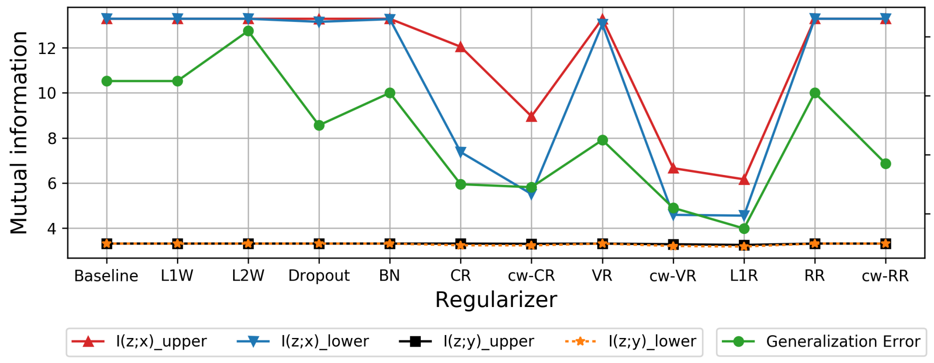


Figure 4.7. Mutual information and generalization error. The same MLP and MNIST dataset were used to conduct this experiment. The regularizers were applied to the fifth layer, and the upper and lower bounds of mutual information were calculated using the layer's activation vectors.

term). In Achille and Soatto (2018b), they develop 'information dropout' method that implicitly minimizes $I(\mathbf{z}_l;\mathbf{x})$. In their limited performance experiments, they showed that information dropout could improve MNIST classification performance by about 0.25% for the best case. In another work by Kolchinsky, Tracey, and Wolpert (2017), an upper bound derived using a non-parametric estimator of mutual information and a variational approximation is used to develop a gradient-based optimization method. They showed $I(\mathbf{z}_l;\mathbf{x})$ and $I(\mathbf{z}_l;\mathbf{y})$ are indeed reduced by the method, but did not report anything on performance.

In this work, we neither tried the aforementioned techniques nor explicitly implemented an $I(\mathbf{z}_l;\mathbf{x})$ regularizer. Instead, we simply applied the twelve regularizers (including baseline) and calculated the upper and lower bounds of $I(\mathbf{z}_l;\mathbf{x})$ and $I(\mathbf{z}_l;\mathbf{y})$. The bounds can be calculated using the results of Kolchinsky, Tracey, and Wolpert (2017) where a pairwise distance function between mixture components is used. They prove that the Chernoff α-divergence and the Kullback-Leibler divergence provide lower and upper bounds when they are chosen as the distance function, respectively.

Figure 4.7 shows the results for the last hidden layer together with the generalization error where the same network and dataset as in Figure 4.1 were used. Regularizers were also applied to the last hidden layer. One can observe that all the regularizers end up with almost the same $I(\mathbf{z}_l; \mathbf{y})$ value. However, the bounds of $I(\mathbf{z}_l; \mathbf{x})$ can be seen to be strongly dependent on which regularizer is used, and the upper and lower bounds show a similar pattern as the generalization error's pattern. In fact, the correlation between the lower bound and generalization error can be calculated to be 0.84, and the correlation between the upper bound and generalization error can be calculated to be 0.78. Therefore, it can be surmised that the regularizers might be indirectly affecting the performance by influencing $I(\mathbf{z}_l; \mathbf{x})$.

When a mutual information regularizer is excluded, the rest of the representation regularizers fail to provide general reasoning on why any of the statistical characteristics should be pursued. In fact, one can argue that even a single-neuron in layer l (activation becomes a scalar) can be a sufficient condition for encoding to have a chance to achieve the maximum possible $I(\mathbf{z}_l; \mathbf{y})$, i.e., lossless in terms of relevant information. Such an encoding on a scalar activation might be very inefficient, and a practical learning method might never reach such an encoding. Nonetheless, there is no reason why such encoding should be impossible. Obviously, many of the statistical characteristics become meaningless for such a scalar representation, and it is high time to reconsider the so-called conventional wisdom on representation characteristics.

Chapter 5. Practical Ways of Using Representation Regularizers

In this chapter, with representation regularizers proposed in this work, we address how to use representation regularizers in practice. Performance tuning and compression of deep networks are considered as practical ways of using representation regularizers.

5.1. Tuning Deep Network Performance Using Representation Regularizers

In the previous chapter, we have investigated representation characteristics and their relationship to performance. All the results, except for mutual information that is shown in Figure 4.7, indicate that there might be no firm ground to believe that $\mathbf{z}_l$'s representation characteristics are strongly related to performance. However, there have been numerous reports that performance was improved by utilizing newly designed regularizers. In this section, we investigate if (representation) regularizers can indeed consistently improve the performance for a given task condition. Here, a task condition means a learning task with small data size, a small layer width, a specific dataset, a large number of classes, or a specific optimizer. We perform experiments on MNIST, CIFAR-10, and CIFAR-100 datasets using twelve regularizers. The representation regularizers are explained in Table 4.2 of Chapter 4. The details of experimental settings and architecture parameters can be found in the next subsection. Based on the results of this section, we further discuss how to use representation regularizers as a hyperparameter in Chapter 6.

5.1.1. Experimental Settings and Conditions

By default, we chose ReLU, SGD with the Adam optimizer, and a learning rate of 0.0001 for networks. Mini-batch size is set to 100 by default but is set to 500 only for CIFAR-100. We evaluated validation performance for {0.001, 0.01, 0.1, 1, 10, 100} and chose the one with the best performance for each regularizer and condition. Then, performance was evaluated through five trainings using the pre-fixed weight value. In the case of CIFAR-10 and CIFAR-100, the last 10,000 instances of 50,000 training data were used as the validation data, and after the weight values were fixed, the validation data was merged back into the training data. All experiments in this work were carried out using TensorFlow 1.5.

Architecture for MNIST

For classification tasks, a 6-layer MLP that has 100 hidden units per layer was used. For image reconstruction task, a 6-layer autoencoder was used. The number of hidden units in each layer is 400, 200, 100, 200, 400, and 784 in the order of hidden layers.

Architecture for CIFAR-10 and CIFAR-100

A CNN with four convolutional layers and one fully connected layer was used for both of CIFAR-10 and CIFAR-100. Detailed architecture hyperparameters are shown in Table 5.1.

Experimental Conditions

Default conditions are shown in bold, and the full experimental conditions are listed below.

- Training data size: 1k, 5k, **50k**
- Layer width: (MNIST) 2, 8, **100** / (CIFAR-10/100): 32, **128**, 512

Table 5.1. Default architecture hyperparameters of CIFAR-10/100 CNN model.

Layer	# of filters (or units)	Filter size	Conv. stride	Pooling size	Pooling stride
Convolutional layer-1	32	3×3	1	-	-
Convolutional layer-2	64	3×3	1	-	-
Max-pooling layer-1	-	-	-	2×2	2
Convolutional layer-3	128	3×3	1	-	-
Max-pooling layer-2	-	-	-	2×2	2
Convolutional layer-4	128	3×3	1	-	-
Max-pooling layer-3	-	-	-	2×2	2
Fully connected layer	128	-	-	-	-

- Optimizer (CIFAR-10): **Adam**, Momentum (lr=0.01, momentum=0.9), RMSProp (lr=0.0001)
- Number of classes (CIFAR-100): 16, 64, **100**

5.1.2. Consistently Well-performing Regularizer

We analyze if there is a logical dependency between a regularizer and its effect on the performance when a particular regularizer is applied to a particular task condition. Our results, as shown in Table 5.2, 5.3, and 5.4, indicate that there is no consistently well-performing regularizer for a specific task condition. As an example, consider the entire CIFAR-10 dataset results in Table 5.2. While task conditions change over different columns, the data remains common for all the tasks. If there is a representation characteristic that fits the data-generation process well and one of the regularizers could match the representation well, it might have outperformed across all the columns. In the table, the best performing regularizer for each task (column) is marked in bold, and any other regularizer whose performance overlaps with the best one is highlighted in gray. Looking at the bold and gray-highlight patterns, one can easily conclude that no single regularizer works well for all the tasks of CIFAR-10. A similar observation can be made for other task conditions. For instance, one can examine the data size of 1k. For the 1k columns of the three tables, no single regularizer always performs distinctively well.

In fact, we have experimented many more settings than what are shown in this dissertation. We hoped to find a strong match between task conditions and representation regularizers, but we have failed to find anything that looks consistent. Many previous works on regularizers have compared their regularizers with only a small number of other known regularizers. When many regularizers are compared over many different tasks as in our work, one can easily conclude that there is no apparent relation to declare where a specific representation characteristic is advantageous.

5.1.3. Performance Improvement Using Regularizers as a Set

Even though no single representation characteristic consistently outperforms, it can be seen that one can improve performance by using the twelve regularizers as a set and by choosing the best performing regularizer for the given task. This is in line with the usual theme of *tuning* in many areas of deep learning. Looking more carefully into Table 5.2, we can see that cw-VR and L1R often had the best performance for CIFAR-10 test cases. In our experiments, we observed that one of the representation regularizers often outperforms weight regularizers (L1W, L2W), dropout, and BN. Though representation regularizers do not seem to have a direct impact on the performance, they might have indirect effects on mutual information as we have seen in Chapter 4 or on the optimization process. When many representation regularizers are tried as a set, perhaps there is a more significant chance of one of such indirect effects that improves the performance.

5.2. Enhancing Network Compression Using Representation Regularizers

In this section, we propose network compression methods that utilize representation regularizers. We begin this section by addressing why network compression is required

Table 5.2. The best performing regularizer in each condition (each column) is shown in bold, and other regularizers whose performance range overlaps with that of the best ones are highlighted in gray. For the default condition, the standard values of data size=50k and layer width=128 were used, and Adam optimizer was applied. For other columns, all the conditions were the same as the default, except for the condition indicated on the top part of the column. Regularizers were applied to the last hidden layer.

Regularizer	Default	Data Size		Layer Width		Optimizer	
		1k	5k	32	512	Momentum	RMSProp
Baseline	26.64 ± 0.16	56.07 ± 0.36	43.95 ± 0.43	28.54 ± 0.63	28.52 ± 1.06	25.78 ± 0.37	28.52 ± 1.21
L1W	26.46 ± 0.39	56.64 ± 0.91	44.32 ± 0.66	28.65 ± 1.14	27.96 ± 0.72	25.73 ± 0.40	28.30 ± 0.99
L2W	25.71 ± 0.98	56.57 ± 0.22	44.87 ± 0.81	28.54 ± 0.30	27.79 ± 0.83	26.35 ± 0.54	28.02 ± 0.88
Dropout	26.38 ± 0.21	56.11 ± 0.83	44.78 ± 0.41	27.66 ± 0.51	28.43 ± 0.88	25.95 ± 0.57	27.69 ± 0.38
BN	31.97 ± 3.10	56.49 ± 0.32	43.75 ± 0.76	28.83 ± 0.47	28.20 ± 0.40	25.50 ± 0.55	28.38 ± 0.86
CR	24.96 ± 0.63	57.40 ± 2.11	45.16 ± 0.94	26.45 ± 0.22	28.65 ± 1.21	26.72 ± 0.61	27.94 ± 0.43
cw-CR	22.99 ± 0.58	53.50 ± 1.05	42.15 ± 0.64	26.40 ± 0.62	28.54 ± 1.01	25.93 ± 0.59	27.77 ± 0.88
VR	21.44 ± 0.88	53.90 ± 0.97	42.33 ± 0.57	**24.96 ± 0.26**	26.61 ± 0.47	25.01 ± 0.41	26.06 ± 0.72
cw-VR	21.58 ± 0.21	**51.93 ± 1.09**	43.00 ± 0.95	25.81 ± 0.64	**26.46 ± 0.25**	24.42 ± 0.31	26.19 ± 1.35
L1R	**20.63 ± 0.50**	52.39 ± 0.99	**40.92 ± 0.33**	25.49 ± 0.61	27.81 ± 0.43	25.13 ± 0.52	26.49 ± 0.96
RR	26.46 ± 0.25	57.09 ± 1.08	44.35 ± 1.09	26.58 ± 0.66	26.87 ± 0.58	**23.92 ± 0.37**	**25.80 ± 0.85**
cw-RR	26.29 ± 0.41	57.55 ± 0.46	44.71 ± 1.59	26.62 ± 0.77	27.12 ± 0.46	24.34 ± 0.27	26.10 ± 0.59
Best improvement	6.01	4.14	3.03	3.58	2.06	1.86	2.72

Table 5.3. Condition experiment results for MNIST MLP model. The best performing regularizer in each condition (each column) is shown in bold, and other regularizers whose performance range overlaps with that of the best ones are highlighted in gray. For the default condition, the standard values of data size=50k and layer width=100 were used, and Adam optimizer was applied. For other columns, all the conditions were the same as the default, except for the condition indicated on the top part of the column. Regularizers were applied to the last hidden layer.

Regularizer	Default	Data Size		Layer Width	
		1k	5k	2	8
Baseline	2.85 ± 0.11	11.41 ± 0.19	6.00 ± 0.07	31.62 ± 0.07	10.52 ± 0.57
L1W	2.85 ± 0.06	11.64 ± 0.27	5.96 ± 0.11	31.67 ± 0.15	11.02 ± 0.58
L2W	3.02 ± 0.40	11.38 ± 0.18	5.86 ± 0.10	31.66 ± 0.13	10.65 ± 0.23
Dropout	2.70 ± 0.08	**10.29 ± 0.23**	**5.59 ± 0.11**	62.09 ± 1.32	13.94 ± 1.05
BN	2.81 ± 0.12	10.81 ± 0.04	5.60 ± 0.10	42.08 ± 0.93	**7.51 ± 0.58**
CR	2.50 ± 0.05	11.63 ± 0.24	6.05 ± 0.06	34.80 ± 0.25	10.25 ± 0.74
cw-CR	2.49 ± 0.10	10.62 ± 0.05	5.80 ± 0.15	31.50 ± 0.11	10.81 ± 1.11
VR	2.65 ± 0.11	14.42 ± 0.14	6.90 ± 0.22	32.39 ± 0.13	9.22 ± 0.28
cw-VR	2.42 ± 0.06	10.44 ± 0.18	5.90 ± 0.12	**30.34 ± 0.06**	10.01 ± 0.63
L1R	**2.35 ± 0.08**	11.60 ± 0.20	6.20 ± 0.13	64.39 ± 0.26	88.65 ± 0.00
RR	2.81 ± 0.10	10.92 ± 0.17	6.61 ± 0.05	38.35 ± 0.20	12.31 ± 0.16
cw-RR	2.57 ± 0.08	10.89 ± 0.19	6.60 ± 0.17	38.57 ± 0.12	12.63 ± 0.39
Best improvement	0.5	1.12	0.41	1.28	3.01

Table 5.4. Condition experiment results for CIFAR-100 CNN model. The best performing regularizer in each condition (each column) is shown in bold, and other regularizers whose performance range overlaps with that of the best ones are highlighted in gray. For the default condition, the standard values of data size=50k, layer width=128, and the number of classes=100 were used. For other columns, all the conditions were the same as the default, except for the condition indicated on the top part of the column. Regularizers were applied to the last hidden layer.

Regularizer	Default	Data Size		Layer Width		Number of Classes	
		1k	5k	32	512	16	64
Baseline	61.26 ± 0.52	90.89 ± 0.30	82.21 ± 0.72	62.41 ± 0.34	61.30 ± 0.64	45.75 ± 0.73	58.02 ± 0.40
L1W	60.97 ± 0.64	91.33 ± 0.37	82.3 ± 0.6	62.23 ± 0.58	60.92 ± 0.47	45.08 ± 1.53	58.08 ± 1.18
L2W	60.23 ± 0.31	90.53 ± 0.39	82.05 ± 0.70	62.78 ± 0.36	61.55 ± 0.99	45.28 ± 1.59	57.47 ± 0.66
Dropout	63.88 ± 0.72	$\mathbf{90.22 \pm 0.48}$	81.68 ± 0.81	64.08 ± 0.99	64.31 ± 0.37	45.73 ± 1.57	59.14 ± 0.46
BN	60.93 ± 0.39	91.18 ± 0.36	82.01 ± 0.58	62.18 ± 1.49	62.16 ± 0.57	44.55 ± 1.43	57.72 ± 0.66
CR	59.88 ± 0.50	91.70 ± 0.14	82.47 ± 0.41	$\mathbf{60.47 \pm 0.63}$	60.70 ± 0.94	44.55 ± 1.10	56.76 ± 0.86
cw-CR	57.03 ± 0.73	90.85 ± 0.29	81.29 ± 0.62	61.41 ± 0.67	58.02 ± 0.25	43.50 ± 1.21	54.24 ± 0.64
VR	57.68 ± 0.94	91.43 ± 0.32	81.85 ± 0.38	61.35 ± 0.45	56.87 ± 0.74	42.33 ± 1.03	54.32 ± 0.40
cw-VR	56.75 ± 0.64	90.45 ± 0.22	$\mathbf{81.03 \pm 0.57}$	60.67 ± 0.59	56.91 ± 0.73	$\mathbf{41.38 \pm 0.53}$	54.23 ± 1.06
L1R	$\mathbf{56.03 \pm 0.81}$	91.15 ± 0.35	81.98 ± 0.47	61.11 ± 0.31	$\mathbf{56.46 \pm 0.62}$	42.51 ± 1.43	$\mathbf{53.65 \pm 1.00}$
RR	62.68 ± 0.35	91.20 ± 0.27	81.32 ± 0.36	68.54 ± 0.46	59.29 ± 0.32	44.16 ± 0.80	60.25 ± 0.35
cw-RR	62.62 ± 0.31	90.62 ± 0.34	81.57 ± 0.14	68.11 ± 0.31	59.15 ± 0.29	44.10 ± 0.65	60.03 ± 0.41
Best improvement	5.23	0.67	1.18	1.94	4.84	4.37	4.37

and elucidating common network compression methods.

5.2.1. The Need for Network Compression

Over the past decade, deep learning has been a success in several applications, including computer vision and speech. A variety of advanced optimization methods and ReLU activation function enabled the training of deep networks, and cross-entropy loss function improved the performance of deep learning classification task. A few popular techniques such as batch normalization and dropout further enhanced the performance of deep learning. In recent years, deep learning has been widely used in practical applications based on the success in research. Especially, with the proliferation of mobile devices, many deep learning applications are deployed and operated on mobile devices. Mobile devices, of course, are not limited to mobile phones. Smart glasses and smartwatches have become popular as well. In addition to mobile devices, drones and small robots have also been equipped with deep learning applications. Probably, the field in which network compression is most useful is a military application. This is because military devices usually have to be made very small or made into low power.

The interface of deep learning application is often installed on mobile devices due to memory and computation power limitations, and the computing part of the application runs in a cloud server. This architecture potentially can cause a few problems. First, network delay between a mobile device and the cloud server could occur. In particular, if real-time processing is considered important as in a self-driving car, this drawback must be eliminated. Second, a privacy issue may befall. For example, an image, a popular data format for deep learning, often contains much private or security information. To mitigate the problems, installing a deep learning application on a mobile device can be considered. However, the size or power consumption of the application may be too big or high to be installed in a mobile device. The number of weight parameters in a deep network often exceeds the memory capacity of a mobile device. Even if mobile storage is enough to memorize all the parameters, the power consumption for accessing the parameters can be very high, which should be prohibited. Therefore, it is necessary to reduce the network size by decreasing the number of network parameters or restricting parameter precision.

5.2.2. Three Typical Approaches for Network Compression

Network compression methods are roughly categorized into three: parameter pruning, low-rank matrix decomposition, and parameter quantization. We briefly describe each and present related works. First, parameter pruning is to treat individual unit or weight and simply prune unimportant units or weights that meet certain criteria such as those with small second order derivative values (Hassibi and Stork 1993) and weight parameters (Han et al. 2015). A method proposed by Denil et al. (2013) is somewhat different. They use the structure of learned weight parameters. More specifically, they randomly pick some of the weights and predict the other weights from the chosen ones. Obviously, there are many more possible criteria that can be used to cut out units or weights. Originally, this approach was proposed to prevent a neural network from overfitting and reduce model complexity (LeCun, Denker, and Solla 1990; Hassibi and Stork 1993). However, as explained in Chapter 2, overfitting rarely occurs in deep networks, so recent related works mainly used this approach for network compression. Second, the low-rank matrix decomposition method considers all the weights in a layer together. A single weight matrix of a layer can be decomposed into two matrices by applying Singular Value Decomposition (SVD) (Denton et al. 2014). This method reduces the number of weights by ignoring unimportant dimensions with small singular values. Like the pruning method, this method might lead to better generalization due to the reduced number of weight parameters. In this work, we focus on this approach which we further detail in the next subsection. Finally, parameter quantization is to make weights have low precision values by quantizing or binarizing weight parameters (Han, Mao, and Dally 2016; Gupta et al. 2015; Gong et al. 2015; Courbariaux et al. 2016). This approach is somewhat orthogonal to the other two approaches because quantization can be combined with the two approaches. In practice, this approach is a bit trickier to implement than the other two approaches. It is necessary to implement a logic for writing and reading indices to memorize weight clusters and encode/decode weights.

5.2.3. Proposed Approaches and Experimental Results

Among the three approaches explained in the previous subsection, we focus on low-rank matrix decomposition using SVD, which is one of the most popular strategies. Proposed representation regularizer in this work can be used to enhance network compression in two ways when an SVD approach is considered.

Network Compression Using cw-VR

Representation regularizers are used to make a compact representation, reducing the rank of the corresponding weight matrix. This method can be used to extend Domain Adaptive Low Rank (DALR) of Masana et al. (2017) that increases the compression rate in consideration of the activation distribution. The compact representation introduced by the representation regularizers may result in less performance degradation at a higher compression ratio than that of a deep network without any regularization (baseline).

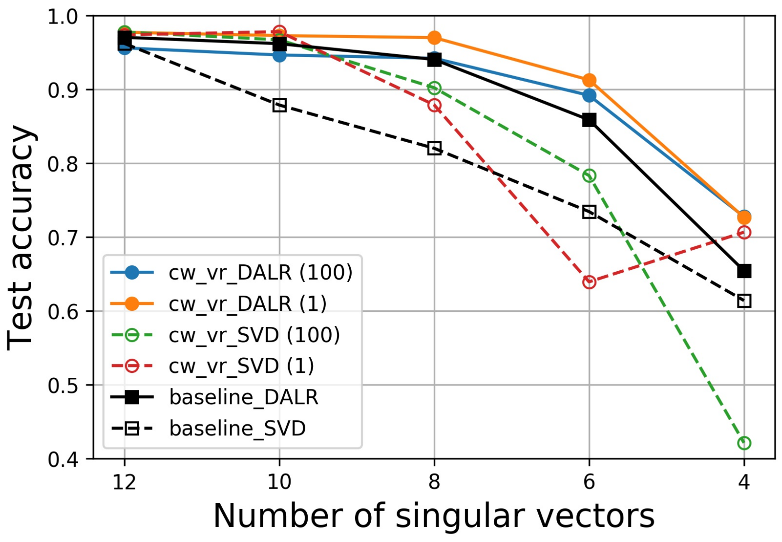
We performed network compression experiments with cw-VR and compared a few compression methods with different conditions. We used Lenet-5 network (LeCun et al. 1998) using MNIST digit dataset and applied cw-VR to the two fully-connected layers (named FC1 and FC2) only. Table 5.1 shows overall results. The first observation is that DALR outperforms SVD, This is an expected outcome considering that DALR considers the distribution of activations. Secondly, representation is better compressible when cw-VR is applied than when no regularizers are used, which is in line with our initial conjecture. Finally, loss weight of regularizers (numbers in the parentheses) does not provide any consistency. We believe that further investigation is needed to uncover the relationship between the compactness of representations and compressibility of the corresponding weight matrix.

An approach using IONs can be used as well. By applying whitening Q matrix, we can use only a small number of dimensions for compressing a network. This approach may have an effect similar to that of the SVD approach. As future work, we intend to compress deep networks using other regularizers and the approach using IONs. It would be interesting to observe how different representation characteristics affect compression ratio.

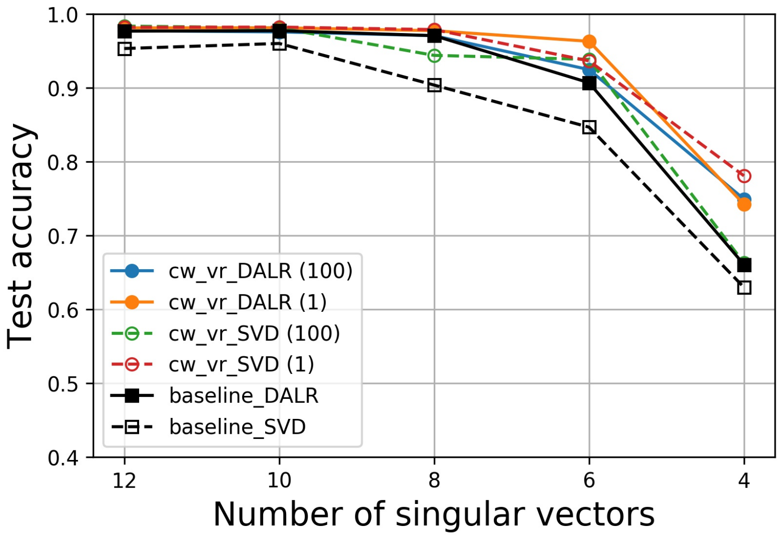
Network Compression Using RR

We can directly compress weight matrices without resorting to representations by applying Rank Regularizer (RR) to weight matrices. It can be possible to compress a network at a higher rate when a weight matrix has a lower rank. That is, it is beneficial if a network is trained to have low-rank weight matrices without performance loss. Recently, this idea has been implemented by Alvarez and Salzmann (2017). They developed a regularizer having similar effects to RR, applied it to a deep network, and showed promising results. However, their method has a drawback. Their method has two stages meaning that they train a network using their regularizer in every training step, but solve convex optimization and update weight matrices at every epoch. RR has an advantage against the method of Alvarez and Salzmann (2017) in that RR does not require two-stage optimization. RR is just added to and affects the original objective function in every mini-batch. However, the comparison of RR's performance and training speed with those of the method of Alvarez and Salzmann (2017) is absent, which we leave out for future work.

We conducted network compression experiments by applying RR with the same setting as that of cw-VR experiments above. Table 5.2 shows similar results for performance comparison between SVD and DALR. However, it seems that RR is less effective than cw-VR. We guess that the two fully-connected layers are already compressed well without RR because they are upper layers of a deep network. As future work, it would



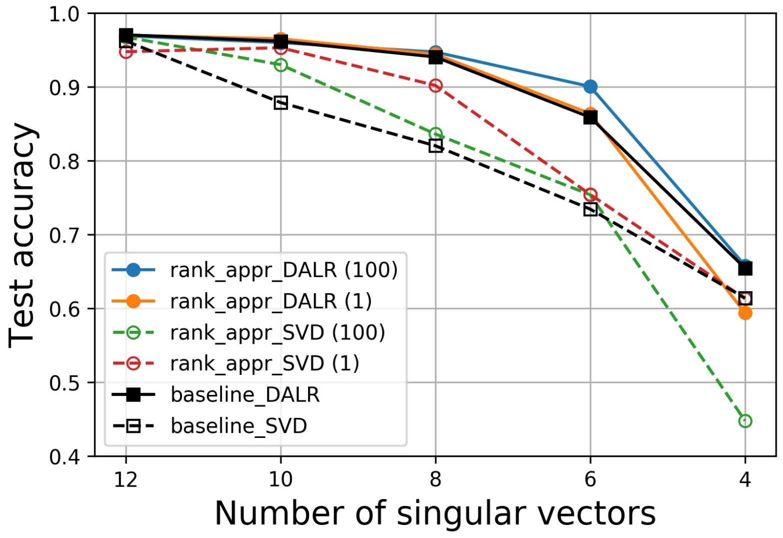
(a) FC1



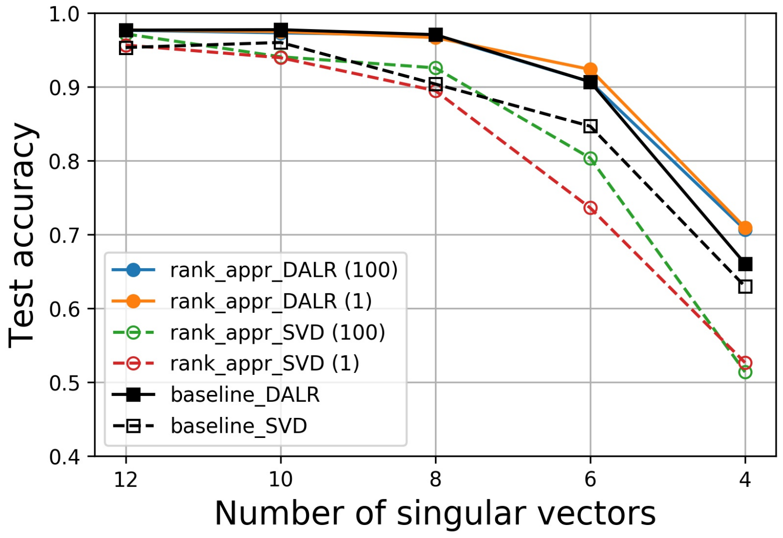
(b) FC2

Figure 5.1. The sensitivity of different compression methods on the number of remaining singular values. Lenet-5 network and MNIST dataset were used. cw-VR were applied to the first (a) and second (b) fully-connected layers' activations, and corresponding two weight matrices were compressed using DALR and SVD. The numbers in parentheses are loss weights (λ) that balance between a cross-entropy loss and a penalty loss.

be interesting to apply RR to lower layers and compress the network to see whether doing so is effective.



(a) FC1



(b) FC2

Figure 5.2. The sensitivity of different compression methods on the number of remaining singular values. Lenet-5 network and MNIST dataset were used. RR were applied to the first (a) and second (b) fully-connected layers' weight matrices, and the matrices were compressed using DALR and SVD. The numbers in parentheses are loss weights (λ) that balance between a cross-entropy loss and a penalty loss.

Chapter 6. Discussion

In this chapter, we first discuss the theoretical and practical implications of this work and then address a few limitations of this work. We conclude this chapter by discussing future works.

6.1. Implication

Though a large part of this work is empirical, experimental results provide a few theoretical implications such as how and why utilizing class information for regularization is useful for a classification task and how penalty representation regularizers work differently from dropout and batch normalization. We discuss these implications and also address the theoretical implications of ION. The latter part of this section considers practical implications. In practice, regularization methods using representations are less popular than those using weight parameters such as L1/L2 regularization and weight sharing (LeCun 1989). We examine if representation regularizers can be efficient and useful tools for performance tuning. Also, a few benefits and drawbacks of various representation characteristics are discussed.

6.1.1. Usefulness of Class Information

In Chapter 3 and Chapter 5, it is shown that utilizing class information for classification is beneficial for performance improvement. There are two possible explanations for why the use of class label information is advantageous in terms of the principles of representation learning: feature compactness, and independence.

First, reducing variances of hidden activations per class leads to intra-class compactness of learned representations. Class-wise Variance Regularizer (cw-VR) enforces deep representations to be similar to each other in the same class samples. Overfitting can be avoided because non-class relevant information may be removed in order for each sample to become close to its corresponding class center. By reducing intra-class variance, inter-class distances can be implicitly maximized as well. As a result, classes can become more separable, which is advantageous for a classification task. Feature compactness is somewhat related to the invariance of learned representations to nuisance factors (Achille and Soatto 2018a). When compact representations are formed, other information other than shared information among same class samples can be ignored so that the model can be robust to nuisance. However, forcing too small intra-class variances by choosing large loss weight values may lead to a lower performance because cross-entropy loss cannot be reduced. We observed that there is a tradeoff between performance and feature compactness.

Second, decreasing cross-covariances of hidden activations per class leads to independent representations. It is known that decorrelated models for ensemble (Hansen and Salamon 1990; Perrone and Cooper 1992) or bagging (Breiman 1996) can help improve performance. Likewise, decorrelating hidden representations are often considered in deep learning (Srivastava et al. 2014; Cogswell et al. 2016). Srivastava et al. (2014) reduce co-adaptation by zeroing hidden unit activations to prevent a model from overfitting. Cogswell et al. (2016) explicitly encourage decorrelated representations by reducing cross-covariance of hidden activations. However, the two works do not consider using class information. Different class samples probably have different features, so, intuitively, class-wise independence of features can help enhance the classification capability of a deep network. In Chapter 3, we have shown that class-wise Covariance Regularizer (cw-CR) often outperforms Covariance Regularizer (CR). It turns out that decorrelated representations have some relationship with disentangled representations, which is one of the main principles of representation learning. However, note that decorrelating here only considers 'linear' relationships, and 'disengagement' has not been defined rigorously and mathematically in deep learning (Higgins et al. 2018).

6.1.2. Comparison with Non-penalty Regularizers: Dropout and Batch Normalization

Dropout (Srivastava et al. 2014) and batch normalization (Ioffe and Szegedy 2015) are very popular regularizers in practice (Batch normalization is more often considered as an optimization technique). They are fundamentally different from representation regularizers studied in this work because they are not 'penalty cost function' regularizers. Instead, they are implemented by directly affecting the feedforward calculations.

Dropout has been shown to have effects similar to those of ensemble and data-augmentation through its noisy training procedure with benefits unobtainable from a penalty regularizer. On the other hand, there is a common belief that 'dropout reduces co-adaptation (or pair-wise 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 found that dropout does not actually 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 (Recently, Santurkar et al. (2018) argued that the performance improvement by batch normalization does not stem from reducing internal covariate shift). 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.

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 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.

6.1.3. Identical Output Network

In Chapter 4, we prove that infinitely many IONs for linear networks exist. We also provide the theorem of IONs for ReLU layers and a few possible extensions. The existence of IONs implies that there may be no apparent reason why specific statistical characteristics of representation should be desired. This implication is intensively discussed in Chapter 4 with some examples of characteristics such as correlation and sparsity.

We can consider the implication of IONs from the viewpoint of weight parameter space as well. Different characteristics of representations mean that corresponding weight parameters to the characteristics are different from each other as well. Assume that we train a deep network by applying Variational Network Quantization (VNQ), a Bayesian compression method using a multi-modal quantizing prior of weights (Achterhold et al. 2018). The resulting weights can be well clustered around target clustering values. Obviously, IONs can be considered for this network if the layer is linear, and we can easily obtain a network whose weights are not well grouped but whose output is identical to that of the original network. We guess that it might be possible to obtain better compressible weights just by applying proper Q matrices for linear layers.

6.1.4. Using Representation Regularizers for Performance Tuning

Deep learning practitioners and researchers usually consider L2 weight decay as the first option for tuning the performance of a deep network. This is because L2 weight decay is easy to use and often provides a considerable amount of performance improvement. On the contrary, as presented in Chapter 3, representation regularizers are less popular. There are three possible reasons why L2 weight decay is popular, but representation regularizers are not. One apparent reason is that most software packages provide L2 weight decay as a built-in function, so it is super easy to apply. Another possible reason is that practitioners and researchers may believe that L2 weight decay controls the effective capacity of a deep network; thus performance is improved. However, Zhang et al. (2016) showed that a typical deep network has a large enough capacity to memorize the entire training samples and that the L2 weight regularization method does not sufficiently reduce the network capacity. Finally, the effect of representation regularization on performance improvement has not been thoroughly investigated yet. Only limited test cases have been provided in previous works, so for deep learning practitioners and researchers, it might be too expensive to apply new regularizers to their problems.

In Chapter 3, we showed that like L2 weight decay, cw-VR is conceptually simple, computationally light, and easy to use. Besides, it is useful to improve classification performance. For these reasons, we believe that cw-VR is a compatible option of L2 weight decay. Also, we show that representation regularizers are very useful for performance tuning when they are used as a set. Therefore, if the regularizers are implemented together in a single software library, they can be considered as a powerful tuning tool. Code of representation regularizers used in this work is made available online, so we hope that many practitioners and researchers benefit from it.

6.1.5. Benefits and Drawbacks of Different Statistical Characteristics of Representations

In Chapter 4, the theorem of ION indicates that different statistical characteristics of deep representations lead to the same output. Although the outputs are identical, there can exist other benefits and drawbacks of different characteristics. In this subsection, we explore a few benefits and drawbacks of the characteristics concerning training speed, interpretation of features, and network compression.

First, all the IONs of an original network have the same training speed, which is because IONs are not formed by actual training of deep networks. Instead, they are made by simple matrix calculations after the training of the original network ends. However, forming specific representation characteristics by using representation regularizers, not by IONs usually requires more calculation than training without regularizers (baseline), so training deep networks using regularizers is usually slower than that with the baseline. In our experience, the training speed of all the regularizers we developed is comparable to the baseline because the regularizers are computationally light. In the case of class-wise regularizers, when the number of classes is too large, training speed tends to become significantly slower. However, we think that this can be optimized if we choose the same number of samples per class and sort them in the order of class in each mini-batch such that deep learning software can benefit from the structure of mini-batch samples.

Second, a particular representation characteristic might give better feature interpretation than the others. For example, decorrelated features by using whitening Q matrix might be more explainable than the others. Visualization of features captured by hidden units may confirm this guess. We discuss this more in Section 6.3.

Finally, a deep network with a particular representation characteristic can be better compressed than networks with other characteristics. By using RR (Rank Regularizer) and cw-RR (class-wise Rank Regularizer), one can pursue correlated representations that have a potential of high compression rate. If a network with highly correlated representations has a comparable performance to those other networks with less correlated representations, then the network can be better compressed by applying the whitening Q matrix and removing unimportant hidden units. By the theorem of ION, one can find a proper Q matrix to align the null spaces of $\mathbf{C}_l$ to some of the hidden units. This means that the hidden units might be removed with minimal performance degradation. However, like the training time mentioned above, all the IONs can be compressed with the same compression rate because they have the same rank. Note that different representation characteristics lead to different compression rates only when representation regularizers are applied during network training not when affine transformations are applied after the training phase.

6.2. Limitation

Even though our experimental results confirm that representation regularization is useful for performance improvement, and theoretical explanation on statistical characteristics of deep network representations is provided in the previous chapters, it is still unclear how representation regularizers work. Also, most of the experimentations are classification tasks that are relatively easier than other tasks such as regression and reconstruction. Datasets are restricted to images as well. In this subsection, we discuss the main limitations of this work that lies in lack of theoretical proofs and limited experimentation.

6.2.1. Understanding the Underlying Mechanism of Representation Regularization

Our main concern about the design of representation regularizers is that the behavior and underlying principle of representation regularizers are not fully understood at this point. Such a regularization essentially enforces a loss on hidden unit activations so links to a supervised metric learning scheme. However, Variance Regularizer (VR) often achieves better performance than the baseline, which is counter-intuitive for classification tasks because VR encourages to have similar representations for different class samples. Two possible explanations are presented in Chapter 3, and the usefulness of class information was discussed in the previous section. In this subsection, we consider the viewpoint of deep network optimization that is non-convex.

In practice, deep learning practitioners and researchers may pursue specific statistical characteristics of deep network representation not only for performance improvement itself based on priors but also optimization and conditioning of $\mathbf{z}$. Also, representation regularization can add stability to the numerics. As summarized in

Chapter 5, the conditioning affects the optimization process and leads to different local minima thereby achieving high or low performances. To sum up, the effect of regularizers on optimization would affect performance. When multiple regularizers are considered, the effect can be addressed as 'tuning effect' because there were no systematic patterns found.

6.2.2. Manipulating Representation Characteristics other than Covariance and Variance for ReLU Networks

As shown in Theorem 2, Q matrices for ReLU's IONs are limited to (permutated) positive diagonal matrices. Even though we provide possible extensions for ReLU's IONs, this limitation is quite problematic due to the following reason. By applying the matrices of ReLU IONs, only covariance and variance of representations can be manipulated. Since these characteristics can be affected by the scale of activations, it is less meaningful to explain the relationship between two hidden units and the distribution of activations in a single unit. On the contrary, correlation and sparsity cannot be altered by applying ReLU's IONs. These two characteristics have been considered to be important for explaining feature independence and compactness.

Comparable Performance Networks (CPNs) can be considered to overcome this problem. Correlation and sparsity can be altered by ReLU's CPNs with a small performance loss. We guess that CPNs might be only applicable to relatively simple classification tasks like MNIST digit classification. When more complicated tasks are considered, CPNs may not exist or are difficult to be found with a simple brute-force search.

6.2.3. Investigating Representation Characteristics of Complicated Tasks

A sufficient amount of experimental and qualitative analysis was provided to verify the effectiveness of our new regularizers and the relationship between representation characteristics and performance. However, in this work, we apply representation regularizers mainly to deep networks for image classification tasks. Obviously, representation regularizers can be applied to deep networks for other types of datasets and tasks as well.

First, regularizers can be applied to image reconstruction tasks. It is known that image reconstruction tasks may have entirely different representation characteristics from those of image classification tasks. Their objective function is different, so representation preserves or removes different information from the input. For example, an image classification network removes information not related to the label, so their representations may be more compact than those of deep networks for image reconstruction.

Second, we can apply representation regularizers to generative models. Precisely, a discriminator of Generative Adversarial Network (GAN) (Goodfellow et al. 2014) can be regularized with representation regularizers. The way we apply representation regularizers to GANs is the same as what we do for classification networks, but we use different label information that is 'real' or 'fake' rather than class information given by datasets. Since the discriminator should be trained to classify real and fake samples well, it is beneficial for the discriminator to have distinct learned representations per label that is real or fake. We can use cw-VR to reduce the variance of the hidden unit activations within the same label samples. As a result of applying this regularizer, a generator of GAN may contribute to producing trickier samples to deceive the discriminator. Intuitively, applying class-wise regularizers to the discriminator is not expected to be a great help because class information is not used for discriminator's loss. Nonetheless, applying class-wise regularizers is possible, and it would be interesting to look at experimental results. Utilizing class information in a latent variable may have a positive effect. Other all-class representation regularizers can be applied to the discriminator as well.

Finally, representation regularizers can also be applied to meta-learning. For example, Gidaris and Komodakis (2018) developed a 'few-shot classification weight generator' as a meta-learner component. The weight generator produces a weight vector for a new class by combining weight vectors of the classes used in training. Therefore, if we have a distinct weight vector per class by applying class-wise regularizers, we may not only improve performance but also be able to generate distinct weights for the few shot classes that are very different from other classes. Class-wise regularizers, of course, can be applied to recent promising meta-learning algorithms (Finn, Abbeel, and Levine 2017; Mishra et al. 2018).

Besides the types of machine learning tasks, different input data types such as structured images, text, and audio also can be considered for representation regularization.

6.3. Possible Future Work

In this section, we present three possible future works to extend this work. Interpreting learned representations via visualization to understand the effects of representation regularization is explained. Use of mutual information for regularization is discussed as well. We finally address how to enhance network compression by manipulating deep representations.

6.3.1. Interpreting Learned Representations via Visualization

Deep networks are known to capture task-relevant information from data automatically. In the case of image data, what information is extracted is often verified through learned feature visualization. In some previous studies (Cogswell et al. 2016; Srivastava et al. 2014), learned features were visualized, and the visualization confirmed that features learned by applying their regularizers are different from those of networks without any regularizers (baseline). Srivastava et al. (2014) visualized features learned by hidden units of the first layer of 2-layer autoencoder. Their results show that each hidden unit detects distinct parts of the MNIST image, so they argue that the visualization of the learned features is evidence that co-adaptations are broken up by applying dropout. Cogswell et al. (2016) conducted the same experiment and showed that DeCov learns different features from dropout and the baseline. However, both studies confirmed only the features of shallow networks, and there was no verification of the upper layer features of deep networks. In Chapter 3, we show that regularizers are more effective when applied to the upper layers. Therefore, we need to visualize the features captured by hidden units in the upper layer of deep networks.

It is shown that representation regularizers proposed in this work manipulate representation characteristics distinctly. Therefore, the features learned by each regularizer can be expected to be different from each other. In particular, the features learned by class-wise regularizers may be significantly different from those learned by all-class regularizers. RR and cw-RR make representations compact and independent, so it would be interesting to visualize the features they learn. Also, it might be a surprising result if deep networks with similar performance detect distinct features or deep networks having an entirely different performance detect very similar features to each other. This is because it is beyond the common belief that the task performance of deep networks can be dependent on learned features.

6.3.2. Designing a Regularizer Utilizing Mutual Information

As described in Chapter 4, the Information Bottleneck Lagrangian $\mathcal{L}(p(\mathbf{z}_l \mid \mathbf{x})) = H(\mathbf{y} \mid \mathbf{z}_l) + \beta I(\mathbf{z}_l; \mathbf{x})$ can be considered as the combination of cross-entropy and mutual information between the input and $\mathbf{z}_l$. Our experimental results show that $I(\mathbf{z}_l; \mathbf{x})$ can be indirectly controlled by representation regularizers, and each regularizer has a different degree of controlling $I(\mathbf{z}_l; \mathbf{x})$. Also, it is observed that $I(\mathbf{z}_l; \mathbf{x})$ and generalization performance have a strong correlation. Based on the result, we can design a regularizer that explicitly reduces $I(\mathbf{z}_l; \mathbf{x})$ using the methods explained in Chapter 4. A mutual information regularizer may perform better than other regularizers and can be a useful tool to study the effects of regularizers from an information-theoretic viewpoint.

6.3.3. Applying Multiple Representation Regularizers to a Network

In Chapter 3 and Chapter 4, it confirms that each representation regularizer distinctly shapes deep network representations and that no statistical characteristics consistently outperform than those of the others. Therefore, we can guess that the combination of multiple representation regularizers may give a promising performance depending on the task. For example, using cw-VR and L1R together could possibly maximize the benefits of each by simultaneously making representation more compact and sparser. However, as observed in Chapter 5, in many cases we do not know which representation characteristics are suitable for a given task. In other words, since there is no systematic pattern for improving performance using the regularizers, it might be better to apply multiple regularizers together to a network for determining their proportions. It is also possible to learn such proportions for the regularizers in some combination they form. In order to find high performing regularizer combinations and their loss weights, one possible sophisticated way is to use a hyperparameter optimization framework like Bayesian optimization. First, we briefly explain what Hyperparameter Optimization (HPO) is and why it is needed for deep networks. Then, we discuss how to apply HPO for regularizer combinations.

For traditional machine learning, feature engineering is usually necessary for achieving high performance. However, feature engineering is typically labor-intensive and can be performed well only by experts. In contrast, deep networks can automatically capture task-relevant information in the data and significantly reduce the effort needed for feature engineering. As a result, deep learning has achieved promising results for many applications in recent years. Deep learning makes less use of feature engineering for improving task performance but requires careful HPO instead. This is because deep networks usually have many more hyperparameters than traditional machine learning algorithms, making the task more sophisticated to approach with feature engineering alone and because its performance can be highly dependent on the choice of hyperparameters.

HPO of deep networks can be considered as a problem of finding an optimal hyperparameter configuration of DNN performance which can be regarded as a black-box function. In the real world, what practitioners typically do is to repeat the steps of choosing a hyperparameter configuration, training the network, and evaluating the configuration. That is, based on the result of the previous evaluation, selection and evaluation process are repeated within a given amount of time budget or until a predetermined objective value is obtained. However, manually carrying out this process is time- and effort-consuming, and doing it well requires in-depth comprehension of deep learning which may be more difficult than learning feature engineering techniques.

To overcome such difficulty, practitioners often use simple grid or random search as the default automated HPO method (Bergstra and Bengio 2012). As a more sophisticated solution, one can use Bayesian Optimization (BO), a sequential model-based optimization method. BO has been successfully applied to many traditional machine learning and deep learning tasks. Spearmint (Snoek, Larochelle, and Adams 2012) using Gaussian Process (GP), Sequential Model-Based Algorithm Configuration (SMAC) (Hutter, Hoos, and Leyton-Brown 2011) using random forests (RF), and Tree-structured Parzen Estimator (TPE) (Bergstra, Yamins, and Cox 2013) using non-parametric density estimation are widely used in practice.

Obviously, regularizers can be one of the hyperparameters for improving deep network performance in a BO framework. We can define a set of regularizers and the range of loss weights such that a BO algorithm sequentially searches the best combination of regularizers with proper loss weights. We believe that this approach can be quite useful for improving performance even though the interpretability of regularization mechanism would become difficult.

6.3.4. Enhancing Deep Network Compression via Representation Manipulation

In Chapter 5, we show that compressing network using cw-VR and RR is useful. However, the experiments are limited to a relatively small network and simple dataset. Also, further investigation into compressing convolutional layers is necessary. Another direction is to investigate other representation characteristics. As observed in Chapter 3 and Chapter 4, each representation regularizer does encourage distinct representation characteristic, so it would be interesting to compare compression performances of different representation characteristics. Finally, a theoretical analysis of the relationship between representation compactness (or other properties) and the rank of a weight matrix is required to design better compression methods.

Chapter 7. Conclusion

This dissertation aimed to find the statistical characteristics of deep representations that high-performing networks commonly have and to better understand the relationship between the characteristics and the performance of deep networks. To this end, we introduced a few regularizers that manipulate statistical characteristics of deep representations and showed the impact of applying the regularizers on performance and representation. In doing so, we broadened the understanding of deep network regularization using representation shaping.

In Chapter 3, 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 useful, and they should be considered as essential and high-potential configurations for learning of deep networks. In Chapter 4, we have studied the most popular statistical characteristics and information-theoretic characteristics of deep representations. All the statistical characteristics that were studied failed to show any general or causal pattern for improving performance. Empirical results consistently showed that none of the studied statistical characteristics is a requirement for achieving good performance. In contrast to the statistical characteristics, information-theoretic characteristic $I(\mathbf{z};\mathbf{x})$ showed a strong correlation with the performance of a classification task. In Chapter 5, as practical ways of using representation regularizers, we have tried applying twelve different regularizers over many classification tasks with different task conditions. The results show that still no systematic pattern can be found, but the set of regularizers can be used as a very compelling tool for tuning performance. Also, we have investigated the usefulness of representation regularizers for network compression. Our results indicate that manipulating representations with representation regularizers is quite useful to enhance compression ratio with a smaller trade-off with performance.

In conclusion, the most important contribution of this work is probably to provide early work on developing rigorous and general theories and methodologies that can be used to understand the learned representations better.

Bibliography

Achille, A., and Soatto, S. 2018a. Emergence of invariance and disentangling in deep representations. *Journal of Machine Learning Research* 19(50):1–34.

Achille, A., and Soatto, S. 2018b. Information dropout: Learning optimal representations through noisy computation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Achterhold, J.; Koehler, J. M.; Schmeink, A.; and Genewein, T. 2018. Variational network quantization. *International Conference on Learning Representations*.

Alvarez, J. M., and Salzmann, M. 2017. Compression-aware training of deep networks. In *Advances in Neural Information Processing Systems*, 856–867.

Arpit, D.; Jastrzębski, S.; Ballas, N.; Krueger, D.; Bengio, E.; Kanwal, M. S.; Maharaj, T.; Fischer, A.; Courville, A.; Bengio, Y.; et al. 2017. A closer look at memorization in deep networks. *International Conference on Machine Learning*.

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

Belghazi, I.; Rajeswar, S.; Baratin, A.; Hjelm, R. D.; and Courville, A. 2018. Mine: mutual information neural estimation. *International Conference on Machine Learning*.

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

Bengio, Y.; Courville, A.; and Vincent, P. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence* 35(8):1798–1828.

Bergstra, J., and Bengio, Y. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research* 13(Feb):281–305.

Bergstra, J.; Yamins, D.; and Cox, D. D. 2013. Making a science of model search:

Hyperparameter optimization in hundreds of dimensions for vision architectures. *ICML (1)* 28:115–123.

Breiman, L. 1996. Bagging predictors. *Machine learning* 24(2):123–140.

Cheung, B.; Livezey, J. A.; Bansal, A. K.; and Olshausen, B. A. 2015. Discovering hidden factors of variation in deep networks. *Workshop Track of International Conference on Learning Representations*.

Choi, D., and Rhee, W. 2019. Utilizing class information for deep network representation shaping. *AAAI*.

Cogswell, M.; Ahmed, F.; Girshick, R.; Zitnick, L.; and Batra, D. 2016. Reducing overfitting in deep networks by decorrelating representations. *International Conference on Learning Representations*.

Courbariaux, M.; Hubara, I.; Soudry, D.; El-Yaniv, R.; and Bengio, Y. 2016. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*.

Denil, M.; Shakibi, B.; Dinh, L.; De Freitas, N.; et al. 2013. Predicting parameters in deep learning. In *Advances in neural information processing systems*, 2148–2156.

Denton, E. L.; Zaremba, W.; Bruna, J.; LeCun, Y.; and Fergus, R. 2014. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in neural information processing systems*, 1269–1277.

Dinh, L.; Pascanu, R.; Bengio, S.; and Bengio, Y. 2017. Sharp minima can generalize for deep nets. *International Conference on Machine Learning*.

Dziugaite, G. K., and Roy, D. M. 2017. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *Conference on Uncertainty in Artificial Intelligence*.

Eldan, R., and Shamir, O. 2016. The power of depth for feedforward neural networks. In *Conference on Learning Theory*, 907–940.

Farabet, C.; Couprie, C.; Najman, L.; and LeCun, Y. 2013. Learning hierarchical features for scene labeling. *IEEE transactions on pattern analysis and machine intelligence* 35(8):1915–1929.

Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. *Proceedings of the 30th international conference on machine learning (ICML-17)*.

Gidaris, S., and Komodakis, N. 2018. Dynamic few-shot visual learning without forgetting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4367–4375.

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.

Gong, Y.; Liu, L.; Yang, M.; and Bourdev, L. 2015. Compressing deep convolutional networks using vector quantization. *International Conference on Learning Representations*.

Goodfellow, I.; Bengio, Y.; and Courville, A. 2016. *Deep learning*. MIT press.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems*. 2672–2680.

Gupta, S.; Agrawal, A.; Gopalakrishnan, K.; and Narayanan, P. 2015. Deep learning with limited numerical precision. In *International Conference on Machine Learning*, 1737–1746.

Han, S.; Pool, J.; Tran, J.; and Dally, W. 2015. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, 1135–1143.

Han, S.; Mao, H.; and Dally, W. J. 2016. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *International Conference on Learning Representations*.

Hansen, L. K., and Salamon, P. 1990. Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence* 12(10):993–1001.

Hassibi, B., and Stork, D. G. 1993. Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in neural information processing systems*, 164–171.

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.

Higgins, I.; Amos, D.; Pfau, D.; Racaniere, S.; Matthey, L.; Rezende, D.; and Lerchner, A. 2018. Towards a definition of disentangled representations. *arXiv preprint arXiv:1812.02230*.

Hinton, G.; Deng, L.; Yu, D.; Dahl, G. E.; Mohamed, A.-r.; Jaitly, N.; Senior, A.; Vanhoucke, V.; Nguyen, P.; Sainath, T. N.; et al. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine* 29(6):82–97.

Hinton, G. E. 1986. Learning distributed representations of concepts. In *Proceedings of the eighth annual conference of the cognitive science society*, volume 1, 12. Amherst, MA.

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

Hoffer, E.; Hubara, I.; and Soudry, D. 2017. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. In *Advances in Neural Information Processing Systems*, 1731–1741.

Hutter, F.; Hoos, H. H.; and Leyton-Brown, K. 2011. Sequential model-based optimization for general algorithm configuration. In *International Conference on Learning and Intelligent Optimization*, 507–523. Springer.

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.

Jaderberg, M.; Vedaldi, A.; and Zisserman, A. 2014. Speeding up convolutional neural networks with low rank expansions. *arXiv preprint arXiv:1405.3866*.

Kolchinsky, A., and Tracey, B. D. 2017. Estimating mixture entropy with pairwise distances. *Entropy* 19(7):361.

Kolchinsky, A.; Tracey, B. D.; and Wolpert, D. H. 2017. Nonlinear information bottleneck. *arXiv preprint arXiv:1705.02436*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

Krueger, D.; Ballas, N.; Jastrzebski, S.; Arpit, D.; Kanwal, M. S.; Maharaj, T.; Bengio, E.; Fischer, A.; and Courville, A. 2017. Deep nets don't learn via memorization. *Workshop Track of International Conference on Learning Representations*.

Lebedev, V.; Ganin, Y.; Rakhuba, M.; Oseledets, I.; and Lempitsky, V. 2015. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. *International Conference on Learning Representations*.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

LeCun, Y.; Denker, J. S.; and Solla, S. A. 1990. Optimal brain damage. In *Advances in neural information processing systems*, 598–605.

LeCun, Y. 1989. Generalization and network design strategies. *Connectionism in perspective* 143–155.

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

Littwin, E., and Wolf, L. 2018. Regularizing by the variance of the activations' sample-variances. In *Advances in neural information processing systems*, 2119–2129.

Masana, M.; van de Weijer, J.; Herranz, L.; Bagdanov, A. D.; and MAlvarez, J. 2017. Domain-adaptive deep network compression. *network* 16:30.

Michalski, R. S.; Carbonell, J. G.; and Mitchell, T. M. 2013. *Machine learning: An artificial intelligence approach.* Springer Science & Business Media.

Mikolov, T.; Deoras, A.; Povey, D.; Burget, L.; and Černockỳ, J. 2011. Strategies for training large scale neural network language models. In *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*, 196–201. IEEE.

Mishra, N.; Rohaninejad, M.; Chen, X.; and Abbeel, P. 2018. A simple neural attentive meta-learner. *International Conference on Learning Representations*.

Montufar, G. F.; Pascanu, R.; Cho, K.; and Bengio, Y. 2014. On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*, 2924–2932.

Mood, A. M. 1950. Introduction to the theory of statistics.

Nakkiran, P.; Alvarez, R.; Prabhavalkar, R.; and Parada, C. 2015. Compressing deep neural networks using a rank-constrained topology. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Neyshabur, B.; Bhojanapalli, S.; McAllester, D.; and Srebro, N. 2017. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, 5947–5956.

Pascanu, R.; Montufar, G.; and Bengio, Y. 2014. On the number of response regions of deep feed forward networks with piece-wise linear activations. *International Conference on Learning Representations*.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Duchesnay, E. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Perrone, M. P., and Cooper, L. N. 1992. When networks disagree: Ensemble methods for hybrid neural networks. Technical report, BROWN UNIV PROVIDENCE RI INST FOR BRAIN AND NEURAL SYSTEMS.

Raghu, M.; Poole, B.; Kleinberg, J.; Ganguli, S.; and Sohl-Dickstein, J. 2017. On the expressive power of deep neural networks. *International Conference on Machine Learning*.

Sainath, T. N.; Mohamed, A.-r.; Kingsbury, B.; and Ramabhadran, B. 2013. Deep convolutional neural networks for lvcsr. In *Acoustics, speech and signal processing (ICASSP), 2013 IEEE international conference on*, 8614–8618. IEEE.

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.

Santurkar, S.; Tsipras, D.; Ilyas, A.; and Madry, A. 2018. How does batch normalization help optimization? In *Advances in Neural Information Processing Systems*, 2488–2498.

Saxe, A. M.; Bansal, Y.; Dapello, J.; Advani, M.; Kolchinsky, A.; Tracey, B. D.; and Cox, D. D. 2018. On the information bottleneck theory of deep learning.

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.

Shwartz-Ziv, R., and Tishby, N. 2017. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*.

Snoek, J.; Larochelle, H.; and Adams, R. P. 2012. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, 2951–2959.

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.

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1–9.

Tai, C.; Xiao, T.; Zhang, Y.; Wang, X.; et al. 2016. Convolutional neural networks with low-rank regularization. *International Conference on Learning Representations*.

Telgarsky, M. 2015. Representation benefits of deep feedforward networks. *arXiv preprint arXiv:1509.08101*.

Telgarsky, M. 2016. Benefits of depth in neural networks. *Conference on Learning Theory*.

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

Tishby, N., and Zaslavsky, N. 2015. Deep learning and the information bottleneck principle. In *Information Theory Workshop (ITW), 2015 IEEE*, 1–5. IEEE.

Tishby, N.; Pereira, F. C.; and Bialek, W. 1999. The information bottleneck method. *The 37th annual Allerton Conference on Communication, Control, and Computing*.

Tompson, J. J.; Jain, A.; LeCun, Y.; and Bregler, C. 2014. Joint training of a convolutional network and a graphical model for human pose estimation. In *Advances in neural information processing systems*, 1799–1807.

Vapnik, V. N. 1999. An overview of statistical learning theory. *IEEE transactions on neural networks* 10(5):988–999.

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.

Wolpert, D. H., and Macready, W. G. 1997. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation* 1(1):67–82.

Wu, L.; Zhu, Z.; and E, W. 2017. Towards understanding generalization of deep learning: Perspective of loss landscapes. *Workshop Track of International Conference on Machine Learning*.

Xiong, W.; Du, B.; Zhang, L.; Hu, R.; and Tao, D. 2016. Regularizing deep convolutional neural networks with a structured decorrelation constraint. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, 519–528. IEEE.

Xue, J.; Li, J.; Yu, D.; Seltzer, M.; and Gong, Y. 2014. Singular value decomposition based low-footprint speaker adaptation and personalization for deep neural network. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, 6359–6363. IEEE.

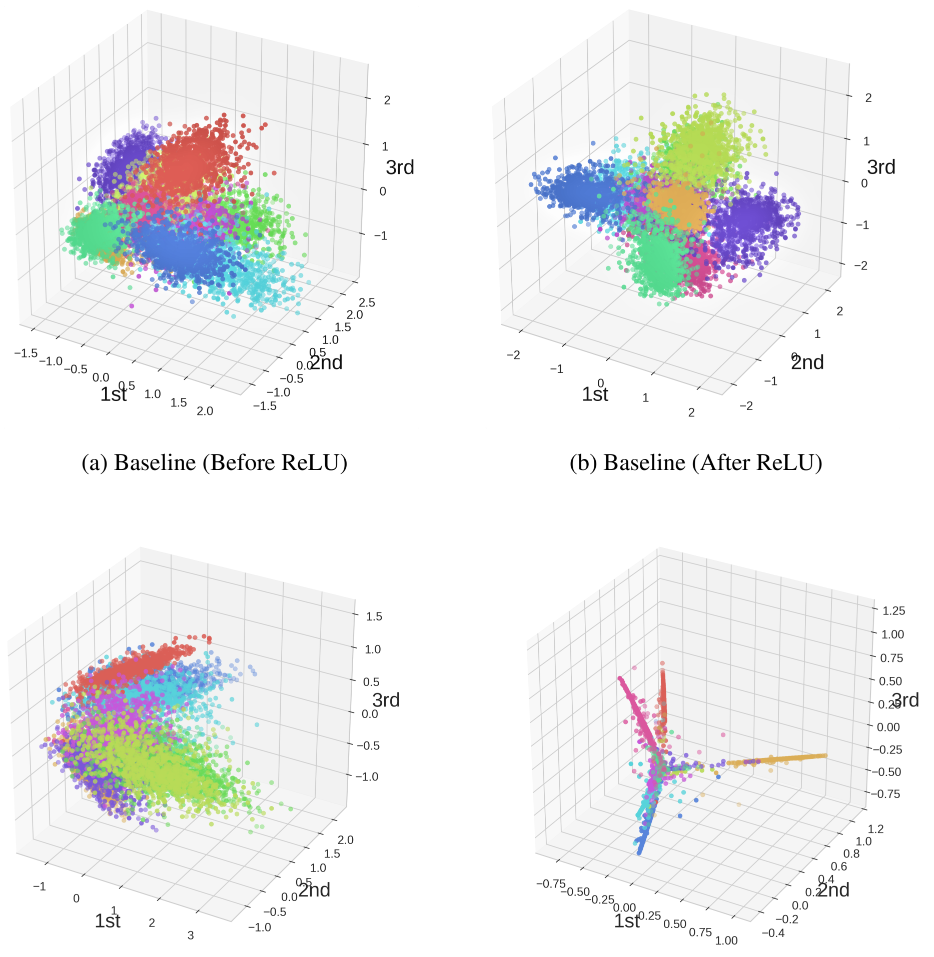
Xue, J.; Li, J.; and Gong, Y. 2013. Restructuring of deep neural network acoustic models with singular value decomposition. In *Interspeech*, 2365–2369.

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. 2017. Understanding deep learning requires rethinking generalization. *International Conference on Learning Representations*.

Appendix

A. Principal Component Analysis of Learned Representations



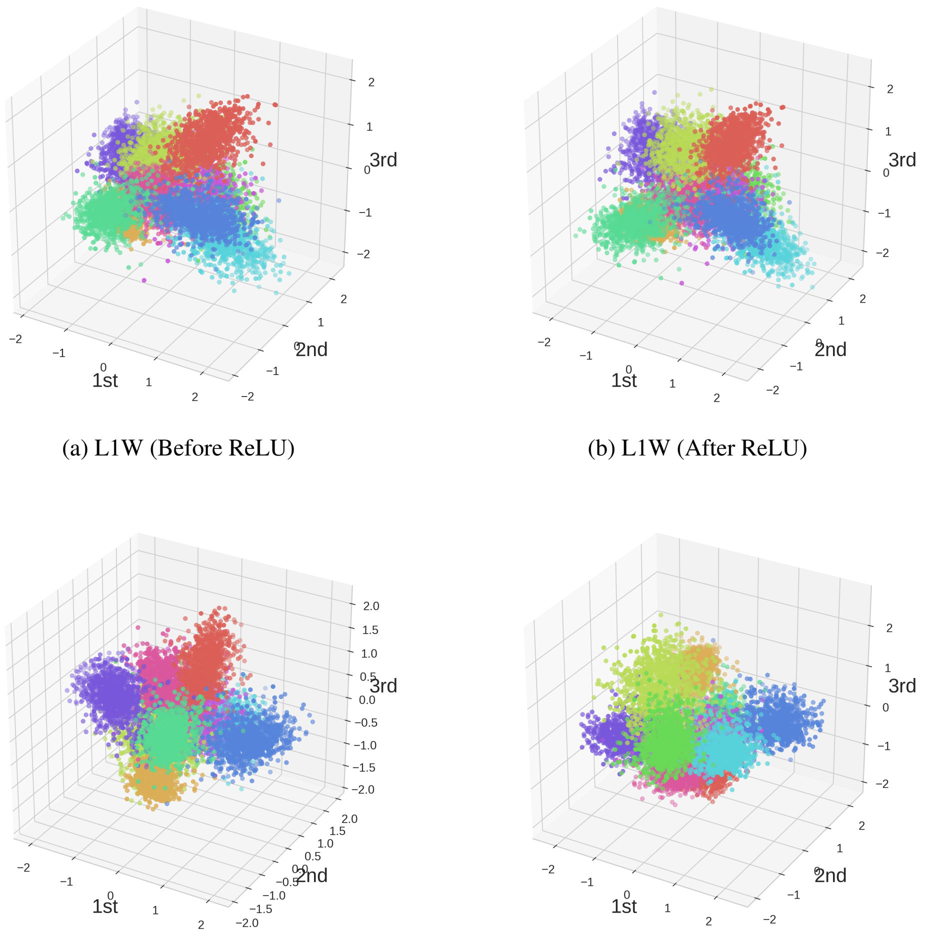
(a) Baseline (Before ReLU)

(b) Baseline (After ReLU)

(c) L1R (Before ReLU)

(d) L1R (After ReLU)

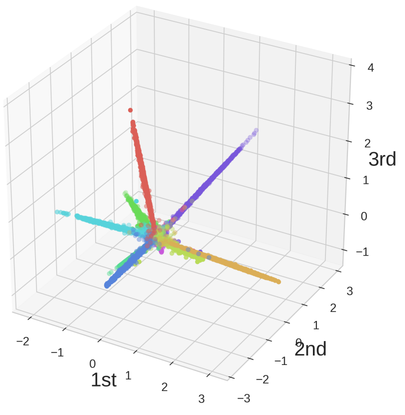
Figure A.1. The top three principal components of learned representations (Baseline, L1R).



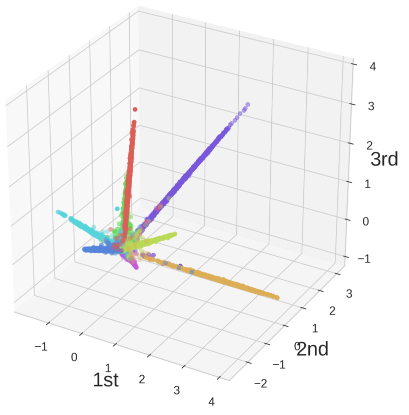
(a) L1W (Before ReLU) (b) L1W (After ReLU)

(c) L2W (Before ReLU) (d) L2W (After ReLU)

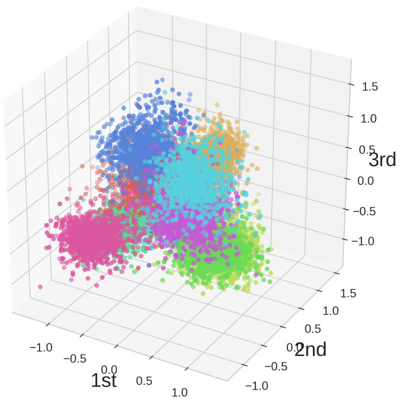
Figure A.2. The top three principal components of learned representations (L1W, and L2W). Note that representation characteristics of L1W and L2W are very similar to those of the baseline because weight decay methods do not directly shape representations.



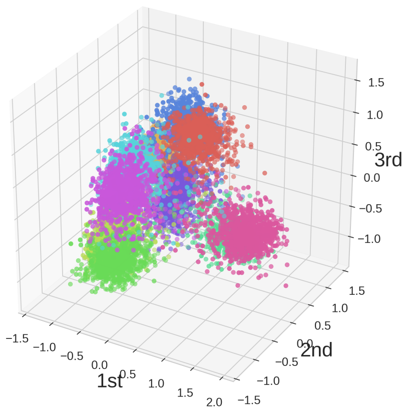
(a) Dropout (Before ReLU)



(b) Dropout (After ReLU)

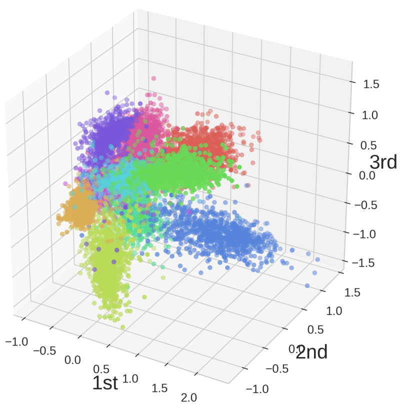


(c) BN (Before ReLU)

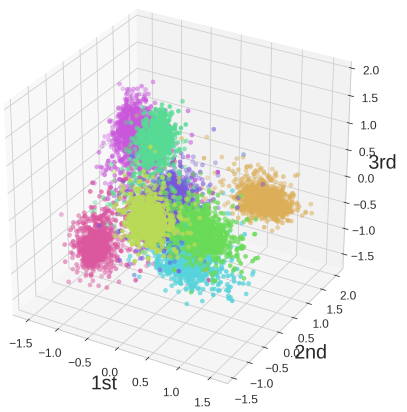


(d) BN (After ReLU)

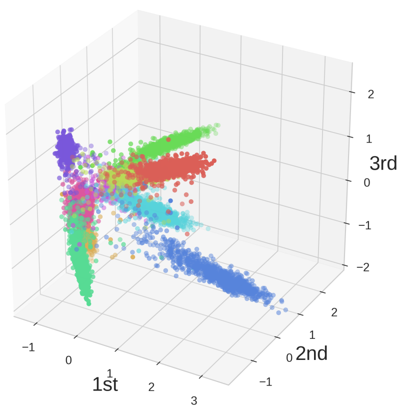
Figure A.3. The top three principal components of learned representations (Dropout, Batch normalization).



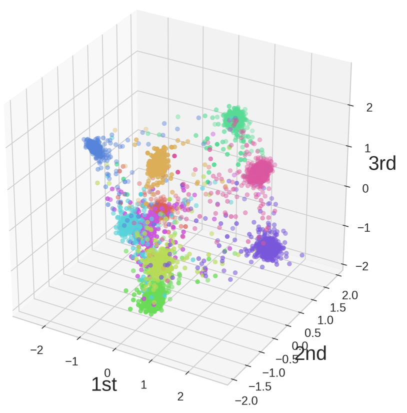
(a) CR (Before ReLU)



(b) CR (After ReLU)

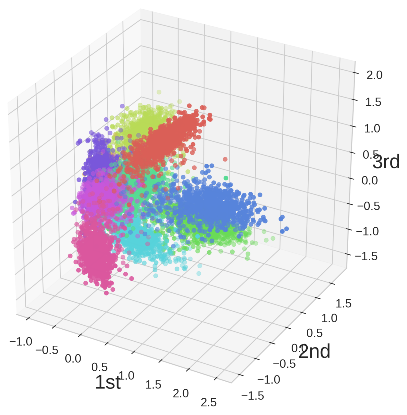


(c) cw-CR (Before ReLU)

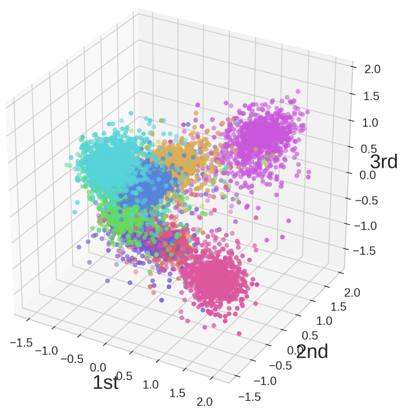


(d) cw-CR (After ReLU)

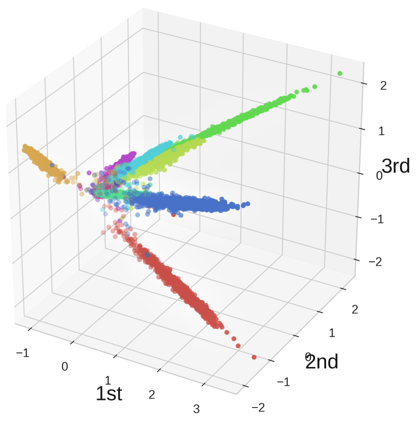
Figure A.4. The top three principal components of learned representations (CR, cw-CR).



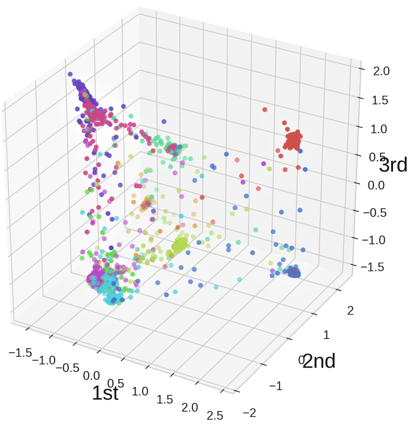
(a) VR (Before ReLU)



(b) VR (After ReLU)

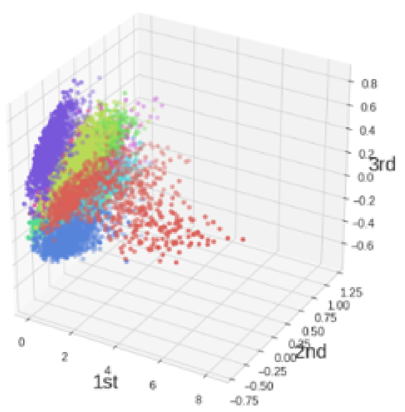


(c) cw-VR (Before ReLU)

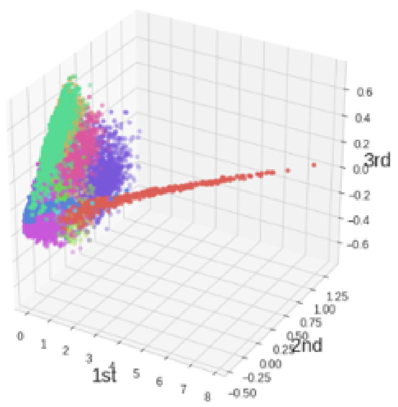


(d) cw-VR (After ReLU)

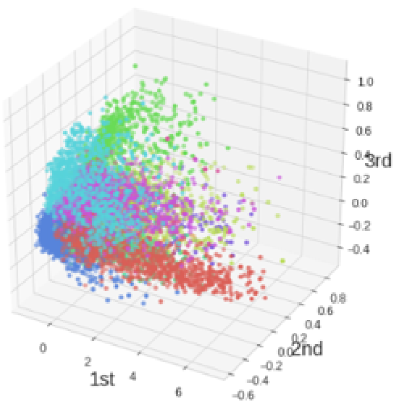
Figure A.5. The top three principal components of learned representations (VR, cw-VR).



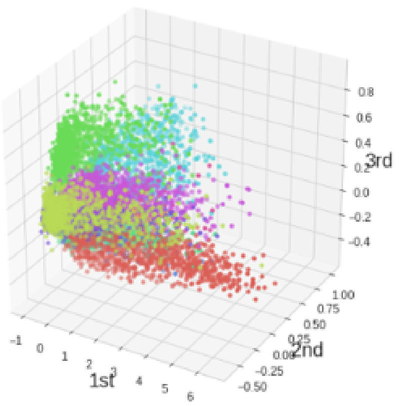
(a) RR (Before ReLU)



(b) RR (After ReLU)



(c) cw-RR (Before ReLU)



(d) cw-RR (After ReLU)

Figure A.6. The top three principal components of learned representations (RR, cw-RR).

B. Proofs

Theorem 1 *For a deep network $\mathcal{N}_{\mathcal{A}} = (\mathbf{W}, \mathbf{b})$ whose layer l is linear, there exists $\widetilde{\mathcal{N}}_{\mathcal{A}} = (\widetilde{\mathbf{W}}, \widetilde{\mathbf{b}})$ that satisfy the following conditions:*

$$\forall\, \mathbf{x},\ \mathcal{N}_{\mathcal{A}}(\mathbf{x}) = \widetilde{\mathcal{N}}_{\mathcal{A}}(\mathbf{x});$$
$$\forall\, \mathbf{x},\ \widetilde{\mathbf{z}}_l = \mathbf{Q}(\mathbf{z}_l - \mathbf{m}),$$

where $\mathbf{Q}$ is any nonsingular square matrix of a proper size and $\mathbf{m}$ is any vector of a proper size.

Proof. Proof is based on a simple construction. Choose the weights of $\widetilde{\mathcal{N}}_{\mathcal{A}}$ as below.

$$\begin{aligned}
\widetilde{\mathbf{W}}_l^T &= \mathbf{Q}\,\mathbf{W}_l^T \\
\widetilde{\mathbf{b}}_l &= \mathbf{Q}(\mathbf{b}_l - \mathbf{m}) \\
\widetilde{\mathbf{W}}_{l+1}^T &= \mathbf{W}_{l+1}^T\,\mathbf{Q}^{-1} \\
\widetilde{\mathbf{b}}_{l+1} &= \mathbf{b}_{l+1} + \mathbf{W}_{l+1}^T\,\mathbf{m}
\end{aligned}$$

For all the other layers, choose the same as $\mathcal{N}_{\mathcal{A}}$'s weights. Then, clearly $\widetilde{\mathbf{z}}_{l-1} = \mathbf{z}_{l-1}$ and therefore $\widetilde{\mathbf{z}}_l = \widetilde{\mathbf{W}}_l^T\,\mathbf{z}_{l-1} + \widetilde{\mathbf{b}}_l = \mathbf{Q}(\mathbf{W}_l^T\,\mathbf{z}_{l-1} + \mathbf{b}_l - \mathbf{m}) = \mathbf{Q}(\mathbf{z}_l - \mathbf{m})$. Also, $\widetilde{\mathbf{z}}_{l+1} = \widetilde{\mathbf{W}}_{l+1}^T \widetilde{\mathbf{z}}_l + \widetilde{\mathbf{b}}_{l+1} = \mathbf{W}_{l+1}^T\,\mathbf{Q}^{-1}\,\mathbf{Q}(\mathbf{z}_l - \mathbf{m}) + \mathbf{b}_{l+1} + \mathbf{W}_{l+1}^T\,\mathbf{m} = \mathbf{z}_{l+1}$. Because the activation vector of layer $l+1$ is the same for $\mathcal{N}_{\mathcal{A}}$ and $\widetilde{\mathcal{N}}_{\mathcal{A}}$, the resulting outputs $\mathcal{N}_{\mathcal{A}}(\mathbf{x})$ and $\widetilde{\mathcal{N}}_{\mathcal{A}}(\mathbf{x})$ are exactly the same as well. □

Theorem 2 *For a deep network $\mathcal{N}_{\mathcal{A}} = (\mathbf{W}, \mathbf{b})$ whose activation function of layer l is ReLU, there exists $\widetilde{\mathcal{N}}_{\mathcal{A}} = (\widetilde{\mathbf{W}}, \widetilde{\mathbf{b}})$ that satisfy the following conditions:*

$$\forall\, \mathbf{x},\ \mathcal{N}_{\mathcal{A}}(\mathbf{x}) = \widetilde{\mathcal{N}}_{\mathcal{A}}(\mathbf{x});$$
$$\forall\, \mathbf{x},\ \widetilde{\mathbf{z}}_l = \mathbf{Q}\,\mathbf{z}_l,$$

where $\mathbf{Q}$ is any permuted positive diagonal matrix of a proper size. Furthermore, it can be shown that any $\mathbf{Q}$ that satisfy the above two conditions must be a permuted positive diagonal matrix.

Proof. For simplicity, we denote $\mathbf{a}^+ = \text{ReLU}(\mathbf{a})$ and $\mathbf{a}^- = \text{ReLU}(-\mathbf{a})$. Then $\mathbf{a} = \mathbf{a}^+ + \mathbf{a}^-$. We denote $\mathbf{h}_l = \mathbf{W}_l^T \mathbf{z}_l + \mathbf{b}_l$, which is the representation before applying the activation function, such that $\mathbf{z}_l = \mathbf{h}_l^+$. If we choose $\widetilde{\mathbf{W}}_l^T = \mathbf{Q}\,\mathbf{W}_l^T$ and $\widetilde{\mathbf{W}}_{l+1}^T = \mathbf{W}_{l+1}^T\,\mathbf{Q}^{-1}$, our focus is to find an invertible matrix $\mathbf{Q}$ that satisfy $\mathbf{W}_{l+1}^T\,\mathbf{Q}^{-1}(\mathbf{Q}\,\mathbf{h}_l)^+ + \mathbf{b}_{l+1} = \mathbf{W}_{l+1}^T\,\mathbf{h}_l^+ + \mathbf{b}_{l+1}$ for all $\mathbf{x}$. This reduces down to finding $\mathbf{Q}$ that satisfies $(\mathbf{Q}\,\mathbf{h}_l)^+ = \mathbf{Q}\,\mathbf{h}_l^+$. We denote i'th row of $\mathbf{Q}$ as $\mathbf{q}_i^T$, and the statement mentioned above can be written as: $(\mathbf{q}_i^T\,\mathbf{h}_l)^+ = \mathbf{q}_i^T\,\mathbf{h}_l$. For $\mathbf{q}_i^T$ that satisfy $(\mathbf{q}_i^T\,\mathbf{h}_l) \geq 0$, obviously $(\mathbf{q}_i^T\,\mathbf{h}_l)^+ = \mathbf{q}_i^T\,\mathbf{h}_l^i$. If we substitute $\mathbf{h}_l$ with $\mathbf{h}_l^+ - \mathbf{h}_l^-$, then $\mathbf{q}_i^T\,\mathbf{h}_l^- = 0$. For $\mathbf{q}_i^T$ that satisfy $\mathbf{q}_i^T\,\mathbf{h}_l < 0$, we can derive the condition $\mathbf{q}_i^T\,\mathbf{h}_l^+ = 0$.

Now, we will show that $\mathbf{Q}$ should be a permuted positive diagonal matrix using the statements proved above. For a permuted positive diagonal matrix, $\{\mathbf{q}_i^T\}$ are linearly independent and each $\mathbf{q}_i^T$ has only one positive element. Because $\mathbf{Q}$ needs to be invertible (otherwise information loss occurs and $\mathcal{N}_{\mathcal{A}}(\mathbf{x}) = \widetilde{\mathcal{N}}_{\mathcal{A}}(\mathbf{x})$ cannot be achieved), it is trivial that each $\mathbf{q}_i^T$ is linearly independent. To show each $\mathbf{q}_i^T$ has only one positive element, let's assume $\mathbf{q}_i^T$ has more than one non-zero elements. If we denote $\mathbf{q}_{ik}^T$ as the $\mathbf{q}_i^T$'s k-th element, and $\mathbf{h}_{lk}$ as the $\mathbf{h}_l$'s k-th element, we can divide the element indexes as follow.

$$A = \{k|\,\mathbf{q}_{ik}^T \neq 0 \text{ and } \mathbf{h}_{lk} > 0\}$$

$$B = \{k|\,\mathbf{q}_{ik}^T \neq 0 \text{ and } \mathbf{h}_{lk} < 0\}$$

Then, we can consider $\mathbf{h}_l$ such that $A \neq \emptyset$ and $B \neq \emptyset$. If $\mathbf{q}_i^T\,\mathbf{h}_l >= 0$, then

$$\sum_{j\in A} \mathbf{q}_{ij}^T\,\mathbf{h}_{lj} > \sum_{j\in B} \mathbf{q}_{ij}^T\,\mathbf{h}_{lj}\,.$$

In this case, the right side should be zero because $\mathbf{q}_i^T\,\mathbf{h}_l^- = 0$. However, $\mathbf{q}_{ij}^T$ should be zero to satisfy $\mathbf{q}_i^T\,\mathbf{h}_l^- = 0$ because $\mathbf{h}_l$ can be chosen arbitrary in the range that $A \neq \emptyset$ and $B \neq \emptyset$ and $\mathbf{q}_i^T\,\mathbf{h}_l >= 0$. This is a contradiction due to the definition of B. We can prove the case of $\mathbf{q}_i^T\,\mathbf{h}_l < 0$ similarly, which shows that each $\mathbf{q}_i^T$ has only one non-zero element. To show the $\mathbf{q}_i^T$'s one element is positive, we denotes the $\mathbf{q}_i^T$'s only one element as $\mathbf{q}_{ij}^T$, in which j is the index of the non-zero element. When $\mathbf{q}_{ij}^T < 0$, we can consider $\mathbf{h}_l$ such that $\mathbf{h}_{lj} < 0$. In this case, $(\mathbf{q}_i^T \mathbf{h}_l) > 0$, but $\mathbf{q}_i^T \mathbf{h}_l^- > 0$, which contradicts the condition $\mathbf{q}_i^T \mathbf{h}_l^- = 0$. If we consider $\mathbf{h}_l$ such that $\mathbf{h}_{lj} > 0$, $(\mathbf{q}_i^T \mathbf{h}_l) < 0$, but $\mathbf{q}_i^T \mathbf{h}_l^+ > 0$, which contradicts the condition $\mathbf{q}_i^T \mathbf{h}_l^+ > 0$. Therefore, $\mathbf{q}_{ij}^T$ is positive. $\square$

Acknowlegement

Firstly, I have to thank my advisor Professor Wonjong Rhee for his endless support, guidance, and advice during my Ph.D. time. I am very grateful to Professor Rhee for all I learned from him regarding research style. He taught me how to learn and study things in-depth by myself and had a significant influence on not only my Ph.D. study but also my thinking and attitude toward life. This work would have been impossible without his support and encouragement.

Besides my advisor, I would also like to thank my committee members: Professor Kyogu Lee, Professor Nojun Kwak, Professor Bongwon Suh, and Professor Yongjae Lee for their insightful comments and valuable suggestions that broadened this work from diverse perspectives.

Most of the experimental works in Chapter 3 and 5 was done in collaboration with Kyungeun Lee. I am grateful to her for the significant amount of help and words of encouragement. I would like to express my sincere gratitude to Hyunghun Cho for advice on software implementations and for helpful discussions concerning hyperparameter optimization of deep networks. I also thank Changho Shin for an extensive literature review related to Chapter 3 and for his contribution to the work presented in Chapter 5.

I thank my friends in Applied Data Science Lab. for their valuable comments and conversations to improve this work. I was lucky enough to have such great colleagues: Jeongyun Han, Eunjung Lee, Won Shin, Seungwook Kim, Euna Jung, Moonjung Eo, Sedong Kim, Jungwook Shin, Duhun Hwang, and Yongjin Kim. All the fun we have had in the last four years will be best remembered for me.

Last but not least, I really appreciate my family: my mother, wife, daughter, and son for providing me with unfailing support throughout my years of study and my life in general.

The author was supported by the Ministry of National Defense and Republic of Korea Air Force. This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2017R1E1A1A03070560) and by SK Telecom Co., Ltd.