DNN REPRESENTATIONS AS CODEWORDS: MANIPULATING STATISTICAL PROPERTIES VIA PENALTY REGULARIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Performance of Deep Neural Network (DNN) heavily depends on the characteristics of hidden layer representations. Unlike the codewords of channel coding, however, the representations of learning cannot be directly designed or controlled. Therefore, we develop a family of penalty regularizers where each one aims to affect one of representation's statistical properties such as sparsity, variance, and covariance. The regularizers are extended to perform class-wise regularization, and the extension is found to provide distinct shaping capabilities. A variety of statistical properties are investigated for 10 different regularization strategies, and several interesting findings are reported. Using the family of regula

Performance improvement is observed for the MNIST, CIFAR-100, and CIFAR-10 classification problems. But more importantly, statistical properties of representations are confirmed to be significantly affected by the choice of regularizers. While the designed regularizers show more explicit relation with the targeted statistical properties, the widely used dropout and batchnorm turn out to mildy affect most of the properties.

that were tested.

The results show that

Shaping and its effect on performance, generalization, and optimization speed. (Therefore, penalty regularization is not really the same as the traditional concept of regularization.

We call this "property shaping"

Motivated by statistical properties of optimal code designs for well known channel coding problems, we focus on penalty regularizers on L1, variance, and covariance of DNN representations and also their class-wise versions. We find that

1 INTRODUCTION

With a Deep Neural Network (DNN), information contained in the input data x is transformed into multiple *representations* over multiple layers. Performance of machine learning tasks is known to heavily depend on the choice of representations over $p(x)$, but $p(x)$ is almost always unknown and the representations cannot be directly controlled to match an arbitrary design even if $p(x)$ was known. As of today, the best that can be done is to indirectly affect the representations by adding constraints, modifying cost function, tuning learning process, etc.

In Shannon's information theory, channel coding theory deals with the problem of reliably sending the maximum amount of information through a given channel $p(y|x)$ (Cover & Thomas, 2012). Because the channel is known and fixed, coding becomes a design problem where one needs to design codebook, encoder, and decoder. Usually, the channel is used n times in a sequence to send a codeword of length n (expressed as x^n). A codebook is a collection of all codewords that can be chosen and sent to the channel, and each message w with information of interest is mapped into one of the codewords during the design phase.

Because channel coding is a design problem, the optimal solutions are well understood for some of the important applications such as Gaussian channel and Binary Symmetric Channel (BSC). Gaussian channel is the most important continuous alphabet channel problem. It assumes that the received signal y^n is a noisy version of x^n, where the noise is independent of x and additive with an i.i.d. Gaussian distribution over the n symbols. Surprisingly, when $n \to \infty$, the optimal codebook turns out to be a collection of codewords that are generated by randomly drawing numbers from a Gaussian distribution. Then, a codeword's n symbols form an i.i.d. Gaussian distribution (for instance, see Chap. 9 of Cover & Thomas (2012)). BSC is one of the most popular discrete alphabet channel problem, and x can take a value of 0 or 1. The received signal y is a corrupted version of x where it is flipped with a fixed probability. For BSC, Hamming code is a well known solution where redundant bits are included in x^n to resist the corruption (for instance, see Chap. 7 of Cover & Thomas (2012)). The correction capability is dependent on the minimum Hamming distance among all pairs of two different codewords.

It would be helpful if the elegant channel coding theories can be applied to the design and control of DNN representation, but unfortunately the representation problem is clearly different from the channel coding problem. First of all, a learning problem does not have a fixed and known channel $p(y^n|x^n)$. Secondly, we can design and control the DNN model to use, but we do not have the luxury of explicitly designing codebook, and representations can be formed in anyway.

Nonetheless, we can attempt to gain insights and ideas from the established coding theory. In this work, we first observe that the optimal solution to Gaussian channel problem has i.i.d. Gaussian codewords. Although it is unclear if forcing representations of a layer to have an i.i.d. Gaussian property, we experiment the idea by expanding known penalty regularization strategies to include L1, variance, and covariance. L1 and covariance (Cogswell et al., 2016) have been studied before (individually), but with our best knowledge, variance of a unit (neuron) and using combination of them have not been considered in the literature. Secondly, we observe that only a single codeword is assigned to a message (label for learning problems) for a well designed codebook. When this idea is applied to learning problems via penalty regularization, the penalty term needs to be applied per-class such that we can shape the codeword of each label. Note that almost all of the existing penalty regularization strategies have been applied to all classes together. Thirdly, we observe that Gaussian codebook and Hamming codebook are fundamentally different. A Gaussian codebook uses continuous alphabets in an uncorrelated manner over n symbols, but Hamming codebook uses only 0 or 1. With the difference, it is inevitable for Gaussian code to utilize long codewords (very large n) and distinguish codewords in a stochastic manner, while Hamming code needs to utilize carefully designed vector-space structures (orthogonality, null space, etc.) using relatively short codewords. Because we are often interested in a relatively small number of neurons for representations, we consider a regularization strategy where each label's activation for a unit is 'hardened' (by cw-VR regularizer that is introduced later) such that the representation vector is closer to an on-off hard activation than a Gaussian-shaped soft activation.

1.1 Related Works

Regularization

The classical regularizers apply L2 (Hoerl & Kennard, 1970) and L1 (Tibshirani, 1996) penalties to the *weights* of models, and they are widely used for DNN as well. Wen et al. (2016) extended L1 regularizer by using group lasso to regularize the structures of DNN (i.e., filters, channels, filter shapes, and layer depth). Regularization has been applied to *representations*, too. Srivastava et al. (2014) devised dropout that randomly applies activation masking over the neurons. While dropout is applied in a multiplicative manner, Glorot et al. (2011) used L1 penalty regularization on the activations to encourage sparse representations. XCov proposed by Cheung et al. (2014) minimizes the covariance between autoencoding units and label encoding units of the same layer such that representations can be disentangled. DeCov, developed by Cogswell et al. (2016), is also a penalty regularizer and it minimizes the off-diagonals of a layer's representation covariance matrix. DeCov reduces co-adaptation of units by encouraging units to be decorrelated. It is called CR (Covariance Regularizer) in this study for the purpose of consistent naming. *Statistics* over mini-batch samples or in-layer activations have been used for regularization, too. Batch normalization proposed by Ioffe & 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 good regularizer. In line with batch normalization, weight normalization, developed by Salimans & Kingma (2016),

uses mini-batch statistics to normalize weight vectors. Layer normalization proposed by Ba et al. (2016) is a RNN version of batch normalization, where they compute the mean and variance used for normalization from all of the summed inputs to the neurons in a layer on a single training case. There are many other publications on DNN regularization techniques, but we still do not have a sufficient understanding on how they really work. A recent work by Zhang et al. (2016) shows that the traditional concept of controlling generalization error by regularizing the effective capacity cannot be applied to DNN.

Class-wise Learning
True class information is available for supervised learning problems. Traditionally, the class information has been used only for evaluating correctness of predictions and the relevant cost function terms. Some of the recent works, however, have adopted the class-wise concept in the learning algorithm itself. In those works, class information is used as a switch or for emphasizing the discriminative aspects over different classes. As an example, Li et al. (2008) proposed a kernel learning method using class-wise information to model the manifold structure. They modify locality preserving projection to be class dependent. Jiang et al. (2011) added label consistent regularizers for learning a discriminative dictionary. As for DNN, a recent work by Belharbi et al. (2017) uses a regularization function that penalizes the pair-wise distance of two representations that belong to the same class. This approach is similar to cw-VR (class-wise Variance Regularizer) in our work, but they use sum of pair-wise distances while we use representation variance of same class samples.

2 Three Statistical Properties and Class-Wise Regularization

For channel coding problems, we can characterize the statistical properties of optimal codewords as discussed in Section 1. Our goal is to make DNN representation vectors to have such statistical properties and analyze their effects. Because an explicit design and control of representation vector is not possible for the learning problems, instead we utilize penalty regularizers to manipulate the statistical properties.

2.1 Three Statistical Properties

Three of the most basic statistical properties are considered in this work - sparsity, variance, and covariance. Sparsity over layer l's representation vector $\mathbf{h}_l$ has been extensively studied in literature. For variance, we are referring to the variance of a unit's activation values over mini-batch samples. When the variance is forced to be very small, the activation value needs to be close to the sample mean for all labels, and therefore the unit loses its discriminative power over multiple labels. While this is undesirable, regularizing variance turns out to be meaningful because the cross-entropy cost function prevents the variance becoming zero and a healthy compromise can be achieved between cross-entropy and variance terms. This is similar to the situation of classic weight regularization, where the weights actually never become zero by regularization. For covariance, we calculate pair-wise covariance over the unit activations of a layer. When covariance is evaluated to be large for a pair of units (neurons) in the same layer, it indicates that the two are strongly correlated. This is undesirable if we are pursuing i.i.d. property over unit activations, and having a regularizer to control the level of correlation can be useful.

2.2 Class-wise regularization

To pursue statistical properties for each class, we adopt the concept of class-wise learning[1]. For instance, it is undesirable if the variance becomes exactly zero for a unit's activation as mentioned above. Variance of zero for a class, however, can be desirable because it simply states that a consistent activation value will be observed over all samples with the same class label. Note that overall variance over all labels can be still large while class-wise variance is zero - as long as inter-class difference exists, the overall variance will not be zero. We combine this concept of class-wise regularization to the three concepts of sparsity, variance, and covariance. Analytical formulations can be found in the following section.

[1]A very recent work by Belharbi et al. (2017) contains an idea that is similar to ours, but our work was performed before encountering the paper.

3 PENALTY LOSS FUNCTIONS

In this section, we provide the model for calculating basic statistics and formulate the penalty loss functions that are used for regularization.

3.1 BASIC STATISTICS

For layer l, the output activation vector of a linear filter followed by ReLU is defined as $\mathbf{h}_l = \max(\mathbf{W}_l^\top \mathbf{h}_{l-1} + \mathbf{b}_l, 0)$. Because we will be focusing on layer l for most of the explanations, we drop the layer index and $\mathbf{h}$ is used to indicate $\mathbf{h}_l$ instead. Then, h_i is the ith element of $\mathbf{h}$ (i.e. activation of ith unit), and w_{ki} is the (k, i) element of $\mathbf{W}$.

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.

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

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

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

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

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

$$\mu_i^m = \frac{1}{|S_m|} \sum_{n \in S_m} h_{i,n} \tag{4}$$

$$c_{i,j}^m = \frac{1}{|S_m|} \sum_{n \in S_m} (h_{i,n} - \mu_i^m)(h_{j,n} - \mu_j^m) \tag{5}$$

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

Here, S_m is the set containing indexes of the samples whose label is m, and $|S_m|$ is the cardinality of the set S_m.

3.2 PENALTY LOSS FUNCTIONS

Using the notations in Section 3.1, the loss functions and their derivatives can be derived and summarized as in Table 1. L1-weight and L2-weight are well-known, and they apply L1 and L2 penalties on the weights, respectively. The rest in the table apply penalty on the representation. L1-rep is similar to L1-weight, but the penalty is applied to the representation $\mathbf{h}$. Obviously, L2 can also be applied to representation, but it is excluded in this study because it tends to perform worse than L1 when applied to representation. VR (Variance Regularization) calculates variance of each unit's activation over mini-batch dataset and uses the calculated value as the penalty. CR(Cross-covariance Regularization) uses off-diagonal terms of the mini-batch covariance matrix of activations as the penalty term. As mentioned earlier, CR in this work is the same as DeCov presented by Cogswell et al. (2016) but we use the term CR for the consistency of naming. As in DeCov, we subtract variance terms and consider cross-covariance terms only (see penalty loss function in Table. 1). cw-VR and cw-CR are similar to VR and CR, respectively, except that the values are calculated for each class using the mini-batch samples with the same class label. cw-L1-rep can be defined, but its penalty loss function turns out to be the same as L1-rep's loss function. Therefore, cw-L1-rep is excluded in this study.

Interpretation of derivatives
While the penalty functions were chosen from the three distinct statistical properties and class-wise concept, their derivatives show that some of them are closely related. For the derivatives of VR and

Table 1: Penalty loss functions of regularizers

	Penalty loss function	Derivatives
$\Omega_{L1\text{-}weight}$	$= \sum_k \sum_i \lvert w_{ki} \rvert$	$\frac{\partial \Omega_{L1\text{-}weight}}{\partial w_{ki}} = sign(w_{ki})$
$\Omega_{L2\text{-}weight}$	$= \sum_k \sum_i w_{ki}^2$	$\frac{\partial \Omega_{L2\text{-}weight}}{\partial w_{ki}} = 2w_{ki}$
$\Omega_{L1\text{-}rep}$	$= \sum_n \sum_i \lvert h_{i,n} \rvert$	$\frac{\partial \Omega_{L1\text{-}rep}}{\partial h_{i,n}} = sign(h_{i,n})$
Ω_{VR}	$= \sum_i v_i$	$\frac{\partial \Omega_{VR}}{\partial h_{i,n}} = \frac{2}{N}(h_{i,n} - \mu_i)$
Ω_{CR}	$= \sum_i \sum_j (c_{i,j})^2 - \sum_i (v_i)^2$	$\frac{\partial \Omega_{CR}}{\partial h_{i,n}} = \frac{2}{N} \sum_{j \neq i} c_{i,j}(h_{j,n} - \mu_j)$
$\Omega_{cw\text{-}VR}$	$= \sum_m \sum_i v_i^m$	$\frac{\partial \Omega_{cw\text{-}VR}}{\partial h_{i,n}} = \frac{2}{\lvert S_m \rvert}(h_{i,n} - \mu_i^m), n \in S_m$
$\Omega_{cw\text{-}CR}$	$= \sum_m (\sum_i \sum_j (c_{i,j})^2 - \sum_i (v_i)^2)$	$\frac{\partial \Omega_{cw\text{-}CR}}{\partial h_{i,n}} = \frac{2}{\lvert S_m \rvert} \sum_{j \neq i} c_{i,j}^m (h_{j,n} - \mu_j^m), n \in S_m$

CR, it can be observed that they have similar structures. If VR's derivative $\frac{\partial \Omega_{VR}}{\partial h_{i,n}}$ becomes zero for all i, then CR's derivative $\frac{\partial \Omega_{CR}}{\partial h_{i,n}}$ becomes zero as well. The vice versa does not hold, but the effects of VR and CR can be expected to be similar or at least related for the learning process. In the same way, the relationship between cw-VR and cw-CR is the same as the relationship between VR and CR. Therefore, we can expect cw-VR and cw-CR to have similar effects, too. On the other hand, the derivative of L1-rep has a distinct formulation and it can be expected to have a distinct effect on learning.

There is another important effect that is not necessarily obvious from the derivative formulations. For L1-weight and L2-weight, the derivatives are dependent on the weights w_{ki} only, and they are independent of the activations $h_{i,n}$. Therefore, the weights need to become smaller to reduce the regularization penalty. For the other five representation regularizers, their derivatives are all dependent on activation $h_{i,n}$. So, a simple way to reduce the regularization penalties is to scale the activations to small values (instead of satisfying the balances among the terms in the equation to reach zero gradient and force the desired statistical properties). This scaling will not have any effect on prediction output as long as all the elements of $\mathbf{h}^l$ are scaled together to $\alpha \mathbf{h}^l$ - the last softmax layer works as a normalization function for the output layer, and therefore the cross-entropy penalty term is not affected by such a scaling. This means that there is a chance for the learning algorithm to squash activations just so that representation regularization terms can be ignored. As we will see later, indeed activation squashing happens by learning, but the desired statistical properties are still sufficiently enforced. Nonetheless, it must be possible to design better penalty regularizers that are immune to activation squashing, and such regularizers might be much more effective for manipulating statistical properties of representations.

4 EXPERIMENTS - MNIST

To compare performances of 10 regularization strategies, experiments with the MNIST dataset were conducted(LeCun et al., 1998). The results indicate that explicit shaping strategies slightly outperform than other strategies and perform better when the strategies are applied to upper layers of DNN. Next, we investigate and compare statistical properties among them, and finally visualize representation to understand the statistical properties intuitively.

Table 2: Error performance of popular regularizers (MNIST)

Layer	Baseline	Penalty on weight		Implicit method	
		L1-weight	L2-weight	Dropout	BN
All	**3.06**±0.15	**2.90**±0.08	**2.96**±0.09	**4.08**±0.06	**2.69**±0.06

Table 3: Error performance of representation shaping regularizers (MNIST)

Layer	All classes			Class-wise	
	L1-rep	VR	CR	cw-VR	cw-CR
Output	**2.61**±0.04	**2.67**±0.15	**2.62**±0.07	**2.56**±0.02	**2.55**±0.08
Layer 5	**2.61**±0.11	2.70±0.03	2.67±0.04	2.63±0.05	2.61±0.06
Layer 4	2.75±0.05	2.89±0.11	2.69±0.13	2.67±0.12	2.71±0.04
Layer 3	3.35±0.08	3.16±0.09	3.11±0.13	3.22±0.06	3.22±0.06
Layer 2	3.40±0.11	3.15±0.21	3.01±0.10	3.14±0.10	3.24±0.11
Layer 1	4.31±0.14	2.98±0.09	3.13±0.09	3.25±0.04	3.14±0.03

Table 4: Error performance of representation shaping regularizers - multiple layers (MNIST)

Layer	All classes			Class-wise	
	L1-rep	VR	CR	cw-VR	cw-CR
Output	2.61±0.04	2.67±0.15	2.62±0.07	2.56±0.02	2.55±0.08
Output, 5	**2.48**±0.12	2.67±0.11	**2.43**±0.08	**2.46**±0.07	2.55±0.10
Output, 5, 4	2.78±0.11	**2.58**±0.06	2.80±0.12	2.53±0.07	**2.48**±0.07
Output, 5, 4, 3	2.79±0.10	2.78±0.08	2.83±0.14	2.80±0.10	2.72±0.04
Output, 5, 4, 3, 2	3.19±0.10	2.91±0.13	2.77±0.07	2.90±0.10	2.75±0.07
All	3.26±0.09	2.86±0.07	2.80±0.08	2.83±0.07	2.85±0.12

4.1 PERFORMANCE RESULTS

We use a Multilayer Perceptron (MLP) with five hidden fully connected layers and an output layer. Each hidden fully connected layer has 100 units with Rectified Linear Unit (ReLU) activation function, and the output layer is composed of 10 softmax units. Penalty loss weight is determined using validation dataset with grid search. We trained each model five-times and measured error performance using average and one standard deviation of the five. All experiments in this work were carried out using TensorFlow 1.3. The MNIST problem shows high performance even with a simple model, so there is little difference in error performance. However, we can see that cw-CR performs the best among the 10 regularizers (Table 2, 3). When explicit regularization strategies are applied to upper layers, it shows better performance. In addition, we can see that better performance is obtained when the regularizers are applied to upper layers like Layer 5 or an output layer. Regularizers are applied to representation before softmax function. Table 4 shows the best performance when applying explicit shaping strategies to upper 2 or 3 layers including the output layer. This may be because back propagation starts from the layer applied regularizers.

4.2 STATISTICAL PROPERTIES OF 10 REGULARIZATION STRATEGIES

We use 9 metrics to compare statistical properties of 10 regularization strategies. Among them, seven metrics are calculated from equations of penalty loss functions in section **??**. The raw numbers of the metrics are difficult to interpret because the power (?) is different for each metric, so we normalize metrics as follows (see the raw data of the metrics in 10). First, the metrics of each strategy are divided by $\sqrt{\Omega_{L2\text{-}weight}}$ of the its strategy. The reason for normalizing to $\sqrt{\Omega_{L2\text{-}weight}}$ is because XXX and it is power term. The metrics to be normalized are also squared term or . Next, for comparison with baseline metrics, we divide the metrics of each strategy by the metrics of baseline, and multiply 100 to make readable numbers. Since $\Omega_{L2\text{-}weight}$ is used for normalization, the value of all regularizers are 100, and all the metrics of baseline is also 100 because they are used for normalization term.

The remaining two metrics are the average number of activated classes per unit as measure of sparsity and ratio of dead units. $\Omega_{L1\text{-}weight}$ and $\Omega_{L2\text{-}weight}$ are calculated from the weights of all layers

Table 5: Evaluation of statistical properties - popular strategies

Metric	Baseline	Penalty on weight		Implicit method	
		L1-weight	L2-weight	Dropout	BN
$\Omega_{L1\text{-}weight}$ (all)	100.00	88.05	92.25	99.14	84.82
$\Omega_{L2\text{-}weight}$ (all)	100.00	100.00	100.00	100.00	100.00
$\Omega_{L1\text{-}rep}$	100.00	115.28	110.26	36.11	16.94
Ω_{VR}	100.00	113.97	109.58	61.18	27.69
Ω_{CR}	100.00	111.55	107.51	39.35	5.80
$\Omega_{cw\text{-}VR}$	100.00	114.08	109.68	72.91	50.50
$\Omega_{cw\text{-}CR}$	100.00	112.68	108.55	78.49	20.54
Avg_Act_Class	6.57	7.08	6.93	4.65	3.98
$Ratio_Dead_Unit$	14%	4%	9%	0%	1%

Table 6: Evaluation of statistical properties - representation shaping regularizers

Metric	All classes			Class-wise	
	L1-rep	VR	CR	cw-VR	cw-CR
$\Omega_{L1\text{-}weight}$ (all)	93.08	96.42	95.83	86.85	84.14
$\Omega_{L2\text{-}weight}$ (all)	100.00	100.00	100.00	100.00	100.00
$\Omega_{L1\text{-}rep}$	**1.07**	9.16	9.73	3.41	5.49
Ω_{VR}	7.77	9.24	9.42	**3.91**	5.28
Ω_{CR}	0.33	0.64	0.63	**0.15**	0.27
$\Omega_{cw\text{-}VR}$	19.85	28.12	29.61	**11.25**	14.27
$\Omega_{cw\text{-}CR}$	3.69	6.79	7.15	**1.66**	2.37
Avg_Act_Class	**0.26**	5.53	6.04	4.14	5.29
$Ratio_Dead_Unit$	**74%**	6%	4%	23%	7%

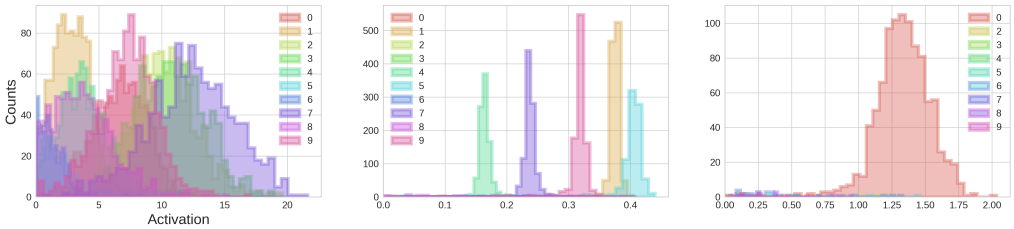
excluding bias, and the others are calculated from activations of test dataset in Layer 5. We can observe two distinct groups of regularizers when it comes to metric magnitude of representation regularization (Table 5, 6). We can clearly observe that representation shaping regularizers have much smaller values on represention metrics. This is because represenation regularizes squash activations in the way described in Section 3.2. As mensioned in Section 3.2, VR and cw-VR are related to CR and cw-CR, respectively. We can see that their values of metrics are similar to each other. Despite of this similarity of the five representation regularization, L1-rep and cw-VR have unique characteristics. L1-rep obviously enforces sparsity and causes much more dead units that the others. Representation regularizer cw-VR always shows the smallest metric among four starategies (VR, CR, cw-VR, cw-CR). This can be evidence of their close relationship. Dropoust and BN are located somewhere between baseline and representation regularizers. They have similar effects on represenation metrics, but much less than the five represenatation regularizers.

4.3 Visualization of representation

Due to activation squashing by learning, metrics of statistical properties tend to be exaggerated. We, therefore, visualize representation of Layer 5 to more intuitively understand the difference in the statistical properties of regularizers.

Histogram of a single unit

We first visualize the distribution of activation per unit to observe sparsity and variance properties. Activation histogram shows the distribution of 10000 test data and each color indicates different classes. Since activations are generated as output of ReLU activation function, they have some portion of zeros, which can distort the histgram. We, therefore, excluded zeros from activations when drawing the histogram. Baseline has a large class-wise variance and overlaps with other classes. The histogram of cw-VR, however, shows the effect of separating the classes because class-wise variance is reduced. On the other hand, L1-rep can be confirmed that only one class is acitvated to provide sparsity. As described in Table 6, Avg_Act_Class of L1-rep is close to zero, so most of its histograms shows very few samples.



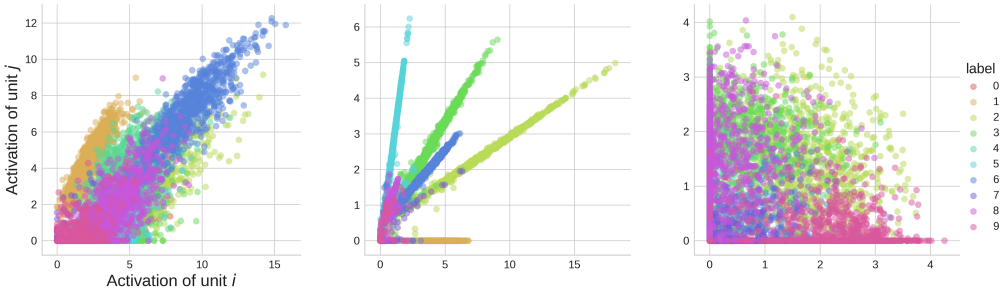
(a) Baseline (b) cw-VR (c) L1-rep

Figure 1: Histogram samples figure.

TODO:describe something regarding figure if required...

Scatter plot of a unit pair

unit unit representation scatter plot . figure XXX, Ł baseline Ł metric covariance Ł . lcr unit covariance baseline . cw-CR class-wise cross-covariance class .



(a) Baseline (b) Dropout (c) BN

Figure 2: Scatter samples figure.

TODO:describe something regarding figure if required...

5 EXPERIMENTS - CIFAR-10/100

While performance improvement is not the main focus of this work, we provide additional test results with performance evaluations and show that the representation regularizers are useful for pushing the accuracy performance to the next level. In particular, we provide additional test results for CIFAR-100 and CIFAR-10 datasets (Krizhevsky & Hinton, 2009). For CIFAR-100, we have chosen a toy CNN architecture to confirm performance improvement of representation regularizers. Concurrently using two of the regularizers is experimented as well. For CIFAR-10, we have tested

Table 7: Error performance of regularizers (CIFAR-100)

Regularizer			Train error	Test error
Baseline		None	25.50	**56.02**
Penalty on weight		L1-weight	18.16	55.99
		L2-weight	33.75	54.93
Implicit method		Dropout (fc)	28.02	55.28
		Dropout (all)	79.28	80.08
		BN (fc)	28.28	55.33
		BN (all)	8.63	57.82
Penalty on representaion	Single	L1-rep	*98.93*	*99.00*
		VR	27.02	**53.66**
		CR	33.24	54.67
		cw-VR	22.85	54.15
		cw-CR	27.84	53.78
	Combination	VR + CR	13.88	54.68
		VR + cw-VR	19.43	56.12
		VR + cw-CR	28.53	54.94
		CR + cw-VR	21.11	**53.30**
		CR + cw-CR	18.05	54.75
		cw-VR + cw-CR	25.77	55.64
		L1-rep + VR	*98.93*	*99.00*
		L1-rep + CR	*98.93*	*99.00*
		L1-rep + cw-VR	*98.93*	*99.00*
		L1-rep + cw-CR	*98.93*	*99.00*

representation regularizers using ResNet that is known as one of the best performing deep neural networks for image data.

5.1 Combining multiple strategies: CIFAR-100

We use a toy CNN network for experimenting with CIFAR-100. The CNN network consists of four convolution layers and a fully connected layer, all with 100 hidden units. ReLU is used as the activation function. The second, third, and fourth convolution layers are followed by a max pooling layer. The last 10,000 instances of 50,000 training data were used as the validation data. Using the validation data, validation performance was evaluated for the regularizer weight values of $\{0.1, 0.01, 0.001, 0.0001\}$. The best weight value was found for each regularizer, and the test performance was evaluated for the fixed weight values. For representation regularizers, regularization was applied to the fully connected layer. The performance results are shown in Table 7. From the table, it can be seen that the test error is improved from baseline 56.02% to 53.66% by using single regularizer (VR) and to 53.30% by using two regularizers (CR and cw-VR). Therefore, 2.72% of improvement is achieved by the best performing regularizer combination. Aside from the performance improvement, it is interesting to observe that L1-rep consistently fails to train for the CIFAR-100 data. With 100 labels, too much sparsity might hurt. This is a plausible hypothesis considering that we have only 100 neurons to encode 100 labels. A shared use of neurons over multiple classes might be a better direction to pursue. In general, the relationship between the number of labels and the desired statistical properties of representation remains as a topic to be studied.

5.2 Performance improvement of ResNet-32

Residual network (ResNet) was first proposed by He et al. (2016). ResNet consists of multiple basic blocks that are serially connected, and shortcut connections to force residuals to be calculated. We apply five regularization strategies without modifying the ResNet-32 architecture. Regularization was applied to the output layer. Experimental results in Table 8 show that performance is improved over the state-of-the-art ResNet-32 model. This indicates that representation regularizers are compatible with ResNet, and most likely also with other state-of-the-art models.

Table 8: Error performance of regularizers on ResNet-32 (CIFAR-10)

Model	He et al.	Ours
ResNet-32	**7.51**	**7.39**
ResNet-32 + L1-rep		7.27
ResNet-32 + VR		7.22
ResNet-32 + CR		7.27
ResNet-32 + cw-VR		**7.17**
ResNet-32 + cw-CR		7.21

6 CONCLUSION

We have found...

Two things are needed. Explicit manipulation of statistical property Better understanding on how channel and noise are formed during the learning process.

(Therefore, statistical shaping should not be considered as exclusively for generalization. Penalty regularization is really for performance, generalization, and optimization (and possibly beyond).)

What is the role of regularization? Tuning?

What is happening during learning? Encoding is one thing, but noise and channel are being formed/designed as well.

Example of Gaussian code and Hamming code, where codes are designed for fixed channel, noise, transmit energy, and possibly encoder/decoder complexity instead of learned as representations as in DNN.

How to do learning? Encoding is one thing, but the choice of shaping direction affects the statistical properties of noise/interference as well. They need to be jointly considered, and it might be better to guarantee a simple statistical property for interference such that learning can be simple and focus on encoding only.

ACKNOWLEDGMENTS

To be added.

REFERENCES

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Soufiane Belharbi, Clement Chatelain, Romain Herault, and Sebastien Adam. Neural networks regularization through invariant features learning. *arXiv preprint arXiv:1709.01867*, 2017.

Brian Cheung, Jesse A Livezey, Arjun K Bansal, and Bruno A Olshausen. Discovering hidden factors of variation in deep networks. *arXiv preprint arXiv:1412.6583*, 2014.

Michael Cogswell, Faruk Ahmed, Ross Girshick, Larry Zitnick, and Dhruv Batra. Reducing overfitting in deep networks by decorrelating representations. 2016.

Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 315–323, 2011.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

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

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

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

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Jun-Bao Li, Jeng-Shyang Pan, and Shu-Chuan Chu. Kernel class-wise locality preserving projection. *Information Sciences*, 178(7):1825–1835, 2008.

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

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014.

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

Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Advances in Neural Information Processing Systems*, pp. 2074–2082, 2016.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

APPENDIX

A PERFORMANCE OF POPULAR STRATEGIES WHEN APPLIED TO EACH LAYER

Table 9: Error performance of popular regularizers - applied to each layer

Layer	Baseline	Penalty on weight		Implicit method	
		L1-weight	L2-weight	Dropout	BN
All		**2.90**±0.08	2.96±0.09	4.08±0.06	2.69±0.06
Output		3.02±0.15	2.96±0.06	2.99±0.18	2.97±0.08
Layer 5		2.98±0.05	2.99±0.13	2.80±0.08	3.04±0.09
Layer 4	**3.06**±0.15	2.98±0.08	2.98±0.09	**2.67**±0.05	2.84±0.15
Layer 3		3.04±0.09	3.03±0.18	**2.67**±0.16	2.94±0.13
Layer 2		2.91±0.05	2.76±0.16	2.70±0.08	2.84±0.16
Layer 1		2.93±0.05	**2.52**±0.10	3.07±0.07	**2.58**±0.07

B STATISTICAL PROPERTIES OF REPRESENTATION

Table 10: Evaluation of statistical properties(raw) - popular strategies

Property	Baseline	Penalty on weight		Implicit method	
		L1-weight	L2-weight	Dropout	BN
$\Omega_{L1\text{-}weight}$ (all)	9795.03	7504.60	8220.21	9461.52	9488.60
$\Omega_{L2\text{-}weight}$ (all)	607.46	459.85	502.71	576.60	792.24
$\Omega_{L1\text{-}rep}$	3.24×10^6	3.25×10^6	3.25×10^6	1.14×10^6	6.27×10^5
Ω_{VR}	865.69	851.24	860.34	307.64	86.59
Ω_{CR}	58178.00	54803.10	55650.20	8551.84	255.31
$\Omega_{cw\text{-}VR}$	2398.03	2327.79	2377.46	610.58	265.46
$\Omega_{cw\text{-}CR}$	63726.20	58891.60	60610.20	21795.60	193.26

Table 11: Evaluation of statistical properties(raw) - representation shaping regularizers

Property	All classes			Class-wise	
	L1-rep	VR	CR	cw-VR	cw-CR
$\Omega_{L1\text{-}weight}$ (all)	9975.62	9732.16	9826.06	9843.84	9772.89
$\Omega_{L2\text{-}weight}$ (all)	727.22	645.03	665.57	813.20	853.98
$\Omega_{L1\text{-}rep}$	38183.20	3.06×10^5	3.39×10^5	1.28×10^5	2.11×10^5
Ω_{VR}	6.26	7.85	8.43	1.78	3.40
Ω_{CR}	0.80	2.55	2.55	0.19	0.64
$\Omega_{cw\text{-}VR}$	5.34	16.91	22.15	0.69	1.97
$\Omega_{cw\text{-}CR}$	0.17	1.53	2.00	8.83×10^{-3}	0.04

C METRICS

Average number of activated classes
The number of activated labels in each unit is related to class-wise sparsity. Relu is activated when

the input of Relu is positive. Whether each unit is activated in a class is defined by whether the ratio of activated Relu exceeds the threshold. The number of activated classes means the number of classes that each unit is activated. The average of activation class is defined by the average of the number of activated classes for all units. The average of the activation class represents class-wise sparsity by measuring how many classes are activated for each units. I is a indicator function and N_u is the number of units.

$$Num_Act_InClass(i, m) = \sum_{n \in S_m} I(h_{i,n} > 0)$$

$$Act_Class(i, m) = I(\frac{Num_Act(i, m)}{|S_m|} > \textit{threshold})$$

$$Num_Act_Class(i) = \sum_m Act_Class(i, m)$$

$$Avg_Act_Class = \frac{\sum_i Num_Act_Class(i)}{N_u}$$

Ratio of dead units
Dead neurons is widely used to represent neurons that are not activated. However, the concept of dead neurons does not represent class-wise information. So, we defined the dead unit which is not activated on all classes. The dead unit indicates that it is not significant to any class. The ratio of dead units measures the ratio of dead units in all units. The lower the ratio of dead units is, the harder the information is coded in the units.

$$All_Class_Dead(i) = \sum_m (1 - Act_Class(i, m))$$

$$Ratio_Dead_Unit = \frac{\sum_i All_Class_Dead(i)}{N_u}$$

D VISUALIZATION OF REPRESENTATION

D.1 REPRESENTATION HISTOGRAMS OF 10 REGULARIZATION STRATEGIES

D.2 REPRESENTATION SCATTER PLOTS OF 10 REGULARIZATION STRATEGIES

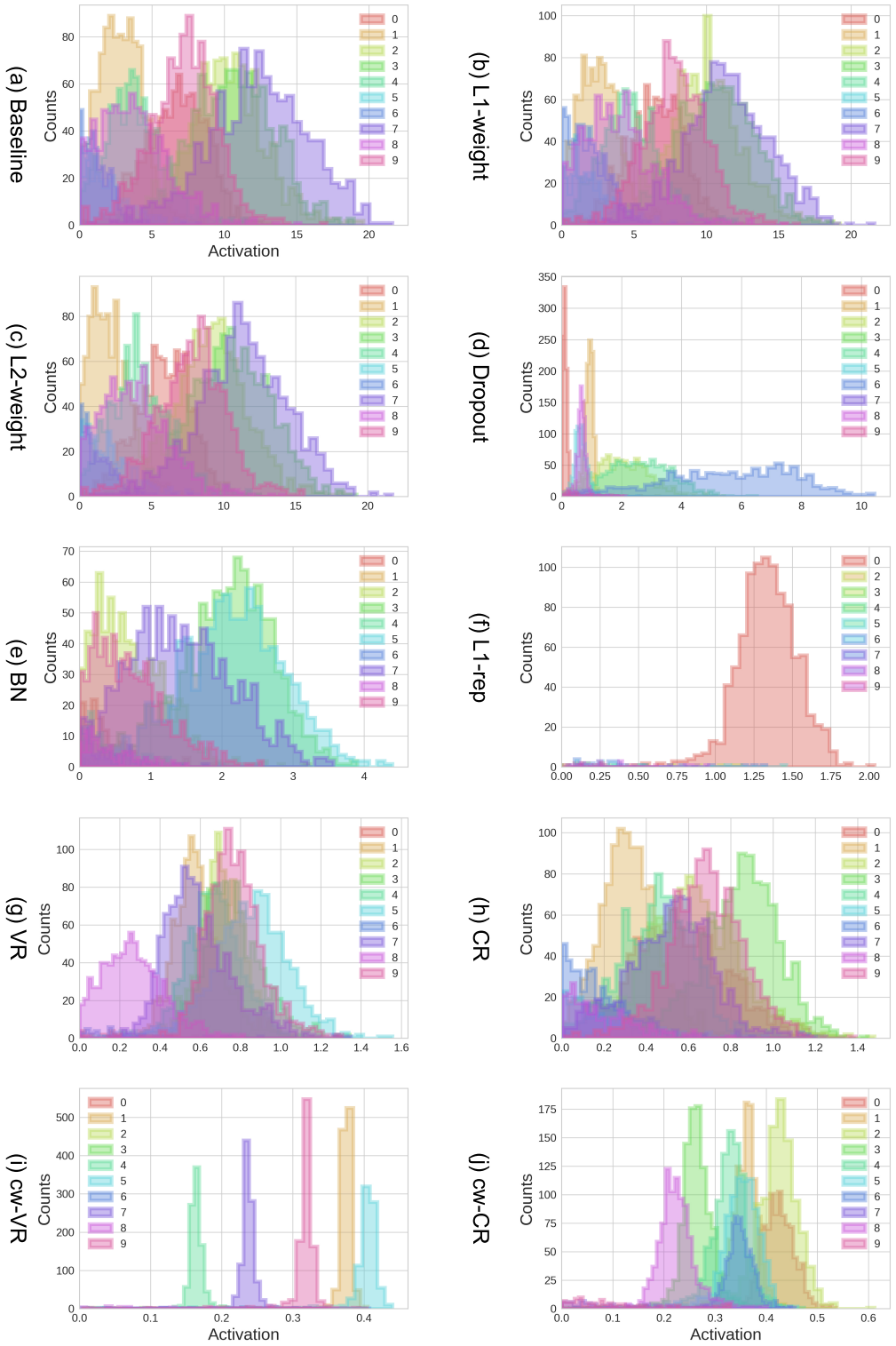


Figure 3: Histograms of of 10 regularization strategies.

TODO:describe plots here if required