SHADE: Information-Based Regularization for Deep Learning

Michael Blot, Thomas Robert, Nicolas Thome, Matthieu Cord

SHADE: Information-Based Regularization for Deep Learning

Michael Blot, Thomas Robert, Nicolas Thome, and Matthieu Cord

Sorbonne Université, CNRS, Laboratoire d'Informatique de Paris 6, LIP6, F-75005 Paris, France

Abstract

Regularization is a big issue for training deep neural networks. In this paper, we propose a new information-theory-based regularization scheme named SHADE for SHAnnon DEcay. The originality of the approach is to define a prior based on conditional entropy, which explicitly decouples the learning of invariant representations in the regularizer and the learning of correlations between inputs and labels in the data fitting term. Our second contribution is to derive a stochastic version of the regularizer compatible with deep learning, resulting in a tractable training scheme. We empirically validate the efficiency of our approach to improve classification performances compared to common regularization schemes on several standard architectures.

1 Introduction

Deep neural networks (DNN) have shown impressive state-of-the-art results in the last years on numerous tasks and especially on image recognition [19, 14]. In image classification particularly, deep convolutional neural networks, have demonstrated an outstanding ability to generalize, despite being able to completely learn the training set. Indeed, [34] experimentally demonstrate that generalization performance of DNN remains an open question, that usual machine learning tools, such as VC dimension [22] or Rademacher [6] complexity fail to explain. Further more, many regularization tricks such as weight decay [20], dropout [29] or batch normalization [15] seem crucial in the training of DNN, whereas they lake for theoretical foundations enabling to interpret their effect.

In this article, we study the possibility to design a regularization scheme that can be applied efficiently to deep learning and that has theoretical motivations. Our approach requires to define a regularization criterion, which is our first contribution. We claim that, for any model, the entropy of its intermediate representation, conditionally to the class variable, is a good criterion to apprehend the generalization potential of the model. More formally let's note $X \in \mathcal{X}$ the input variable, $C \in \mathcal{C}$ the class variable, w the model parameters with $Y = h(w, X)$ the (deep) representation of the input leading to the class prediction. Then, the objective is to penalize $H(Y \mid C)$, where H denotes the Shannon entropy measure (see [10] for definition). As explain in next section, the measure $H(Y \mid C)$ is perfectly suitable to quantify how invariant the representation Y is, accordingly to the underlying task of class prediction. This criterion also stands as a valid instantiate of the "Minimum Description Length Principle" [26], an interpretation of the Occam Razor.

Unfortunately, information measures are usually difficult to estimate when the number of data is low, compared to the size of the variable support space. As a second contribution, we propose an implementation of a tractable loss that proves to reduce our criterion when minimized. Indeed, based on an interpretation of the class information encoding within neurons activations, we assume that for every neuron exists a random Bernoulli variable that contains most of the class information. This variable ultimately enables to derive a batch-wise estimator of the entropy criterion, that is scalable and integrates easily in a stochastic gradient descent (SGD) optimization scheme. The resulting

loss, called SHADE for SHAnnon DEcay, has the advantage to be layer-wise and more particularly neuron-wise.

Finally, as a third contribution we provide extensive experiments on different datasets to motivate and to validate our regularization scheme.

2 Related work and motivation

Regularization in deep learning. For classification tasks, DNN training schemes usually use an objective function which linearly combines a classification loss $\ell_{\text{cls}}(w, X, C)$ – generally cross-entropy – and a regularization term $\Omega(w, X, C)$, with $\beta \in \mathbb{R}^+$, that aims at influencing the optimization toward a local minima with good generalization properties:

$$\mathcal{L}(w) = \mathbb{E}_{(X,C)}\big[\ell_{\text{cls}}(w, X, C) + \beta \cdot \Omega(w, X, C)\big] \tag{1}$$

For example, the weight decay (WD) loss [20] is supposed to penalize the complexity of the model. While there is strong motivations to WD use on linear models, in term of margins or in term of Lipschitz coefficient for instance, the extension of those theoretical results to DDN is not straightforward and the effects of WD on DNN's generalization performances is still not clear as demonstrated in [34].

Our SHADE regularization scheme belongs to this family as we construct a loss $\Omega_{SHADE}(h(X, w), C)$, that aims at influencing the optimization toward representations with low class conditioned entropy. We show in the experiments that SHADE loss has a positive effect on our theoretical criterion $H(Y \mid C)$, resulting in enhanced generalization performances.

Others popular regularization methods like [29, 32] deactivate part of the neurons during the training. Those methods, which tend to lower the dependency of the class prediction to a reduced set of features, is backed by some theoretical interpretations like [1, 12]. Other methods that add stochasticity to the training such as batch normalization or stochastic poolings [15, 11, 7], beside being comparable to data-augmentation, result in the addition of noise to the gradients during optimization. This property would enable the model to converge toward a flatter minima, that are known to generalize well as shown in [16]. More generally, stochastic methods tend to make the learned parameters less dependant on the training data, which guarantee tighter generalization bounds [5]. In this article we focus on another way to make DNN's models less dependant on the training data. By focusing on representation that are invariant to many transformations on the input variable, you make the training less dependant on the data. Having invariant representation is the motivation for our criterion $H(Y \mid C)$.

Quantifying invariance. Designing DNN models that are robust to variations on the training data and that preserve class information is the main motivation of this work. In the same direction, Scattering decompositions [21] are appealing transforms. They have been incorporated into adapted network architectures like [8]. However, for tasks like image recognition, it is very difficult to design an explicit modelling of all transformations a model should be invariant to.

Inversely, a criterion such as $H(Y \mid C)$ is agnostic to the transformations the representation should be invariant to, and is suitable to quantify how invariant it is in a context of class prediction. Indeed, a model that is invariant to many transformations will produce the same representation for different inputs, making it impossible to guess which input produced a given representation. This characteristic is perfectly captured by the entropy $H(X \mid Y)$ which represents the uncertainty in the variable X knowing its representation $Y = h(w, X)$. For instance, many works relate the reconstruction error to the entropy $H(X \mid Y)$ like in Fano's inequality [10] in the discrete case. A general discussion with illustrations on how to bound the reconstruction error with $H(X \mid Y)$ can be found in Appendix D. Thus, the bigger the measure $H(X \mid Y)$, the more invariant the representation. Let's now analyze the entropy of the representation Y when X is discrete and for a deterministic mapping $Y = h(w, X)$. We have:

$$H(Y) = I(X, Y) + \underbrace{H(Y \mid X)}_{=0 \text{ (determ.)}} = I(X, Y) = H(X) - H(X \mid Y). \tag{2}$$

$H(X)$ being fixed, $H(Y)$ is inversely related to $H(X \mid Y)$ making $H(Y)$ also a good measure of invariance.

However, considering the target classification task, we do not want two inputs of different classes to have the same representation but rather want to focus on intra-class invariance. Therefore, all this reasoning should be done conditionally to the class C, explaining final choice of $H(Y \mid C)$[1] as a measure of intra-class invariance.

Information-theory-based regularization. Many works like [25, 2] use information measures as regularization criterion. Still with the objective of making the trained model less dependant on the data, [2] has built a specific weight regularization but had to model the weight distribution which is not easy. The information criterion that is closer to ours is the one defined in the Information Bottleneck framework (IB) proposed in [31] that suggests to use mutual information $I(X, Y)$ ([10])[2] as a regularization criterion. [3] extends it in a variational context, VIB, by constructing a variational upper-bound of the criterion. Along with IB also come some theoretical investigations, with the definition of generalization bounds in [28]. Using mutual information as a regularizer is also closely connected to invariance since $I(X, Y)$ attempts at compressing as much information as possible from input data. In the case X is discrete and the representation mapping is deterministic ($H(Y \mid X) = 0$), our criterion is related to IB's trough the following development $I(X, Y) = H(Y) = I(C, Y) + H(Y \mid C)$. In a context of optimization with SGD, minimizing $H(Y \mid C)$ appears to be more efficient to preserve the term $I(C, Y)$, which represents the mutual information of the representation variable with the class variable and which must stay high to predict accurately C from Y.

Compressing the representation, without damaging the class information seems in fact to be a holy grail in machine learning. Our work, resulting in SHADE, goes in this direction.

3 SHADE: A new Regularization Method Based on $H(Y \mid C)$

In this section, we will further describe SHADE, a new regularization loss based on the conditional entropy $H(Y \mid C)$ designed to drive the optimization toward more invariant representation. We first show how to derive a layer-wise, neuron-wise criterion before developing a proper tractable estimate of the unit-wise criterion. In order to properly develop entropy inequalities it is necessary to suppose that X is a discrete variable of the space $\{0, ..., 255\}^{H \times W \times 3}$ with H and W respectively the height and width of the images. However it is common to consider X as a discrete quantization of a continuous natural variable, enabling to exploit some properties verified by continuous variables such as gradient computing[3].

3.1 A unit-wise criterion

Layer-wise criterion. A DNN is composed of a number L of layers that sequentially transform the input. Each one can be seen as an intermediate representation variable, noted Y_ℓ for layer $\ell \in \{1, .., L\}$, that is determined by the output of the previous layer $Y_{\ell-1}$ and a set of parameters $\mathbf{w}_\ell$. Each layer filters out a certain part of the information from the initial input X. Thus, the following inequalities can be derived from the data processing inequality in [10]:

$$H(Y_L \mid C) \leq H(Y_{L-1} \mid C) \leq ... \leq H(Y_1 \mid C) \leq H(X \mid C). \tag{3}$$

The conditional entropy of every layer is upper bounded by the conditional entropy of the previous layer. Similarly to the recommendation of [23], we apply this regularization to all the layers, using a layer-wise criterion $H(Y_\ell \mid C)$, and producing a global criterion[4]:

$$\Omega_{\text{layers}} = \sum_{\ell=1}^{L} \beta_\ell H(Y_\ell \mid C). \tag{4}$$

Where β_ℓ is a weighting term differentiating the importance of the regularization between layers. Those coefficient will be omitted in the following as in our experiments all β_ℓ are identical. Adjusting the values of the variables β_ℓ remains open for further research. It is illustrated in Fig. 1 where we see that $I(Y_l, C)$ (in green) remains constant while $H(Y_l \mid C)$ (in red) decreases.

[1] $H(Y \mid C) = H(X \mid C) - H(X \mid Y, C)$

[2] for all variable X, Y, $I(X, Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$

[3] A discussion on this topic can be found in [4]

[4] Confirming the intuition, in our experiments, regularizing all layer has proved to be more efficient than regularizing the final layer representation only

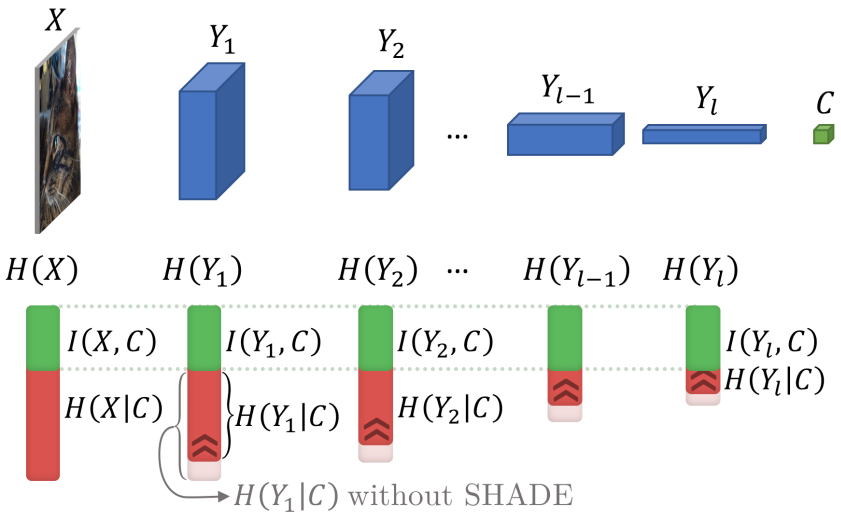


Figure 1: DNN architecture with corresponding layers' entropies, showing the layer-wise action of SHADE. Given that $H(Y_i) = I(Y_i, C) + H(Y_i \mid C)$, SHADE minimizes $H(Y_i \mid C)$ without affecting $I(Y_i, C)$.

Unit-wise criterion. Examining one layer ℓ, its representation variable is a random vector of coordinates $Y_{\ell,i}$ and of dimension D_ℓ: $Y_\ell = (Y_{\ell,1}, ..., Y_{\ell,D_\ell})$. The upper bound $H(Y_\ell \mid C) \leq \sum_{i=1}^{D_\ell} H(Y_{\ell,i} \mid C)$ enables to define a unit-wise criterion that SHADE seeks to minimize. For each unit i of every layer ℓ we design a loss $\Omega_{\text{unit}}(Y_{\ell,i} \mid C) = H(Y_{\ell,i} \mid C)$ that will be part of the global regularization loss:

$$\Omega_{\text{layers}} \leq \sum_{l=1}^{L} \sum_{i=1}^{D_\ell} \underbrace{H(Y_{\ell,i} \mid C)}_{\Omega_{\text{unit}}(Y_{\ell,i}|C)}. \tag{5}$$

For the rest of the paper, we use the notation Y instead of $Y_{\ell,i}$ for simplicity, since the layers and coordinates are all considered independently to define $\Omega_{\text{unit}}(Y_{\ell,i} \mid C)$.

3.2 Estimating the Entropy

In this section, we describe how to define a loss based on the measure $H(Y \mid C)$ with Y being one coordinate variable of one layer output. Defining this loss is not obvious as the gradient of $H(Y \mid C)$ with respect to the layer's parameters may be computationally intractable. Y has an unknown distribution and without modeling it properly it is not possible to compute $H(Y \mid C)$. Since $H(Y \mid C) = \sum_{c \in \mathcal{C}} p(c) H(Y \mid c)$, a direct approach would consist in computing $|\mathcal{C}|$ different entropies $H(Y \mid c)$, $1 \leq c \leq |\mathcal{C}|$. This means that, given a batch, the number of samples used to estimate one of these entropies is divided by $|\mathcal{C}|$ on average which becomes particularly problematic when dealing with a large number of classes such as the 1,000 classes of ImageNet. Furthermore, entropy estimators are extremely inaccurate considering the number of samples in a batch. For example, LME estimators of entropy in [24] converge in $\mathcal{O}(^{(\log K)^2}/_K)$ for K samples. Finally, most estimators require to discretise the space in order to approximate the distribution *via* a histogram. This raises issues on the bins definition considering that the variable distribution is unknown and varies during the training in addition to the fact that having a histogram for each neuron is computationally and memory consuming. Moreover, entropy estimators using kernel density estimation usually have a too high complexity ($\approx \mathcal{O}(K^2)$) to be applied efficient on deep learning models.

To tackle these drawbacks we propose the two following solutions: the introduction of a latent variable that enables to use more examples to estimate the conditional entropies; and a bound on the entropy of the variable by an increasing function of its variance to avoid the issue of entropy estimation with a histogram, making the computation tractable and scalable.

Latent code. First, considering a single neuron Y (before ReLU), the ReLU activation induces a detector behavior toward a certain pattern. If the pattern is absent from the input, the signal is zero; if it is present, the activation quantifies the resemblance with the pattern. We therefore propose to associate a binomial variable Z with each unit variable Y (before ReLU). This variable Z indicates if a particular pattern is present on the input ($Z = 1$ when $Y \gg 0$) or not ($Z = 0$ when $Y \ll 0$). It acts like a latent code from which the input X is generated like in variational models (*e.g.* [17]) or in generative models (*e.g.* [9]).

Furthermore, it is very likely that most intermediate features of a DNN can take similar values for inputs of different classes – this is especially true for low-level features. The semantic information provided by a feature is thus more about a particular pattern than about the class itself. Only the association of features allows determining the class. So Z represents a semantically meaningful factor about the class C. The feature value Y is then a quantification of the possibility for this semantic attribute Z to be present in the input or not.

We thus assume the Markov chain $C \to Z \to X \to Y$. Indeed, during the training, the distribution of Y varies in order to get as close as possible to a sufficient statistic of X for C (see definition in [10]). Therefore, we expect Z to be such that Y draws near a sufficient statistic of Z for C as well. By assuming the sufficient statistic relation $I(Y, C) = I(Y, Z)$ we get the equivalent equality $H(Y \mid C) = H(Y \mid Z)$, and finally obtain:

$$H(Y \mid C) = H(Y \mid Z) = \sum_{z \in \{0,1\}} p(z) H(Y \mid Z = z). \tag{6}$$

This modeling of Z as a Bernoulli variable (one for each unit) has the advantage of enabling good estimators of conditional entropy since we only divide the batch into two sets for the estimation ($z = 0$ and $z = 1$) regardless of the number of classes. The fact that most of Y information about C is contained in such a variable Z is validated in the experiments Sec. 4.4.

Variance bound. The previous trick allows computing fewer entropy estimates to obtain the global conditional entropy, therefore increasing the sample size used for each entropy estimation. Unfortunately, it does not solve the bin definition issue. To address this, we propose to use the following bound on $H(Y \mid Z)$, that does not require the definition of bins:

$$H(Y \mid Z) \leq \frac{1}{2} \ln \big(2\pi e \mathbb{V}\mathrm{ar}(Y \mid Z)\big). \tag{7}$$

This bound holds for any continuous distributions Y and there is equality if the distribution is Gaussian. For many other distributions such as the exponential one, the entropy is also equal to an increasing function of the variance. In addition, one main advantage is that variance estimators are much more robust than entropy estimators, converging in $\mathcal{O}(1/K)$ for K samples instead of $\mathcal{O}(\log(K)^2/K)$. The use of this bound is well justified in our case because the variable Y is the quantization of a continuous variable. Moreover, even if Y is discrete, this inequality still holds with respect to a term depending on the quantization steps.

The $\ln$ function being one-to-one and increasing, we only keep the simpler term $\mathbb{V}\mathrm{ar}(Y \mid Z)$ to design our final loss:

$$\Omega_{\mathrm{SHADE}} = \sum_{\ell=1}^{L} \sum_{i=1}^{D_\ell} \sum_{z \in \{0,1\}} p(Z_{\ell,i} = z \mid Y) \mathbb{V}\mathrm{ar}(Y \mid Z_{\ell,i} = z). \tag{8}$$

In the next section, we detail the definition of the differential loss, computed on a mini-batch, using $\mathbb{V}\mathrm{ar}(Y \mid Z)$ as a criterion.

Algorithm 1 Moving average updates: for $z \in \{0,1\}$, p^z estimates $p(Z=z)$ and μ^z estimates $\mathbb{E}(Y \mid Z=z)$

Initialize: $\mu^0 = -1, \mu^1 = 1, p^0 = p^1 = 0.5, \lambda = 0.8$
for each mini-batch $\{y^{(k)}, k \in 1..K\}$ **do**
  for $z \in \{0,1\}$ **do**
    $p^z \leftarrow \lambda p^z + (1-\lambda)\frac{1}{K}\sum_{k=1}^{K} p(z \mid y^{(k)})$
    $\mu^z \leftarrow \lambda \mu^z + (1-\lambda)\frac{1}{K}\sum_{k=1}^{K} \frac{p(z \mid y^{(k)})}{p^z} y^{(k)}$
  end for
end for

3.3 Instantiating SHADE

For one unit of one layer, the previous criterion writes:

$$\mathbb{V}\mathrm{ar}(Y \mid Z) = \int_{\hat{Y}} p(y) \sum_{z\in\{0,1\}} p(z \mid y)\big(y - \mathbb{E}(Y \mid z)\big)^2 \,\mathrm{d}y \tag{9}$$

$$\approx \frac{1}{K}\sum_{k=1}^{K}\left[\sum_{z\in\{0,1\}} p(z \mid y^{(k)})\big(y^{(k)} - \mathbb{E}(Y \mid z)\big)^2\right]. \tag{10}$$

The quantity $\mathbb{V}\mathrm{ar}(Y \mid Z)$ can be estimated with Monte-Carlo sampling on a mini-batch of K input-target pairs $\left\{(x^{(k)}, c^{(k)})\right\}_{1\le k\le K}$ of intermediate representations $\left\{y^{(k)}\right\}_{1\le k\le K}$ as in Eq. (10).

$p(Z \mid y)$ is interpreted as the probability of presence of attribute Z on the input, and should clearly be modeled such that $p(Z=1 \mid y)$ increases with y. The more similarities between the input and the pattern represented by y, the higher the probability of presence for Z. We suggest using[5]:

$$p(Z=1 \mid y) = 1 - e^{-ReLU(y)} \qquad p(z=0 \mid y) = e^{-ReLU(y)}.$$

For the expected values $\mu^z = \mathbb{E}(Y \mid z)$ we use a classic moving average that is updated after each batch as described in Algorithm 1. Note that the expected values are not changed by the optimization since translating a variable has no influence on its entropy.

The concrete behavior of SHADE can be interpreted by analyzing its gradient as described in Appendix E.

For this proposed instantiation, our SHADE regularization penalty takes the form:

$$\Omega_{\mathrm{SHADE}} = \sum_{\ell=1}^{L}\sum_{i=1}^{D_\ell}\sum_{k=1}^{K}\sum_{z\in\{0,1\}} p\left(Z_{\ell,i} = z \,\middle|\, y_{\ell,i}^{(k)}\right)\left(y_{\ell,i}^{(k)} - \mu_{\ell,i}^{z}\right)^2. \tag{11}$$

We have presented a regularizer that is applied neuron-wise and that can be integrated into the usual optimization process of a DNN. The additional computation and memory usage induced by SHADE is almost negligible (computation and storage of two moving averages per neuron). For comparison, SHADE adds half as many parameters as batch normalization does.

4 Experiments

4.1 Image Classification with Various Architectures on CIFAR-10

We perform image classification on the CIFAR-10 dataset, which contains 50k training images and 10k test images of 32×32 RGB pixels, fairly distributed within 10 classes (see [18] for details). Following the architectures used in [34], we use a small Inception model, a three-layer MLP, and an

[5]Other functions have been experimented with similar results

Table 1: Classification accuracy (%) on CIFAR-10 test set.

	MLP	AlexNet	Inception	ResNet
No regul.	64.68	83.25	91.21	92.95
Weight decay	66.52	83.54	92.87	93.84
Dropout	66.70	85.95	91.34	93.31
SHADE	**70.45**	**85.96**	**93.56**	**93.87**

AlexNet-like model with 3 convolutional layers (64 filters of size 3×3) + max pooling and 2 fully connected layers with 1000 neurons for the intermediate variable. We also use a ResNet architecture from [33] (k=10, N=4). Those architectures represent a large family of DNN and some have been well studied in [34] within the generalization scope. For training, we use randomly cropped images of size 28×28 with random horizontal flips. For testing, we simply center-crop 28×28 images. We use momentum SGD for optimization (same protocol as [34]).

We compare SHADE with two regularization methods, namely *weight decay* [20] and *dropout* [29]. For all architectures, the regularization parameters have been cross-validated to find the best ones for each method and the obtained accuracies on the test set are reported in Table 1. Find more details on the experiments protocol in A.

We obtain the same trends as [34], which shows a small improvement of 0.29% with weight decay on AlexNet. The improvement with weight decay is slightly more important with ResNet and Inception (0.79% and 1.66%). In our experiments dropout improves significantly generalization performances only for AlexNet and MLP. It is known that the use of batch normalization and only one fully connected layer lowers the benefit of dropout, which is in fact not used in [14].

We first notice that for all kind of architectures that the use of SHADE significantly improves the generalization performances, 5.77% for MLP, 2.71% for AlexNet, 2.35% for Inception and 0.92% for ResNet. It demonstrates the ability of SHADE to regularize the training of deep architectures. Finally, SHADE shows better performances than dropout and weight decay on all architectures.

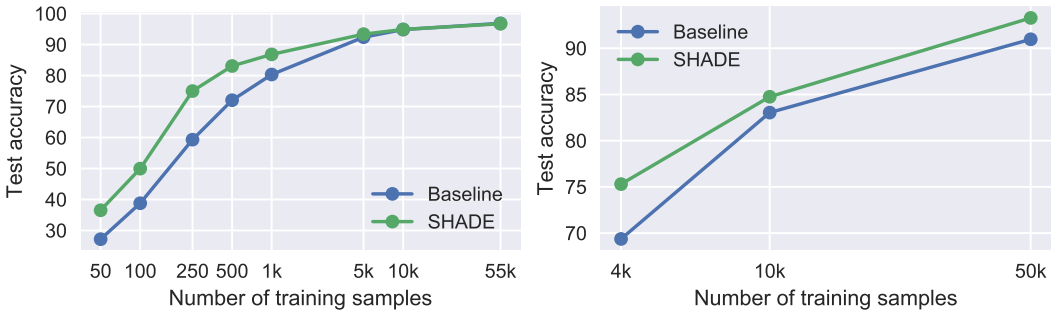
4.2 Large Scale Classification on ImageNet

In order to experiment SHADE regularization on a very large scale dataset, we train on ImageNet [27] a WELDON network from [30] adapted from ResNet-101. This architecture changes the forward and pooling strategy by using the network in a fully-convolutional way and adding a max+min pooling, thus improving the performance of the baseline network. We used the **pre-trained weights of ResNet-101** (from the torchvision package of PyTorch) giving performances on the test set of **77.56%** for top-1 accuracy and 93.89% for top-5 accuracy. Provided with the pre-trained weights, the **WELDON architecture** obtains **78.51%** for top-1 accuracy and 94.65% for top-5 accuracy. After fine tuning the network using **SHADE** for regularization we finally obtained **80.14%** for top-1 accuracy and 95.35% for top-5 accuracy for a concrete improvement. This demonstrates the ability to apply SHADE on very large scale image classification successfully.

4.3 Training with a Limited Number of Samples

When datasets are small, DNN tend to overfit quickly and regularization becomes essential. Because it tends to filter out information and make the network more invariant, SHADE seems to be well fitted for this task. To investigate this, we propose to train DNN with and without SHADE on CIFAR-10 and MNIST-M with different numbers of samples in the training set.

First, we tested this approach on the digits dataset MNIST-M [13]. This dataset consists of the MNIST digits where the background and digit have been replaced by colored and textured information (see Fig. 2c for examples). The interest of this dataset is that it contains lots of unnecessary information that should be filtered out, and is therefore well adapted to measure the effect of SHADE. A simple convolutional network has been trained with different numbers of samples of MNIST-M and the optimal regularization weight for SHADE have been determined on the validation set (see training details in Appendix B). The results can be seen on Figure 2a. We can see that especially for small numbers of training samples (< 1000), SHADE provides an important gain of 10 to 15% points



(a) Results for MNIST-M

(b) Results for CIFAR-10

(c) Examples of MNIST-M images misclassified by the baseline and correctly classified using SHADE, both trained with 250 samples.

Figure 2: Results when training with a limited number of samples in the training set for MNIST-M and CIFAR-10 with and without SHADE.

Table 2: Classification accuracy (%) on CIFAR-10 test set with binarized activation.

MLP	baseline 64.68	layer 3 64.92	layer 2 62.45	layer 1 61.13
AlexNet	baseline 83.25	last 4 82.71	layer 2 82.38	layer 1 82.01
Inception	baseline 91.34	layer 21 91.41	layer 10 90.88	layer 1 90.21
ResNet	baseline 93.24	layer 56 92.67	layer 25 92.09	layer 1 91.99

over the baseline. This shows that SHADE helped the model in finding invariant and discriminative patterns using less data samples.

Additionally, Figure 2c shows samples that are misclassified by the baseline model but correctly classified when using SHADE. These images contain a large amount of intra-class variance (color, texture, etc.) that is not useful for the classification tasks, explaining why adding SHADE, that encourages the model to discard information, allows important performance gains on this dataset and especially when only few training samples are given.

Finally, to confirm this behavior, we also applied the same procedure in a more conventional setting by training an Inception model on CIFAR-10. Figure 2b shows the results in that case. We can see that once again SHADE helps the model gain in performance and that this behavior is more noticeable when the number of samples is limited, allowing a gain of 6% when using 4000 samples.

4.4 Further experiments: exploration on the latent variable

SHADE is based on the intuition that the class information encoded within a neuron is mostly contained in a binary latent variable noted Z. To justify this assumption we propose an experiments that studies trained networks neuron variables. In this experiment we work on the possibility to transform the ReLU activation function of a layer into a binary activation function that can only take two values. By exhibiting such a binary activation which does not affect the accuracy, we show that we can summarize the class information of a neuron into a binary variable and still get the same prediction accuracy as with the continuous ReLU activation. The experiment have been done on CIFAR-10 dataset with the same networks used in Sec. 4.1.

Binary activation. We have replaced the ReLU activation function on a chosen layer of a trained network with a binary activation function. The binary function is $\overline{Y^+}\mathbb{1}(Y \geq \overline{Y^+})$ where $\overline{Y^+}$ stands for the average value of the *positive* variable values before any activation function. After replacing the activation function we fine tune the layers on the top of the chosen layer, in order to adapt the top of the network to the new values and we report the obtained accuracies on the Table 2 for the different networks and different layers. We note that the differences in accuracy are very small losses. This means that for a given layer, the information that is used for the class prediction can be sum up in a binary variable confirming the existence of a binary latent variable containing most of the class information that is exploited by the rest of the network. The fact that this apply for all layers of the network is consistent with the application of SHADE loss for all layers. Note that this binary activation could be further researched to improve the modeling integrated in SHADE.

5 Conclusion

In this paper, we introduced a new regularization method for DNN training, SHADE, which focuses on minimizing the entropy of the representation conditionally to the labels. This regularization aims at increasing the intra-class invariance of the model while keeping class information. SHADE is tractable, adding only a small computational overhead when included into an efficient SGD training. We show that our SHADE regularization method significantly outperforms standard approaches such as weight decay or dropout with various DNN architectures on CIFAR-10. We also validate the scalability of SHADE by applying it on ImageNet. The invariance potential brought out by SHADE is further illustrated by its ability to ignore irrelevant visual information (texture, color) on MNIST-M. Finally, we also highlight the increasing benefit of our regularizer when the number of training examples becomes small. Furthermore there is no doubt that the information-theory-based interpretation of SHADE, from which it has been established, allows for further improvements of SHADE for future work.

References

[1] Achille A. and Soatto S. Information Dropout: learning optimal representations through noisy computation. *ArXiv*, 2016.

[2] Alessandro Achille and Stefano Soatto. Emergence of invariance and disentanglement in deep representations. In *ArXiv*, 2017.

[3] A. A. Alemi, I. Fischer, J. V Dillon, and K. Murphy. Deep variational information bottleneck. In *International Conference on Learning Representations*, 2017.

[4] Rana Ali Amjad and Bernhard C. Geiger. How (not) to train your neural network using the information bottleneck principle. *ArXiv*, 2018.

[5] Xu Aolin and Raginsky Maxim. Information-theoretic analysis of generalization capability of learning algorithms. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 2524–2533. Curran Associates, Inc., 2017.

[6] Bartlett, Peter L, and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 2002.

[7] Graham Benjamin. Fractional max-pooling. *arXiv preprint arXiv:1412.6071*, 2014.

[8] Joan Bruna and Stephane Mallat. Invariant scattering convolution networks. In *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, 2013.

[9] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, Jun 2016.

[10] T. Cover and J. Thomas. Elements of information theory. *Wiley New York*, 1991.

[11] Zeiler Matthew D and Rob Fergus. Stochastic pooling for regularization of deep convolutional neural networks. *arXiv preprint arXiv:1301.3557*, 2013.

[12] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the International Conference on Machine Learning*, 2017.

[13] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *ICML*, pages 1180–1189, 2015.

[14] 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*, 2016.

[15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *Journal of Machine Learning Research*, 2016.

[16] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large batch training for deep learning: generalization gap and sharp minima. In *ICLR*, 2017.

[17] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

[18] A. Krizhevsky. *Learning multiple layers of features from tiny images*. PhD thesis, Computer Science Department University of Toronto, 2009.

[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, 2012.

[20] Anders Krogh and John A. Hertz. A simple weight decay can improve generalization. In *Advances in Neural Information Processing Systems*, 1992.

[21] Stephane Mallat. Group invariant scattering. In *Communications on Pure and Applied Mathematics*, 2012.

[22] Vapnik Vladimir N and Chervonenkis A Ya. On the uniform convergence of relative frequencies of events to their probabilities. In *Measures of Complexity*. Springer, 1972.

[23] Tishby Naftali and Zaslavsky Noga. Deep learning and the information bottleneck principle. In *Information Theory Workshop (ITW)*. IEEE, 2015.

[24] Liam Paninski. Estimation of entropy and mutual information. *Neural Computation*, 2003.

[25] Gabriel Pereyra, George Tucker, Jan Chorowski, Lukasz Kaiser, and Geoffrey E. Hinton. Regularizing neural networks by penalizing confident output distributions. In *International Conference on Learning Representations Workshop*, 2017.

[26] J. Rissanen. Modeling by shortest data description. *Automatica*, 1978.

[27] Olga Russakovsky*, Jia Deng*, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challeng. *International Journal of Computer Vision*, 2015.

[28] O. Shamir, S. Sabato, , and N. Tishby. Learning and generalization with the information bottleneck. *Theoretical Computer Science*, 2010.

[29] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 2014.

[30] Durand Thibaut, Thome Nicolas, and Matthieu Cord. WELDON: Weakly Supervised Learning of Deep Convolutional Neural Networks. In *CVPR*, 2016.

[31] N. Tishby, F. C. Pereira, and W. Bialek. The information bottleneck method. *Annual Allerton Conference on Communication, Control and Computing*, 1999.

[32] Li Wan, Matthew D. Zeiler, Sixin Zhang, Yann LeCun, and Rob Fergus. Regularization of neural networks using dropconnect. In *ICML (3)*, volume 28 of *JMLR Proceedings*, pages 1058–1066. JMLR.org, 2013.

[33] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *arXiv*, 2016.

[34] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *International Conference on Learning Representations*, 2017.

A Experiments details on CIFAR-10

Optimization. For all experiments the learning rate is initialize and a multiplicative decay is apply on it after every batches. The momentum is constant and setted to 0.9. We detail here the initial learning rate and the decay for every networks used in the format (initial learning rate value, decay): mlp (,), alexnet (,), inception (,), resnet(,)

Hyperparameter tuning For weight decay and SHADE, the optimal regularization weight of each model has been chosen to maximize the accuracy on the validation sets. We tried the values $\{[1, 5].10^{-i}, i = 1..7\}$. For the dropout we have apply it on the two last layer of every networks. The optimal activation probabilities for each model has been chosen among $\{(n/10, m/10), n = 1..7,\ m = 3..10\}$ to maximize the accuracy on the validation sets.

B Experiments details on MNIST-M

Dataset splits and creation. To create MNIST-M, we kept the provided splits of MNIST, so we have 55,000 training samples, 5,000 validation samples, and 10,000 test samples. Each digit of MNIST is processed to add color and texture by taking a crop in images from BST dataset. This procedure is explained in [13].

Subsets of limited size. To create the training sets of limited size N, we keep $N/10$ (since there are 10 classes) randomly picked samples from each class. When increasing N we keep the previously picked samples so the training samples for $N = 100$ are a subset of the ones for $N = 250$. The samples chosen for a given value of N are the same across all models trained using this number of samples.

Image preprocessing. The only preprocessing applied to the input images is that their values are rescaled from $[0, 1]$ to $[-1, 1]$.

Optimization. For the training, we use mini-batch of size 50 and use Adam optimizer with the recommended parameters, *i.e.* $\lambda_r = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$.

Hyperparameter tuning. For weight decay and SHADE, the optimal regularization weight of each model (for each value of N) has been chosen to maximize the accuracy on the validation sets. We tried the values $\{10^{-i}, i = 1..7\}$.

Model architecture. The model have the following architecture:

- 2D convolution ($64 \times 5 \times 5$ kernel, padding 2, stride 1) + ReLU
- MaxPooling 2×2
- 2D convolution ($64 \times 3 \times 3$ kernel, padding 1, stride 1) + ReLU
- MaxPooling 2×2
- 2D convolution ($64 \times 3 \times 3$ kernel, padding 1, stride 1) + ReLU
- MaxPooling 2×2
- Fully connected (1024 inputs, 10 outputs) + SoftMax

C Experiments details on Imagenet

The fine tuning in the experiment section 4.2 has been done with momentum-SGD with a learning rate of 10^{-5} and a momentum of 0.9 and a batch size of 16 images. It took 8 epochs to converge.

D Entropy bounding the reconstruction error

In section 2 we highlight a link between the entropy $H(X \mid Y)$ and the difficulty to recover the input X from its representation Y. Here we exhibit a concrete relation between the reconstruction error, that quantifies the error made by a strategy that predicts X from Y, and the conditional entropy. This relation takes the form of an inequality, bounding the error measure in the best case (with the reconstruction strategy that minimizes the error) by an increasing function of the entropy. We note $\hat{x}(Y) \in \mathcal{X}$ the reconstruction model that tries to guess X from Y.

The discrete case In case the input space is discrete, we consider the zero-one reconstruction error for one representation point Y: $\varepsilon(Y) = E_X[\mathbb{1}(\hat{x}(Y) \neq X)]$. This is the probability of error when predicting $\hat{x}(Y)$ from Y. The function that minimizes the expected error $\mathcal{E} = \mathbb{E}_Y(\varepsilon(Y))$ is $\hat{x}(Y) = \arg\max_{x \in \mathcal{X}} p(x \mid Y)$ as shown in Proof 1. We derive the following inequality:

$$\frac{H(X \mid Y) - 1}{\log |\mathcal{X}|} \leq \mathcal{E} \leq 1 - 2^{-H(X|Y)}. \tag{12}$$

The left side of the inequality uses Fano's inequality in [10], the right one is developed in proof 2. This first inequalities show how the reconstruction error and the entropy are related. For very invariant representations, it is hard to recover the input from Y and the entropy of $H(X \mid Y)$ is high.

Besides, there can be an underlying continuity in the input space and it could be unfair to penalize predictions close to the input as much as predictions far from it. We expose another case below that takes this proximity into account.

The continuous case In the case of convex input space and input variable with continuous density, we consider the 2-norm distance as reconstruction error: $\varepsilon(Y) = \mathbb{E}_{X|Y}\left[||X - \hat{x}(Y)||_2^2\right]$. This error penalizes the average distance of the input and its reconstruction from Y. The function that minimizes the expected error $\mathcal{E} = \mathbb{E}_Y[\varepsilon(Y)]$ is the conditional expected value: $\hat{x}(y) = \mathbb{E}[X \mid Y = y]$. Then $\mathcal{E} = \mathbb{V}\mathrm{ar}(X \mid Y)$. Helped by the well-known inequality $H(X \mid Y) \leq \frac{1}{2}\ln(2\pi e \mathbb{V}\mathrm{ar}(X \mid Y))$ we obtain:

$$\frac{e^{2H(X|Y)}}{2\pi e} \leq \mathcal{E}. \tag{13}$$

Here again, notice that a high entropy $H(X \mid Y)$ implies a high reconstruction error in the best case.

D.1 Proof 1

We have

$$\mathcal{E} = \int_{\hat{Y}} p(y)\varepsilon(y)\,\mathrm{d}y \tag{14}$$

$$= \int_{\hat{Y}} p(y)p(X \neq \hat{x}(y) \mid y)\,\mathrm{d}y \tag{15}$$

$$= \int_{\hat{Y}} p(y)(1 - p(\hat{x}(y) \mid y))\,\mathrm{d}y. \tag{16}$$

Since

$$p(\hat{x}(y) \mid y) \leq p(\arg\max_{x \in X} p(x \mid y) \mid y), \tag{17}$$

the reconstruction that minimizes the error is $\hat{x}(y) = \arg\max_{x \in X} p(x \mid y)$. However, this is theoretical because in most cases $p(x \mid Y)$ is unknown.

D.2 Proof 2

We have:

$$\log(1-\mathcal{E}) = \log\left(\int_{\hat{Y}} p(y)(1-\varepsilon(y))\,\mathrm{d}y\right) \tag{18}$$

$$= \log\left(\int_{\hat{Y}} p(y)p(\hat{x}(y) \mid y)\right)\,\mathrm{d}y \tag{19}$$

$$\geq \int_{\hat{Y}} p(y)\log(p(\hat{x}(y) \mid y))\,\mathrm{d}y \tag{20}$$

$$= \int_{\hat{Y}} p(y)\int_{\mathcal{X}} p(x \mid y)\log(p(\hat{x}(y) \mid y))\,\mathrm{d}x\,\mathrm{d}y \tag{21}$$

$$\geq \int_{\hat{Y}} p(y)\int_{\mathcal{X}} p(x \mid y)\log(p(x \mid y))\,\mathrm{d}x\,\mathrm{d}y \tag{22}$$

$$= -H(X \mid Y). \tag{23}$$

The Eq. (20) is obtained using Jensen inequality and Eq. (22) is obtained using the result of Proof 1.

Thus,

$$\mathcal{E} \leq 1 - 2^{-H(X|Y)}. \tag{24}$$

E SHADE Gradients

Here is studied the influence of SHADE on a gradient descent step for a single neuron Y of a single layer and for one training sample X. The considered case of a linear layer, we have: $Y = \mathbf{w}^\top X + b$.

The gradient of Ω_{SHADE} with respect to $\mathbf{w}$ is:

$$\begin{aligned}\nabla_{\mathbf{w}}\Omega_{\mathrm{SHADE}} &= (\delta_1+\delta_2)\mathbf{x}\\ \text{with } \delta_1 &= \sigma'(y)\big((y-\mu^1)^2-(y-\mu^0)^2\big)\\ \text{and } \delta_2 &= 2\sigma(y)(y-\mu^1)+2(1-\sigma(y))(y-\mu^0).\end{aligned}$$

With $\sigma(y) = p(Z=1|y)$ which has positive derivative. We can interpret the direction of this gradient by analyzing the two terms δ_1 and δ_2 as follows:

- $\boldsymbol{\delta_1}$: If $(y-\mu^0)^2$ is bigger than $(y-\mu^1)^2$ that means that y is closer to μ^1 than it is to μ^0. Then δ_1 is positive and it contributes to increasing y meaning that it increases the probability of Z being from mode 1. In a way it increases the average margin between positive and negative detections. Note that if there is no ambiguity about the mode of Z meaning that $\sigma(y)$ or $1-\sigma(y)$ is very small then this term has negligible effect.
- $\boldsymbol{\delta_2}$: This term moves y toward the μ^z of the mode that presents the bigger probability. This has the effect of concentrating the outputs around their expectancy depending on their mode to get sparser activation.