Network as Regularization for Training Deep Neural Networks: Framework, Model and Performance

Kai Tian[†], Yi Xu[†], Jihong Guan[‡], Shuigeng Zhou[†*]
[†]Shanghai Key Lab of Intelligent Information Processing, and School of Computer Science, Fudan University, China
[‡]Department of Computer Science & Technology, Tongji University, China
[†]{ktian14, yxu17, sgzhou}@fudan.edu.cn;[‡]jhguan@tongji.edu.cn

Abstract

Despite powerful representation ability, deep neural networks (DNNs) are prone to over-fitting, because of over-parametrization. Existing works have explored various regularization techniques to tackle the over-fitting problem. Some of them employed soft targets rather than one-hot labels to guide network training (*e.g.* label smoothing in classification tasks), which are called target-based regularization approaches in this paper. To alleviate the over-fitting problem, here we propose a new and general regularization framework that introduces an auxiliary network to dynamically incorporate guided semantic disturbance to the labels. We call it *N*etwork *a*s *R*egularization (*NaR* in short). During training, the disturbance is constructed by a convex combination of the predictions of the target network and the auxiliary network. These two networks are initialized separately. And the auxiliary network is trained independently from the target network, while providing instance-level and class-level semantic information to the latter progressively. We conduct extensive experiments to validate the effectiveness of the proposed method. Experimental results show that NaR outperforms many state-of-the-art target-based regularization methods, and other regularization approaches (*e.g. mixup*) can also benefit from combining with NaR.

Introduction

The great success in many real-world applications (*e.g.* computer vision, natural language processing) has shown the powerful representation learning capability of deep neural networks (DNNs) over a variety of data (Szegedy et al. 2016; Sutskever, Vinyals, and Le 2014). However, they are often extremely large in depth or width with the number of parameters being far more than the number of training examples available. Thus, over-fitting is a big challenge in DNN optimization, which could seriously degrade the networks' generalization power.

In the past decade, various regularization techniques have been developed to tackle the over-fitting problem, from ℓ_1, ℓ_2 regularization schemes that are derived from traditional linear models, to dropout (Srivastava et al. 2014), batch normalization (Ioffe and Szegedy 2015) and these proposed specifically for DNNs (Szegedy et al. 2016; Wan et al. 2013; Liu et al. 2018). According to where these methods act on the networks, we can subsume the existing regularization methods roughly into four types: *weight-based*, *input-based*, *activation-based* and *target-based*.

Specifically, *weight-based* approaches impose constraints to network weights to reduce the hypothesis space; *input-based* methods add noise or impose transformations to samples for training more robust networks; *activation-based* methods such as dropout randomly drop neurons to alleviate feature co-adaptation. These regularization techniques have been extensively studied in the literature, while there are relatively less works on target-based regularization.

The principle of target-based regularization methods is that instead of focusing on the primary class (*i.e.*, the ground truth), the model also pays some attention to the other classes. This makes the model more tolerant and less over-confidence (Szegedy et al. 2016; Pereyra et al. 2017; Yang et al. 2018). In essence, they are equivalent to injecting prior knowledge to the labels. The differences among them mainly lie in two aspects: what kind of prior is used and how to adopt the prior to the labels. For example, label smoothing and confidence penalty penalize the predicted distribution by a uniform distribution (Szegedy et al. 2016; Pereyra et al. 2017); (Xie et al. 2016) proposed a method to replace a small portion of the ground truth labels with incorrect ones in each iteration; Soft target approaches such as teacher-student optimization and bootstrapping (Hinton, Vinyals, and Dean 2015; Reed et al. 2015) offer additional training signals that contain secondary information (Yang et al. 2018; Furlanello et al. 2018) to the network. Most of these methods above adopt the prior knowledge as a regularization term in the original loss function, while teacher-student optimization needs to train a teacher model first, which causes additional time consumption.

In this paper, to tackle the overfitting problem, we propose an efficient and general target-based regularization framework by dynamically incorporating guided semantic disturbance to the labels while avoid multiple training stages that decrease training efficiency. We call the framework *N*etwork *a*s *R*egularization (*NaR* in short). One major source of the se-

[*]Correspondence author.

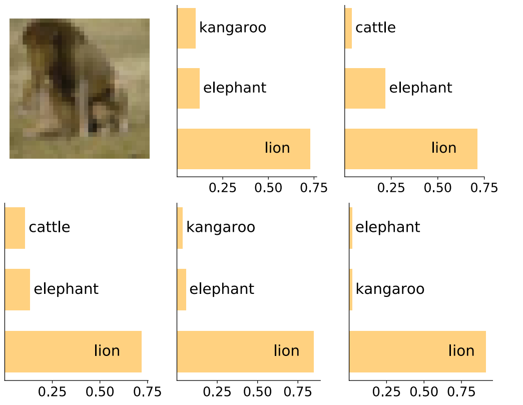

Figure 1: **Top-left:** a training example from CIFAR-100 dataset with ground-truth label *lion*. **From second to sixth**: the semantic labels generated by our framework in different training iteration. The *x-axis* denotes the probability of each class, only top-3 possible classes were drawn.

mantic disturbance is an additional network which is referred to as auxiliary network (AN). The training strategy is simple: at each iteration, we corrupt the label of each example by a convex combination of the ground-truth label and the predictions of the target network and the auxiliary network. Then, we compute the loss function between the target model's prediction and the corrupted labels, and employ a gradient based method to optimize it. Meanwhile, AN is also trained simultaneously but with the loss between the ground-truth labels and its prediction individually. After training, AN is removed and will not incur any more *computational burden*.

The intuition behind this framework is that a network can learn instance-level semantic information even when it is not fully converged. Therefore, inspired by the idea of curriculum learning (Bengio et al. 2009) that easy samples should be learned first, we argue that the semantic information learned during the training process of a network can be used to guide another one. Instead of giving the answer directly, the auxiliary network provides continuous hints till the target network converges. As shown in Fig. 1, the label distributions of the target network are changed iteratively according to AN's guidance. In addition, the dynamic soft labels can avoid memorization in overparameterized models (Zhang et al. 2017a).

We conduct extensive experiments to evaluate our framework with a variety of network architectures from plain CNN to wide residual network (Zagoruyko and Komodakis 2016) for image classification, and LSTM network for language modeling. We also explore how the architecture of the auxiliary network affects the performance of the target network. Moreover, we show that our framework is compatible with some other types of regularization methods such as data augmentation, which can also benefit from our framework.

Related Work

Regularization is a practical technique to solve the over-fitting problem in many machine learning algorithms. Some popular regularization methods such as ℓ_2-norm are effective for linear models (Crambes et al. 2009). Early works of DNNs borrowed regularization techniques directly from traditional machine learning algorithms. As NNs need lots of data, data augmentation is an important trick to train a good model on small datasets. Meanwhile, early stopping can efficiently prevent over-fitting (Bengio 2012). Recently, researchers have also developed some specific regularization techniques for neural networks, *e.g.* Dropout (Srivastava et al. 2014), DropConnect (Wan et al. 2013), batch normalization (Ioffe and Szegedy 2015) and parameter sharing (Ruder 2017). Most of these methods are compatible to our framework, so instead of comparing with them, we simply combine some of these techniques with our model.

Target-based Regularization Label smoothing estimates the marginalized effect of label-dropout during training, reducing over-fitting by preventing a network from assigning full probability to each training example (Zhang et al. 2017a). (Xie et al. 2016) proposed to corrupt a small portion of labels in a mini-batch manner and showed that it can help to regularize neural networks. Teacher-student optimization (Hinton, Vinyals, and Dean 2015) is another kind of target-based regularization methods, by utilizing the predictions of the teacher model (usually a high-capacity network) to soften the labels for the student model. Recently, (Furlanello et al. 2018) proposed born-again networks (BAN) and revealed the phenomenon that when the model capacity of the student network is identical to that of the teacher, the former surprisingly outperforms the latter model after a few generations.

Two-way Online Distillation Online distillation is a new research topic that is attracting increasing attention recently. (Zhang et al. 2018) exploited a two-way knowledge transfer and demonstrated that training two networks simultaneously has a potential performance improvement over conventional distillation. (Anil et al. 2018) investigated the benefits of codistillation in the case of large-scale distributed training. The loss employed in codistillation is to match the class posterior in a peer-to-peer manner. Each student in the cohort is treated equally. (Wang et al. 2018) proposed KDGAN to integrate knowledge distillation with GAN. In their paper, the teacher model and the student learn from each other along with an adversarial loss. Our framework is different from codistillation, as the auxiliary model in our method serves as a regularization term and it is independent from the target network. Moreover, the capacity of AN may be lower than the target network, which is conflict to the idea of distillation.

Collaborative Learning Applying two models to the same task was also explored previously. Co-teaching (Han et al. 2018) has been proposed to train robust models by selecting clean labelled samples with two networks collaboratively. (Batra and Parikh 2017) proposed cooperative learning to learn multiple models jointly for the same task but in different input attributes. These methods are different from NaR, as both models communicate with each other while in NaR the auxiliary network is independent from the target network during training.

Network as Regularization

Given a dataset $\mathcal{D} = \{x_i, y_i\}_{n=1}^N$ with N examples, $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$ and a neural network $f(x;\theta)$ with parameter θ (we may omit θ for simplicity), the standard maximum

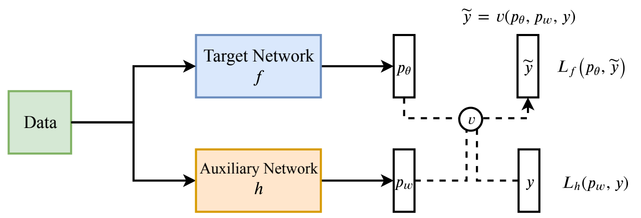

Figure 2: The NaR framework. The dotted lines denote numerical operations (no gradient back-propagation). The target network f is optimized with noisy labels $\widetilde{y}$ that contain the disturbance from the auxiliary network. The function $v(\cdot)$ denotes an aggregation operation. The auxiliary network is trained with the ground-truth label y independently.

likelihood estimates θ^* by minimizing the empirical loss over the training examples:

$$\theta^* = \arg\min_\theta \sum_{i=1}^{N} \ell(y_i, f(x_i;\theta)) \tag{1}$$

For classification task, $\ell(\cdot,\cdot)$ is the cross entropy loss and simply minimizing the cross entropy loss may result in overfitting as the loss considers only the ground-truth label, which will lead the model to assign full probability to the ground-truth label. From geometrical perspective, minimizing cross entropy will force NNs to map all samples of the same class to one point or close to the that point, *e.g.* $\underbrace{[1, 0, ..., 0]}_{C}$, C is the number of classes. This is not desirable for a loss function that is expected to generalize well, as it does not consider the inter-class and intra-class sample variations. Meanwhile, there is also ambiguity and noise in the data. The hard-coded labels are not informative, and deep neural networks may be too lazy to learn and just simply memorize the data (Zhang et al. 2017a).

To solve such a problem, we consider to map different examples to different points in the label space. This is consistent with the hypothesis of manifold learning (Roweis and Saul 2000; Bengio, Monperrus, and Larochelle 2006), the difference is that we focus on the supervised learning problem. Instead of injecting noise to the labels as in previous works (Szegedy et al. 2016; Xie et al. 2016), we propose to introduce semantic disturbance via exploiting the semantic information learned by an auxiliary model. Besides, although static and informative labels are helpful, the learning complexity is also fixed, the NNs can still memorize the data in such setting. However, the dynamically generated semantic labels in NaR can alleviate such problem.

Incorporating Semantic Disturbance to Labels

As shown in Fig. 2, we adopt an auxiliary network to guide the training of the target network. The target network is trained with disturbed labels and the auxiliary network is trained with the ground-truth labels. Specifically, suppose the target network $f(x;\theta)$ outputs the posterior probability of a sample x over ground-truth class c as $p_\theta(c|x)$:

$$p_\theta(c|x) = f(x;\theta) = \frac{\exp(z^c)}{\sum_{k=1}^{C}\exp(z^k)} \tag{2}$$

where z is the logits or unnormalized log probability. Similarly, let $p_w(c|x)$ be the output of the auxiliary model $h(x;w)$, where w is the parameter of $h(x)$. In order to introduce disturbance to the labels, we construct the noise ξ as a convex combination of the predictive distributions of the two models:

$$\xi = \alpha p_\theta(x) + (1-\alpha)p_w(x), \quad 0 \le \alpha < 1 \tag{3}$$

Such a combination is similar to the idea of *mixup* (Zhang et al. 2017b), however, the combination in mixup can be seen as a two-hot embedding without any semantic information. In our case, ξ contains both class-level and instance-level semantic information, and it makes our framework more flexible at a cost of introducing an extra hyperparameter.

The label with semantic disturbance is defined as

$$\widetilde{y} = (1-\lambda)y + \lambda\xi \tag{4}$$

where $\lambda \in [0, 1)$ is the hyper-parameter that leverages the ground-truth label and the disturbance. Note that ξ is a numerical vector and there is no gradient propagation to $f(x;\theta)$ and $h(x;w)$. This characteristic enables us to generalize our framework with a bunch of machine learning algorithms for the candidate of $h(\cdot)$. Meanwhile, the training processes of $f(x)$ and $h(x)$ are independent except that for each iteration $f(x;\theta)$ needs to get the prediction $p_w(x)$ from $h(x;w)$.

Finally, the objective function of NaR is defined as below:

$$L_f = H(\widetilde{y}, p_\theta(x)) = (1-\lambda)H(y, p_\theta(x)) + \lambda H(\xi, p_\theta(x)) \tag{5}$$

where $H(p,q)$ is the cross entropy of two distributions. And the objective function for $h(x)$ is

$$L_h = H(y, p_w(x)). \tag{6}$$

NaR can also be applied to regression tasks. Similar to Eq. (3) and Eq. (4), we can construct ξ with the predictions of $f(x;\theta)$ and $h(x;w)$ and apply the corresponding loss function (*e.g.* min-squared error (MSE) loss). Please refer to a toy example in our experiments.

One key problem in NaR is how to select the auxiliary network. Intuitively, high-capacity models can provide more information than low-capacity ones. However, these models tend to have more parameters and result in over-fitting quickly. Thus, the over-confidence issue will deteriorate the semantic information. Besides, how to evaluate a model's capacity and select a proper one is also a challenge. Inspired by the observation in BAN that distilling a teacher model to a student with an identical architecture can still improves the student's performance, We suggest to use the *same* network architecture as the target network. And empirical results also show that such configuration is reasonable.

Different from teacher-student optimization that each model is trained sequentially, $f(\theta)$ and $h(w)$ can be trained simultaneously and in parallel as $h(w)$ is independent from the target network. So the time consumption is almost the same as conventional training, although NaR introduces an

Algorithm 1: Training procedure of NaR

```
Input: Training set $\mathcal{D}$, hyper-parameters for $h$ and $f$;
/* Training */
Initialize: $\theta$ and $w$ with different initial conditions;
Repeat:
    Randomly sample data $\boldsymbol{x}$ from $\mathcal{X}$;
    1: Update the predictions $p_\theta(x)$ and $q_w(x)$ of $\boldsymbol{x}$ through a forward pass;
    2: Compute the disturbance $\xi$ and the new label $\widetilde{\mathbf{y}}$(Eqs. (3),(4));
    3: Compute the loss function of target network and auxiliary model (Eqs. (5),(6));
    4: Compute the gradient and update $\theta, w$ by SGD algorithm.
Until: converged
/* Inference */
Remove auxiliary model and deploy $f(x;\theta)$.
```

additional model to train. As the computation of ξ needs a forward pass of $h(x)$ on x, the mini-batch orders are the same to both models at each iteration to save time. Besides, the optimization algorithm used for both networks are gradient-based. The detailed optimization procedure of NaR is presented in Alg.1.

Relationship with Teacher-Student Training and Label Smoothing

Here, we discuss the relationship between NaR and other target-based regularization methods, especially label smoothing regularization (LSR) and distillation-based regularization.

For label smoothing, we replace the observed ground-truth labels y with noisy labels $\widehat{y} = v(y, \zeta)$, where $v(\cdot)$ is the noising function and ζ is a data-independent random vector, usually selected as a uniform distribution. The objective function of LSR is defined as below:

$$L_{\text{LSR}} = H(\widehat{y}, p_\theta(x)) = (1-\epsilon)H(y, p_\theta(x)) + \epsilon H(u, p_\theta(x)) \tag{7}$$

where u is a uniform distribution, ϵ is the probability that we replace the ground truth label with u.

From Eq. (7), we can see that label smoothing heavily penalizes the prediction distribution of each sample to be a prior distribution (*e.g.* uniform) that could fail to capture the intrinsic class-level semantics.

Here, we illustrate that **label smoothing regularization is a special case of NaR**. Comparing Eq. (5) and Eq. (7), we can set $\alpha = 0$ and the learning rate of the auxiliary network $h(x; w)$ to zero. Initialize the weights of all layers except the last one of h to be all-zero matrices and the biases to nonzero vectors, the weights and bias terms of the last layer to be a small constant (*e.g.* 0.01). Then, for each iteration, $\xi = q_w(x)$ will be uniform. So Eq. (5) degenerates to Eq. (7).

Different from LSR, knowledge distillation provides some semantic information with the cues learned by a high-capacity model and utilizes these additional information to train a better low-capacity student model. Rather than aiming at model compression, (Furlanello et al. 2018) proposed born-again neural network (BAN) that sequentially transfers knowledge from a well-trained model to new student with the identical capacity. In such training mechanism, BAN consistently improves its performance. As we empirically make AN have the same architecture as the target network, we compare BAN with NaR.

Formally, the k-th model is trained with the knowledge transferred from the (k-1)-th generation. The objective of BAN is defined as

$$L_{\text{BAN}} = (1-\beta)H(t, p_\theta^k(x)) + \beta H(p_\theta^{k-1}(x), p_\theta^k(x)) \tag{8}$$

where $p_\theta^{k-1}(x)$ and $p_\theta^k(x)$ represent the prediction of the (k-1)-th and k-th generation respectively, β is a coefficient for the knowledge transfer that is often set as 1 in BAN.

There are some drawbacks with BAN: (1) It needs to be trained in a sequential mode that undoubtedly increases the computation time. (2) The semantic information provided by the teacher model is static during training, thus it is vulnerable to the over-fitting issue. (3) For high-capacity models, the well-trained teacher can provide rare additional information to the next generation.

Similarly, we can **recover BAN from NaR**, if we set $\alpha = 0$ and $h(x)$ be a pre-trained model with the same architecture of $f(x; \theta)$.

Very interestingly, from the analysis above, we can conclude that the optimization process of NaR starts from label noise and ends at BAN.

Generally, NaR keeps the merits of existing target-based regularization methods while discards their drawbacks. Briefly, the disturbance in NaR is dynamic and semantic that can help the target network to avoid over-fitting and improve its performance. At the beginning of training, AN does not learn any valuable information from the data, the prediction of AN is nearly equivalent to random guess. As the learning proceeds, AN consistently improves its performance on the data and thus it is capable to learn the semantic information such as secondary information (Yang et al. 2018). The norm of the disturbance in the learning process is reduced from large to small. Finally, at the end of training, the disturbance is nearly static, which can be seen as the status of BAN.

Experiments

To evaluate the performance of NaR and compare it with state-of-the-art target-based regularization methods, we conduct experiments on three datasets, including CIFAR-10, CIFAR-100 and PTB. We also explore whether NaR is meaningful for regression task on a toy example. All the networks are implemented in Pytorch v1.0 and all experiments are carried out with two NVIDIA TITAN Xp. Our code is free available [1].

A Toy Example of Regression Task

We first qualitatively evaluate the semantic disturbance of the proposed method on a one-dimensional toy regression dataset. This dataset was used in (Lakshminarayanan, Pritzel, and Blundell 2017; Hernández-Lobato and Adams 2015), and consists of 20 training examples drawn from $y = x^3 + \epsilon$ where $\epsilon \sim \mathcal{N}(0, 3^2)$.

[1]https://github.com/codeforNaR/NaR

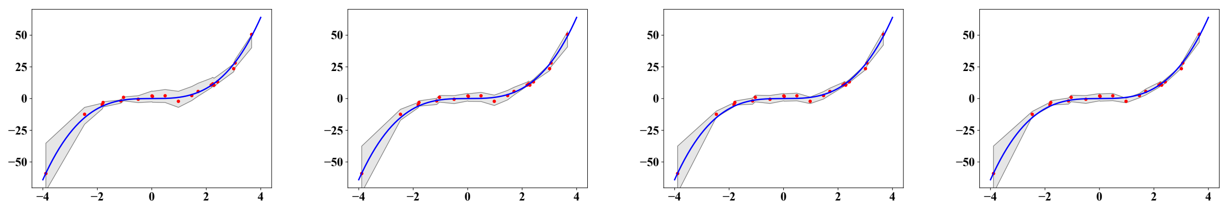

Figure 3: Results on a toy regression task: x-axis denotes x. On the y-axis, the blue line is the *ground truth*, the red dots are observed noisy data points and the gray lines correspond to the mean along with three standard deviations of noisy labels with 5 runs. Both the target network and the auxiliary network are trained with MSE, and from left to right correspond to different training epochs (step size is 50). The results show that for some outliers such as the leftmost example, NaR provides a new target with quite large disturbance even when the model is close to convergence.

For the target network and the auxiliary network, we use one hidden layer neural network with the hidden size 50 and 20, respectively. We run each experiment 5 times and save the intermediate uncertain targets for the examples and plot the mean of the new targets along with three standard deviation (gray shadow in Fig. 3). Similar to classification, the disturbance in the early training stage is large as the auxiliary network is not able to fit the data. However, when the training converges, the disturbance in densely-distributed area (samples larger than -2) is small, while for some outliers the disturbance remains large. Although this is a toy example, we can still validate the intuition that the auxiliary network is able to provide semantic disturbance to the target network. Specifically, in regression task, the disturbance can reduce the estimation error because the data always contain noise.

Multi-class Image Classification

Datasets. The **CIFAR-10** and **CIFAR-100** datasets consist of 32 $\times$ 32 color images containing objects from 10 and 100 classes, respectively, both have 50,000 images in the train set and 10,000 images in the test set.

Evaluation metric. Top-1 classification error rate is used as performance metric for both datasets. We run each setting with 5 times and report the average performance with standard deviation.

Networks. We compare a bunch of networks with different model capacities. A plain CNN with 6 convolution layers and 2 fully connected layers is used for simple architecture validation (Liu et al. 2018), we denote it as PlainCNN-6 for convenience.The other networks include ResNet-32 (He et al. 2016a), PreResNet-110 (He et al. 2016b) and Wide ResNet (WRN-28-10) (Zagoruyko and Komodakis 2016). The number of parameters of all these networks are reported in Table 1. Note that most of the parameters in PlainCNN-6 are from the fully connected layers.

Implementation details. Following the experimental settings of (Zagoruyko and Komodakis 2016), we use SGD with Nesterov momentum, and set the initial learning rate as 0.1, momentum as 0.9 and batch size as 128. As PlainCNN and WRN are two overparameterized networks, the weight-decay is set to 5e-4 and the learning rate dropped by 0.2 (0.1 for PlainCNN) at 60, 120 and 160 epochs, and we train totally 200 epochs. For ResNet-32 and PreResNet-110, the

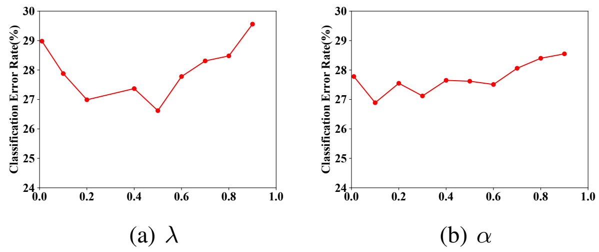

Figure 4: Sensitivity analysis of hyper-parameters on CIFAR-100 with ResNet-32.

weight-decay is 1e-4, the learning rate dropped by 0.1 at 150 and 225 epochs, and the total training epoch is 300. We use horizontal flips and random crops with padding size 4 for data augmentation. The standard data preprocess is adopted by transforming the pixel value to [0,1] and subtract the mean and divided by the std value for each pixel. For heterogeneous settings used for the target network and the auxiliary network, the total training epoch is decided by the target network. For example, if ResNet-32 is used as the auxiliary network for PlainCNN-6, the total training epoch is 200.

Hyperparameter analysis. There are two hyperparameters in *NaR*. Here we evaluate how the selection of hyperparameters affects the performance. We vary λ and α from 10^{-2} to 0.9 on CIFAR-100 with ResNet-32. The performance are shown in Fig. 4. Based on the results in Fig. 4, we set $\lambda = 0.5$ and $\alpha = 0.1$ for the following experiments.

Results on CIFAR. All results are summarized in Table 2. We only present the results of comparing NaR with label smoothing and BAN on CIFAR datasets as the other target-based regularization methods such as confidence penalty and Disturblabel (Xie et al. 2016) can not improve the performance when data augmentation is applied (Pereyra et al. 2017).

From Table 2 we can conclude that NaR outperforms LSR and BAN on both datasets with all these network architectures. The improvements on CIFAR-10 are not as significant as on CIFAR-100, the reason is that the classes of the same superclass in CIFAR-100 are more correlated with each other than the classes in CIFAR-10. It is surprising to note that PlainCNN with NaR outperforms ResNet-32 trained with

Table 1: The number of parameters of different networks on CIFAR-100 dataset.

Networks	PlainCNN-6	ResNet-32	PreResNet-110	WRN-28-10
# Parameters	2.2M	0.5M	1.7M	36.5M

Table 2: Evaluation of NaR on CIFAR datasets and comparison with several state-of-the art target-based regularization methods. BAN-m means the m-th generation of BAN.

Networks	CIFAR10			CIFAR100		
	PlainCNN	ResNet32	WRN	PlainCNN	ResNet32	WRN
Baseline	8.20±0.23	7.41±0.21	3.98±0.08	32.99±0.16	31.05±0.20	18.82±0.10
LSR	7.51±0.12	6.53±0.13	4.32±0.20	31.26±0.14	29.38±0.08	18.47±0.08
BAN-2	6.95±0.10	6.74±0.08	3.88±0.05	29.87±0.11	29.45±0.08	18.40±0.06
BAN-3	**6.64±0.08**	6.54±0.07	3.78±0.08	29.35±0.09	29.21±0.10	18.29±0.08
NaR	**6.64±0.12**	**6.21±0.10**	**3.70±0.08**	**28.78±0.10**	**27.82±0.07**	**17.30±0.06**

BAN for 3 generations on CIFAR-100.

Effect of the model capacity of auxiliary networks. To investigate how the model capacity of the auxiliary network affects the performance of the target network, we design experiments by using the networks in Table. 1 as the target network and the auxiliary network, respectively. The results on both CIFAR datasets are shown in Table 3. We can see that using PreResNet-110 as the auxiliary network achieves the best performance when the target network uses residual networks. However, WRN-28-10 underperforms ResNet-32 and PreResNet when it acts as AN. The reason is that WRN is a very high-capacity network and it can provide less disturbance than the light-weight networks. Besides, PlainCNN-6 has more parameters than ResNet-32 but it underperforms ResNet-32, because most of its parameters belong to the fully connected layers. Meanwhile, as we mentioned before, using identical architecture for the target network and AN usually achieves satisfactory results.

Language Modeling

To further evaluate the effectiveness of our method, we conduct word-level language modeling experiments on Penn Three-bank dataset (PTB) (Marcus, Santorini, and Marcinkiewicz). For simplicity, we used the similar experimental settings as in (Pereyra et al. 2017). Briefly, a 2-layer, 1500-unit LSTM with 65% dropout applied on all non-recurrent connections is employed as the target network. The only difference is that we multiply the learning rate by 0.25 when there is no performance improvement on validation set. We compare our method with label noise, label smoothing (LSR), confidence penalty (CP) and BAN, and use linear search for setting their hyperparameters. We also present the results without using regularization technique as the baseline. We use two NaR settings for the auxiliary network: 1) a smaller network has only half of the hidden units of the target network, which is denoted as *NaR-small*; 2) the auxiliary network uses the same architecture as the target network, which is denoted as *NaR-large*.

As shown in Table 4, CP significantly outperforms LSR and label noise. With the help of semantic soft targets, BAN can achieve slightly better result than CP, and NaR-small

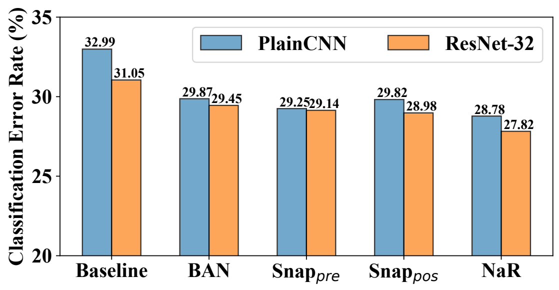

Figure 5: Results of snapshots regularization on CIFAR-100 dataset.

achieves the best results on both validation and test set. These results indicate that for overparameterized models, a small-capacity auxiliary network may be helpful to alleviate overfitting issue.

Replacing NaR with Snapshots Regularization

As mentioned before, the progressive disturbance incorporation from a raw auxiliary network will help the target network to learn better than from a mature auxiliary network. To validate this claim, we replace NaR with snapshots regularization. Briefly, instead of constructing the disturbance from the auxiliary network that is trained synchronously with the target network, we consider the situations that the disturbance is advanced or delayed. In other words, we construct the disturbance from an earlier or later iteration of the AN. Such experimental setting is basically similar to snapshot distillation (Yang et al. 2019). The difference lies in that we just borrow semantic information from an additional model rather than from the earlier epoch of the target network.

Denote the l-th iteration as the current iteration, we use the snapshot of the auxiliary network at the s_l-th iteration to construct the disturbance in Eq. (3). We consider two settings of s_l: 1) snap$_{pre}$, using a previous snapshot of the current iteration, *i.e.*, $s_l = \lfloor \frac{l}{S} \rfloor S$ where S is a constant interval between two snapshots. 2) Snap$_{pos}$, using a following snapshot of the current iteration, that is, $s_l = (\lfloor \frac{l}{S} \rfloor + 1)S$, which means we use a more mature (or better) model (compared to the target

Table 3: The test error rate results when using different auxiliary networks to regularize the target network on CIFAR-10 and CIFAR-100. Each row indicates a kind of target network architectures, and Columns 3-7 represent different choices of the auxiliary network. For comparison, the 3rd column *Single* presents the baseline performance of each target network.

Dataset	Network Types	Single	PlaincCNN-6	ResNet-32	PreResNet-110	WRN-28-10
CIFAR-10	PlainCNN-6	8.20±0.23	6.64±0.12	6.54±0.11	6.48±0.08	**6.24±0.06**
	Resnet-32	7.41±0.21	6.6±0.10	6.21±0.08	**6.12±0.11**	6.14±0.08
	PreResNet-110	5.11±0.11	4.57±0.07	4.24±0.06	**4.15±0.06**	4.44±0.08
	WRN-28-10	3.98±0.08	3.72±0.06	3.69±0.05	**3.50±0.06**	3.70±0.08
CIFAR-100	PlainCNN-6	32.99±0.16	28.78±0.08	**27.52±0.08**	28.89±0.10	29.03±0.12
	Resnet-32	31.05±0.20	28.93±0.13	27.82±0.05	**27.68±0.06**	27.75±0.10
	PreResNet-110	23.22±0.14	22.56±0.08	21.75±0.12	**21.48±0.07**	21.51±0.05
	WRN-28-10	18.82±0.10	18.73±0.04	18.25±0.08	17.9±0.10	**17.30±0.06**

Table 4: Validation and test perplexity for word-level Penn Treebank.

Methods	Validation	Test
Baseline	80.55	77.26
Label noise	79.87	77.67
LSR	79.08	76.98
CP	77.35	74.93
BAN-2	77.54	74.81
NaR-small	**76.78**	**73.77**
NaR-large	77.90	74.44

Table 5: Results of combining NaR with AT and mixup on CIFAR-10 and CIFAR-100 datasets.

Networks	CIFAR-10		CIFAR-100	
	PlainCNN	ResNet-32	PlainCNN	ResNet-32
AT	8.83	7.93	33.11	32.71
AT+NaR	**7.17±0.11**	**6.53±0.13**	**30.26±0.16**	**29.19±0.10**
mixup	7.28	6.77	32.35	29.22
mixup+NaR	**6.22±0.21**	**5.82±0.12**	**27.13±0.13**	**26.87±0.09**

network) to provide semantic information.

We carry out experiments on CIFAR-100 dataset with PlainCNN-6 and ResNet-32. The results are shown in Fig. 5. Both settings outperform BAN while underperform NaR. This suggests that incorporating disturbance to the labels in a progressive manner can improve the performance of NNs.

Combining NaR with Adversarial Training and mixup

To demonstrate that NaR can also work with other regularization techniques, especially some popular data augmentation tricks, we conduct experiments that NaR cooperates with adversarial training (AT) (Szegedy et al. 2014; Goodfellow, Shlens, and Szegedy 2015) and mixup (Zhang et al. 2017b). There are several methods to generate adversarial examples, in this paper we use the fast gradient sign method proposed in (Goodfellow, Shlens, and Szegedy 2015). It has been shown that using adversarial examples for data augmentation can improve classification robustness (Goodfellow, Shlens, and Szegedy 2015). Mixup (Zhang et al. 2017b) constructs a new examples by linear combination of two existing examples and their labels.

We choose two network architectures: PlainCNN-6 and ResNet-32 to conduct the experiments on CIFAR datasets. We first evaluate the AT and mixup methods individually on the two datasets with these two networks respectively, and take their results as the baselines. Then, we combine AT and mixup training with NaR respectively. The results are shown in Table 5. It is remarkable that NaR can improve the performance of the network along with AT. Though mixup alone can significantly boost the performance of neural networks, and much performance improvement can still be achieved when cooperating with NaR.

Conclusion and Future Work

In this paper we propose a novel framework for incorporating semantic disturbance to labels by training an auxiliary network along with the target network. The disturbance from the auxiliary network provides instance-level and class-level semantic information as the examples in the same class have different label distributions. By incorporating these information to the labels progressively, the target network can achieve better generalization capability. We show that some target-based regularization methods are special cases of our framework. Empirical results validate the effectiveness and advantages of dynamic semantic disturbance and the promising potential of combining this framework with other regularization techniques.

In summary, NaR provides a flexible and powerful approach for DNNs to handle over-fitting by learning from dynamic and semantic labels. In the future, we will consider using traditional machine learning algorithms (*e.g.* random forest) to replace the auxiliary network, and explore the application of our framework to complex regression tasks such as human pose recognition.

Acknowledgement. Jihong Guan was supported in part by the Program of Science and Technology Innovation Action of Science and Technology Commission of Shanghai Municipality under Grant 17511105204, and in part by the Shanghai Municipal Commission of Economy and Informatization under Grant 18XI-05.

References

Anil, R.; Pereyra, G.; Passos, A.; Ormandi, R.; Dahl, G. E.; and Hinton, G. E. 2018. Large scale distributed neural network training through online distillation. In *ICLR*.

Batra, T., and Parikh, D. 2017. Cooperative learning with visual attributes. *arXiv preprint arXiv:1705.05512*.

Bengio, Y.; Louradour, J.; Collobert, R.; and Weston, J. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, 41–48. ACM.

Bengio, Y.; Monperrus, M.; and Larochelle, H. 2006. Non-local estimation of manifold structure. *Neural Computation* 18(10):2509–2528.

Bengio, Y. 2012. Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade*. Springer. 437–478.

Crambes, C.; Kneip, A.; Sarda, P.; et al. 2009. Smoothing splines estimators for functional linear regression. *The Annals of Statistics* 37(1):35–72.

Furlanello, T.; Lipton, Z. C.; Tschannen, M.; Itti, L.; and Anandkumar, A. 2018. Born again neural networks. In *International Conference on Machine Learning*.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2015. Explaining and harnessing adversarial examples. In *ICLR*.

Han, B.; Yao, Q.; Yu, X.; Niu, G.; Xu, M.; Hu, W.; Tsang, I.; and Sugiyama, M. 2018. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Advances in Neural Information Processing Systems*, 8527–8537.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016a. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–778.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016b. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, 630–645. Springer.

Hernández-Lobato, J. M., and Adams, R. 2015. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, 1861–1869.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

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

Lakshminarayanan, B.; Pritzel, A.; and Blundell, C. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, 6402–6413.

Liu, W.; Lin, R.; Liu, Z.; Liu, L.; Yu, Z.; Dai, B.; and Song, L. 2018. Learning towards minimum hyperspherical energy. In *Advances in Neural Information Processing Systems*, 6222–6233.

Marcus, M.; Santorini, B.; and Marcinkiewicz, M. A. Building a large annotated corpus of english: The penn treebank. *Computational linguistics* 19(2):313–330.

Pereyra, G.; Tucker, G.; Chorowski, J.; Kaiser, Ł.; and Hinton, G. 2017. Regularizing neural networks by penalizing confident output distributions. In *ICLR Workshop*.

Reed, S.; Lee, H.; Anguelov, D.; Szegedy, C.; Erhan, D.; and Rabinovich, A. 2015. Training deep neural networks on noisy labels with bootstrapping. In *ICLR Workshop*.

Roweis, S. T., and Saul, L. K. 2000. Nonlinear dimensionality reduction by locally linear embedding. *science* 290(5500):2323–2326.

Ruder, S. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

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

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, 3104–3112.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; and Fergus, R. 2014. Intriguing properties of neural networks. In *ICLR*.

Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2818–2826.

Wan, L.; Zeiler, M.; Zhang, S.; Le Cun, Y.; and Fergus, R. 2013. Regularization of neural networks using dropconnect. In *International Conference on Machine Learning*, 1058–1066.

Wang, X.; Zhang, R.; Sun, Y.; and Qi, J. 2018. Kdgan: knowledge distillation with generative adversarial networks. In *Advances in Neural Information Processing Systems*, 775–786.

Xie, L.; Wang, J.; Wei, Z.; Wang, M.; and Tian, Q. 2016. Disturblabel: Regularizing cnn on the loss layer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4753–4762.

Yang, C.; Xie, L.; Qiao, S.; and Yuille, A. 2018. Knowledge distillation in generations: More tolerant teachers educate better students. *arXiv preprint arXiv:1805.05551*.

Yang, C.; Xie, L.; Su, C.; and Yuille, A. L. 2019. Snapshot distillation: Teacher-student optimization in one generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2859–2868.

Zagoruyko, S., and Komodakis, N. 2016. Wide residual networks. In *Proceedings of the British Machine Vision Conference*.

Zhang, C.; Bengio, S.; Hardt, M.; Recht, B.; and Vinyals, O. 2017a. Understanding deep learning requires rethinking generalization. In *ICLR*.

Zhang, H.; Cisse, M.; Dauphin, Y. N.; and Lopez-Paz, D. 2017b. mixup: Beyond empirical risk minimization. In *ICLR*.

Zhang, Y.; Xiang, T.; Hospedales, T. M.; and Lu, H. 2018. Deep mutual learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4320–4328.