ACHIEVING STRONG REGULARIZATION FOR DEEP NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

L1 and L2 regularizers are critical tools in machine learning due to their ability to simplify solutions. However, imposing strong L1 or L2 regularization with gradient descent method easily fails, and this limits the generalization ability of the underlying neural networks. To understand this phenomenon, we investigate how and why training fails for strong regularization. Specifically, we examine how gradients change over time for different regularization strengths and provide an analysis why the gradients diminish so fast. We find that there exists a tolerance level of regularization strength, where the training completely fails if the regularization strength goes beyond it. We propose a time-dependent regularization schedule in order to moderate the tolerance level. Experiments show that our proposed approach indeed achieves strong regularization for both L1 and L2 regularizers and improve both accuracy and sparsity on public data sets. Our source code is published.[1]

1 INTRODUCTION

Regularization has been very common for machine learning to prevent over-fitting and to obtain sparse solutions. Deep neural networks (DNNs), which have shown huge success in many tasks such as computer vision (Krizhevsky et al., 2012; Simonyan & Zisserman, 2014; He et al., 2016) and speech recognition (Hinton et al., 2012), often contain a number of parameters in multiple layers with non-linear activation functions, in order to gain enough expressive power. However, DNNs with many parameters are often prone to over-fitting. Therefore, the need for regularization has been emphasized. While new regularization techniques such as dropout (Srivastava et al., 2014) and pruning (Han et al., 2015) have been proposed to solve the problem, the traditional regularization techniques using L1 or L2 norms have cooperated with them to further improve the performance significantly. L1 regularization, often called Lasso (Tibshirani, 1996), obtains sparse solutions so that the required memory and power consumption are reduced while keeping good accuracy. On the other hand, L2 regularization smoothes the parameter distribution and reduces the magnitude of parameters, so the resulting solution is simple (*i.e.,* less prone to over-fitting) and effective. Indeed, our empirical results show that adding strong L2 regularization to the deep neural networks that already has dropout layers can reduce the error rate by up to 24%.

Strong regularization is especially desired when the model contains too many parameters for the given amount of training data. This is often the case for deep learning tasks in practice because DNNs often contain millions of parameters while labeled training data set is limited and expensive. However, imposing strong L1 or L2 regularization on DNNs is difficult for gradient descent method. If we impose too strong regularization, the gradient from regularization becomes dominant, and DNNs stop learning. In this paper, we first study this interesting phenomenon of strong regularization failure. We find that there exists a tolerance level of regularization strength, where the training completely fails if the regularization strength goes beyond it. We also provide an analysis why the gradients diminish so quickly that learning completely fails. Then, we propose a simple yet effective solution, which carries time-dependent schedule of regularization strength. The proposed approach is general and does not require any additional computation. We perform experiments with popular neural network architectures on public data sets. The experiment results indicate that the proposed approach indeed achieve strong regularization, consistently yielding even higher accuracy

[1] `https://github.com/(anonymized)`

and higher sparsity that could not be achieved. In addition, as it can serve a wide range of regularization strength, the proposed approach is relatively easy to tune.

2 PROBLEM ANALYSIS

2.1 BACKGROUND

Let us denote a generic DNN by $\mathbf{y} = f(\mathbf{x}; \mathbf{w})$ where $\mathbf{x} \in \mathbb{R}^d$ is an input vector, $\mathbf{w} \in \mathbb{R}^n$ is a flattened vector of all parameters in the network f, and $\mathbf{y} \in \mathbb{R}^c$ is an output vector after feed-forwarding $\mathbf{x}$ through multiple layers in f. The network f is trained by finding optimal set of $\mathbf{w}$ by minimizing the cost function within the training data $\{\mathbf{x}_i, \mathbf{d}_i\}_{i=1}^m$ as follows.

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(f(\mathbf{x}_i; \mathbf{w}), \mathbf{d}_i) + \lambda \Omega(\mathbf{w}) \tag{1}$$

where $\mathcal{L}$ is the loss function, which is usually cross-entropy loss for classification tasks. Here, the regularization term $\lambda\Omega(\mathbf{w})$ is added to simplify the solution, and λ is set to zero for non-regularized cost function. A higher value of λ means that stronger regularization is imposed.

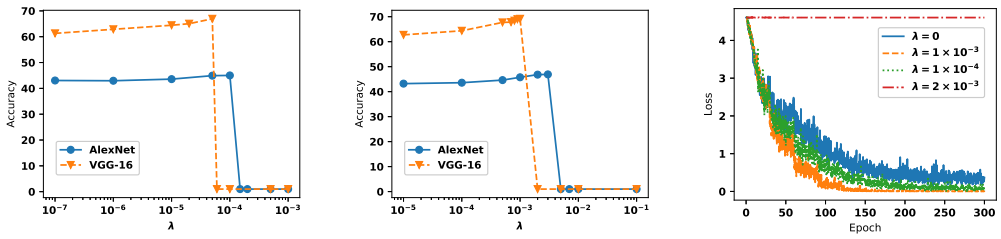
The most commonly used regularization function is a squared L2 norm: $\Omega(\mathbf{w}) = ||\mathbf{w}||_2^2$, which is also called as *weight decay* in deep learning literature. This L2 regularizer has an effect of reducing the magnitude of the parameters $\mathbf{w}$, and the simpler solution becomes less prone to over-fitting. On the other hand, the L1 regularizer: $\Omega(\mathbf{w}) = ||\mathbf{w}||_1$ is often employed to induce sparsity in the model (*i.e.,* makes a portion of $\mathbf{w}$ zero). The sparse solution is often preferred to reduce computation time and memory consumption for deep learning since DNNs often require heavy computation and big memory space.

With the gradient descent method, each model parameter at time t, $\mathbf{w}_i^{(t)}$, is updated with the following formula:

$$\begin{aligned} \mathbf{w}_i^{(t+1)} &= \mathbf{w}_i^{(t)} - \alpha \left(\frac{\partial \mathcal{L}}{\partial \mathbf{w}_i} + \lambda \frac{\partial \Omega}{\partial \mathbf{w}_i} \right) \Bigg|_{\mathbf{w}_i = \mathbf{w}_i^{(t)}} \\ \frac{\partial \Omega}{\partial \mathbf{w}_i} &= \begin{cases} 2 \times \mathbf{w}_i^{(t)}, & \text{if } \Omega(\mathbf{w}) = ||\mathbf{w}||_2^2 \\ 1 \times \text{sign}(\mathbf{w}_i^{(t)}), & \text{if } \Omega(\mathbf{w}) = ||\mathbf{w}||_1 \text{ and } \mathbf{w}_i^{(t)} \neq 0 \end{cases} \end{aligned} \tag{2}$$

where α is a learning rate. As L1 norm is not differentiable at 0, the formula doesn't have value when $\mathbf{w}_i^{(t)} = 0$, but in practice, the subgradient 0 is often used. Please see Section 2.3 for more details. From the formula, we can see that L2 regularizer continuously reduces the magnitude of a parameter proportionally to it while L1 regularizer reduces the magnitude by a constant. In both regularizers, strong regularization thus means greatly reducing the magnitude of parameters.

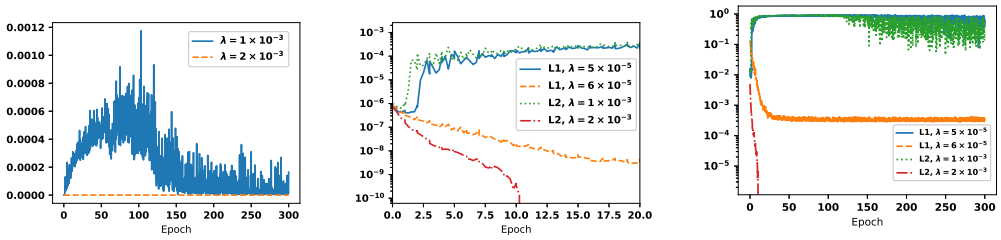
2.2 IMPOSING STRONG REGULARIZATION MAKES LEARNING FAIL.



(a) L1 regularization (b) L2 regularization (c) Training loss

Figure 1: (a,b) Validation accuracies for different λ when L1 and L2 regularization is applied to VGG-16 and AlexNet on the CIFAR-100 data set. Note the sharp accuracy drop. (c) Training loss when different λ for L2 regularization is used for VGG-16 on CIFAR-100. Note that the regularization loss is excluded from the training loss.

Strong regularization is especially useful for deep learning because the DNNs often contain a large number of parameters while the training data is limited in practice. However, we have observed a phenomenon where learning suddenly fails when strong regularization is imposed for gradient descent method, which is the most commonly used solver for deep learning. The example of the phenomenon is depicted in Figure 1. In the example, the architectures VGG-16 (Simonyan & Zisserman, 2014) and AlexNet (Krizhevsky et al., 2012) were employed for the data set CIFAR-100 (Krizhevsky & Hinton, 2009).[2] As shown, the accuracy increases as we enforce more regularization. However, it suddenly drops to 1.0% after enforcing a little more regularization, which means that the model failed learning. The depicted training loss also indicates that it indeed learns faster with a stronger regularization, but the training loss does not improve at all when a little more regularization is imposed.



(a) Average $|\frac{\partial \mathcal{L}}{\partial \mathbf{w}_i}|$ (b) Average $|\frac{\partial \mathcal{L}}{\partial \mathbf{w}_i}|$ (close-up) (c) $\frac{\sum_i |\frac{\partial \mathcal{L}}{\partial \mathbf{w}_i}|}{\sum_i |\frac{\partial \mathcal{L}}{\partial \mathbf{w}_i}| + \lambda |\frac{\partial \Omega}{\partial \mathbf{w}_i}|}$

Figure 2: Gradients for different λ by VGG-16 on CIFAR-100. (a) Average amount of gradient from $\mathcal{L}$ when L2 regularization is applied. (b) A close-up version of (a) with y-axis in log-scale. (c) The proportion of gradients from $\mathcal{L}$ to all gradients. Best shown in color.

In order to look at this phenomenon in more detail, we plot how gradients and their proportion change in Figure 2. As depicted in Figure 2a, a model with moderate L2 regularization ($\lambda = 1 \times 10^{-3}$) follows a path that has a steeper slope during the first 100 epoch, and then it converges with a gentle slope. However, a model with a little more L2 regularization ($\lambda = 2 \times 10^{-3}$) does not follow a path that has a good slope, and it doesn't have a chance to learn from gradients. A close-up view of this in first 20 epochs is depicted in Figure 2b. The models with moderate L1 and L2 regularization seem to follow a good path in a couple of epochs. Through following the good path, the models keep the proportion of the gradients from $\mathcal{L}$ to all gradients dominant, especially for the first 150 epochs (Figure 2c). On the other hand, the models with a little stronger regularization fail to follow such path and the gradients from $\mathcal{L}$ decreases exponentially (Figure 2b). Since the magnitude of gradients from $\mathcal{L}$ decreases faster than that from Ω, the proportion of the latter to all gradients becomes dominant (Figure 2c), and it results in failure in learning. From this observation, we can learn that there exists a *tolerance level* of regularization strength, which affects success or failure of entire learning.

Why does the magnitude of the gradient from $\mathcal{L}$ decrease so fast? It is not difficult to see why the magnitude of $\frac{\partial \mathcal{L}}{\partial \mathbf{w}_i}$ decreases so fast when the regularization is strong. In deep neural networks, the gradients are dictated by back-propagation. It is well known that the gradients at the lth layer are given by

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}^{(l)}} = \boldsymbol{\delta}^{(l)} \left(\mathbf{a}^{(l-1)} \right)^T \tag{3}$$

where $\mathbf{a}^{(l-1)}$ is the output of the neurons at the $(l-1)^{\text{th}}$ layer and $\boldsymbol{\delta}^{(l)}$ is the lth-layer residual which follows the recursive relation

$$\boldsymbol{\delta}^{(l)} = \left(\mathbf{w}^{(l+1)} \right)^T \boldsymbol{\delta}^{(l+1)} \odot \mathbf{a}'^{(l)} \tag{4}$$

where $\odot$ and $\mathbf{a}'$ denote the element-wise multiplications and derivatives of the activation function respectively.

[2]Details of the experiment settings are described in Section 3

Using the recursive relation, we obtain

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}^{(l)}} = (\mathbf{w}^{(l+1)})^T (\mathbf{w}^{(l+2)})^T \cdots (\mathbf{w}^{(L)})^T \boldsymbol{\delta}^{(L)} \odot \mathbf{a}'^{(L-1)} \odot \mathbf{a}'^{(L-2)} \odot \cdots \odot \mathbf{a}'^{(l+1)} \odot \mathbf{a}'^{(l)} (\mathbf{a}^{(l-1)})^T \tag{5}$$

If the regularization is too strong, the weights would be significantly suppressed. From (5), since the gradients are proportional to the product of the weights at later layers (whose magnitudes are typically much less than 1 for strong regularization), they are even more suppressed.

In fact, the suppression is more severe than what we have deduced above. The factor $\mathbf{a}^{(l-1)}$ in (5) could actually lead to further suppression to the gradients when the weights are very small, for the following reasons. First of all, we use ReLU as the activation function and it could be written as

$$\text{ReLU}(x) = x\ \Theta(x) \tag{6}$$

where $\Theta(x)$ is the Heaviside step function. Using this, we could write

$$\mathbf{a}^{(l-1)} = \left(\mathbf{w}^{(l-1)}\,\mathbf{a}^{(l-2)}\right) \odot \Theta\left(\mathbf{w}^{(l-1)}\,\mathbf{a}^{(l-2)}\right) \tag{7}$$

Applying (7) recursively, we can see that $\mathbf{a}^{(l-1)}$ is proportional to the product of the weights at previous layers. Again, when the weights are suppressed by strong regularization, $\mathbf{a}^{(l-1)}$ would be suppressed correspondingly. Putting everything together, we can conclude that in the presence of strong regularization, the gradients are far more suppressed than the weights.

Strictly speaking, the derivations above are valid only for fully-connected layers. For convolutional layers, the derivations are more complicated but similar. Our conclusions above would still be valid.

Normalization Normalization techniques such as batch normalization (Ioffe & Szegedy, 2015) and weight normalization (Salimans & Kingma, 2016) can be possible approaches to prevent the $\mathcal{L}$ gradients from diminishing quickly. However, it has been shown that L2 regularization has no regularizing effect when combined with normalization but only influences on the effective learning rate, resulting in good performance (van Laarhoven, 2017). In other words, the normalization techniques do not really simplify the solution as the decrease of parameter magnitude is canceled by normalization. This does not meet our goal, which is to heavily simplify solutions to reduce over-fitting, so we propose approaches that meet our goal.

2.3 Approach to Accommodate Strong Regularization

Since we have seen that stronger regularization can result in better performance in Figure 1, we propose a method that is able to accommodate strong regularization. Specifically, we introduce a time-dependent regularization strength, λ_t, to the equation (2), and it is defined as

$$\lambda_t = \begin{cases} 0 & \text{if epoch}(t) < \gamma \\ \lambda & \text{otherwise} \end{cases} \tag{8}$$

where epoch(t) gets the epoch number of the time step t, and γ is a hyper-parameter that is set through cross-validation. The formula means that we do not impose any regularization until γ^{th} epoch, and then impose the full regularization. The underlying hypothesis is that once the model follows a good learning path, *i.e.*, the gradient from $\mathcal{L}$ is big enough, it won't easily change its direction because of the steep slope, and thus, it can learn without failure. We empirically verify the hypothesis in the experiment section. The hyper-parameter γ is relatively easy to set because the models often follow the good path in a couple of epochs. We recommend using $5 \leq \gamma \leq 10$.

Proximal gradient algorithm for L1 regularizer Meanwhile, since L1 norm is not differentiable at zero, we employ the proximal gradient algorithm (Parikh et al., 2014), which enables us to obtain proper sparsity (*i.e.,* guaranted convergence) for non-smooth regularizers. We used the following

update formulae:

$$
\begin{aligned}
\mathbf{w}_i^{(t')} &= \mathbf{w}_i^{(t)} - \alpha \left(\frac{\partial \mathcal{L}}{\partial \mathbf{w}_i} + \lambda \frac{\partial \Omega}{\partial \mathbf{w}_i} \right) \Bigg|_{\mathbf{w}_i = \mathbf{w}_i^{(t)}} \\
\mathbf{w}_i^{(t+1)} &= \text{prox}_{\lambda\Omega}(\mathbf{w}_i^{(t')}) = S(\mathbf{w}_i^{(t')}, \alpha\lambda) \\
S(a, z) &= \begin{cases} a - z & \text{if } a > z \\ a + z & \text{if } a < -z \\ 0 & \text{otherwise} \end{cases}
\end{aligned}
\tag{9}
$$

where S is a soft-thresholding operator that assigns zero to a parameter if its next updated value is smaller than $\alpha\lambda$. In other cases, it just decreases the magnitude of the parameter as usual. TODO: vs pruning.

3 Experiments

We evaluate the effectiveness of our proposed method with popular architectures, AlexNet Krizhevsky et al. (2012) and VGG-16 (Simonyan & Zisserman, 2014) on the public data sets CIFAR-10 and CIFAR-100 (Krizhevsky & Hinton, 2009). Please note that we do not employ architectures that contain normalization techniques such as batch normalization (Ioffe & Szegedy, 2015), for the reason described in Section 2.2. The data set statistics are described in Table 1. Regularization is applied to all network parameters except the bias terms. We use PyTorch[3] framework for all experiments, and we use its official vision library[4] for the implementations of the networks. In order to accommodate CIFAR-10 and CIFAR-100 datasets, we made some modifications to the networks. The kernel size of AlexNet's max-pooling layers is changed from 3 to 2, and the first convolution layer's padding size is changed from 2 to 5. All of its fully connected layers are modified to have 256 neurons. For VGG-16, we modified the fully connected layers to have 512 neurons. The output layers of both networks have 10 and 100 neurons for CIFAR-10 and CIFAR-100, respectively. The dropout layers have the default drop probability of 0.5. The networks are learned by stochastic gradient descent algorithm with momentum of 0.9. The parameters are initialized according to He et al. (2015). The batch size is set to 128, and the initial learning rate is set to 0.05 and decays by a factor of 2 every 30 epochs during the whole 300-epoch training.

Table 1: Data set statistics for CIFAR-10 and CIFAR-100.

Data Set	Image Resolution	Classes	Training Images per Class	Test Images per Class	AlexNet Parameters	VGG-16 Parameters
CIFAR-10	32×32	10	5000	1000	2.6M	15.2M
CIFAR-100	32×32	100	500	100	2.6M	15.3M

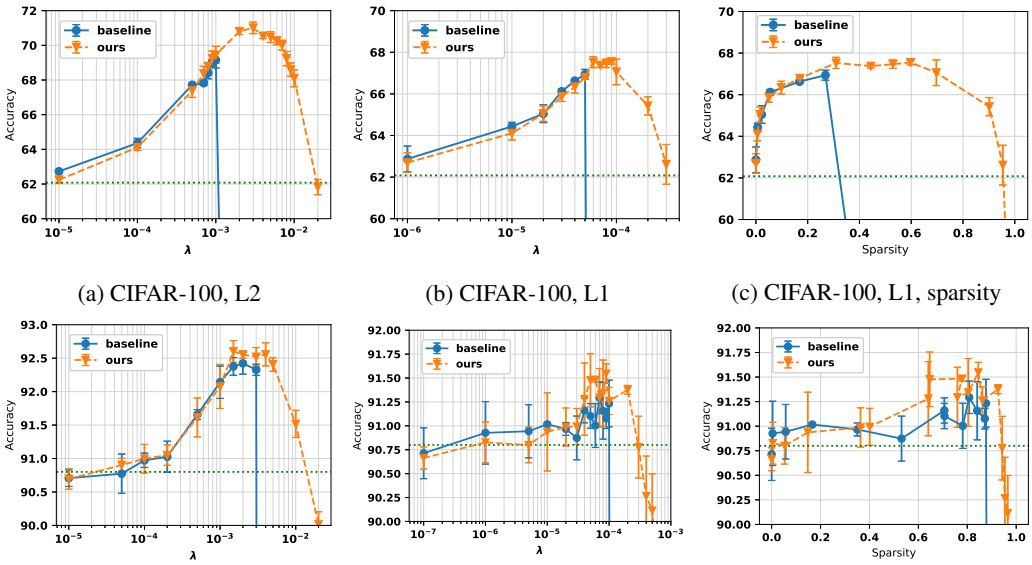
Both architectures are experimented for different regularization methods (L1 and L2) and different data sets (CIFAR-10 and CIFAR-100), yielding 8 combinations of experiment sets. For each experiment set, we set the **baseline** method as the one with well-tuned L1 or L2 regularization but without our time-dependent λ. For each regularization, we try more than 10 different values of λ, and for each value, we report average accuracy of three independent runs and plot 95% confidence interval with error bars. We also report **sparsity** of each trained model, which is the proportion of the number of zero-valued parameters to the number of all parameters. Please note that we mean the sparsity by the one actually derived by the models, not by pruning parameters with threshold after training.

3.1 Experiment Results

The experiment results by VGG-16 are depicted in Figure 3. As we investigated in Section 2.2, the baseline method suddenly fails beyond certain values of λ. However, our proposed method can endure the early suppression from strong regularization, so it does not fail for higher values of λ.

[3] http://pytorch.org/

[4] https://github.com/pytorch/vision



(a) CIFAR-100, L2 (b) CIFAR-100, L1 (c) CIFAR-100, L1, sparsity

(d) CIFAR-10, L2 (e) CIFAR-10, L1 (f) CIFAR-10, L1, sparsity

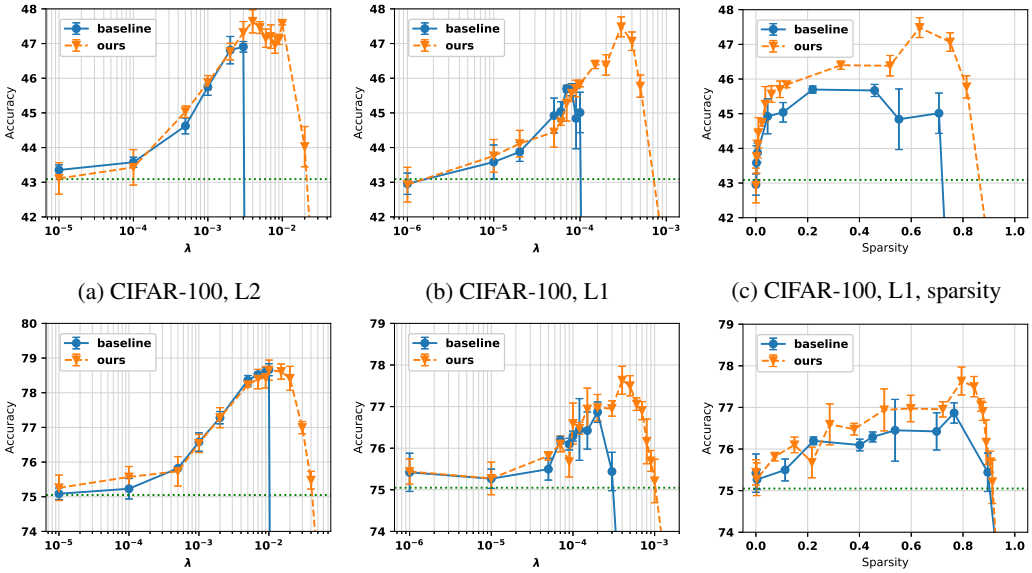
Figure 3: Accuracies obtained by VGG-16 with L1 and L2 regularization on CIFAR-100 (a,b,c) and CIFAR-10 (d,e,f). A green dotted horizontal line is an accuracy obtained by a model without L1/L2 regularization (but with dropout). Accuracy for different sparsity is shown in (c) and (f).

As a result, our model can achieve higher accuracy as well as higher sparsity. L2 regularization is used more often than L1 regularization due to its superior performance, and this is true for our VGG-16 experiments too. Using L2 regularization, our model improves the model without L1 or L2 regularization but with dropout, by 8.9 percent point (pp), which is about 24% of error rate improvement. This shows that L2 regularization is very important for large networks. Tuning L2 parameter is difficult as the curves have somewhat sharp peak, but our proposed method ease the problem to some extent by preventing the sharp drop. Our L1 regularizer obtains a much better sparsity (Figure 3c), which means that strong regularization is important to compress neural networks.

The improvement is more prominent in CIFAR-100 than in CIFAR-10, and we think this is because over-fitting can more likely occur in CIFAR-100 as there are less images per class than in CIFAR-10. Nonetheless, our proposed method often obtains higher accuracies even when the baseline does not fail in CIFAR-10, and this is only prominent when the regularization strength is relatively strong (better shown in Figure 3c, 3f, 4c, 4f). We think that this is because avoiding strong regularization in early stage of training can help the model to explore more spaces freely, and the better exploration results in finding superior local optima.

The experiment results by AlexNet are depicted in Figure 4. Again, our proposed method achieves higher accuracy and sparsity in general. Unlike VGG-16, we obtain more improvement over baseline with L1 regularization than with L2 regularization. In addition, the curves make sharper peaks than those by VGG-16 do especially for the sparsity regularizer (L2). This may be because AlexNet has a smaller number of parameters than VGG-16 has, so it is more sensitive to regularization strength.

The overall experiment results are shown in Table 2. It shows that there is more L1/L2 regularization effect on VGG-16 than on AlexNet, which is reasonable since VGG-16 contains about 6 times more parameters so that is is more prone to over-fitting. L1 regularizers show the best accuracy with higher sparsity on CIFAR-10 data set. This is probably because CIFAR-10 does not require as many parameters as CIFAR-100 requires, because the number of classes to predict is much less in CIFAR-10, so a higher model capacity is not needed. Our proposed model improves the baselines by up to 1.85 percent point, except AlexNet with L1 on CIFAR-10. Our L1 models always obtain higher sparsity with compression rate up to 4.2$\times$ than baselines, meaning that our model is promising for compressing neural networks.



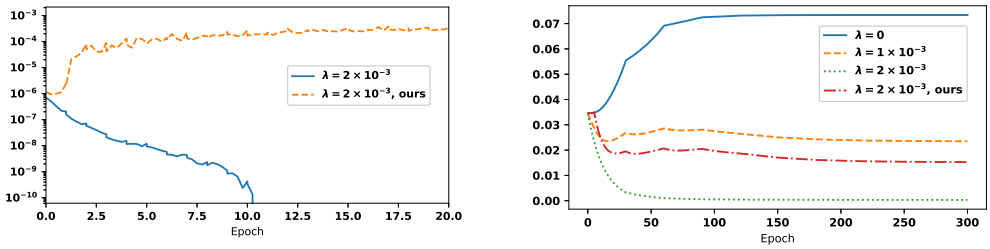
(a) CIFAR-100, L2 (b) CIFAR-100, L1 (c) CIFAR-100, L1, sparsity

(d) CIFAR-10, L2 (e) CIFAR-10, L1 (f) CIFAR-10, L1, sparsity

Figure 4: Accuracies obtained by AlexNet with L1 and L2 regularization on CIFAR-100 (a,b,c) and CIFAR-10 (d,e,f). A green dotted horizontal line is an accuracy obtained by a model without L1/L2 regularization (but with dropout). Accuracy for different sparsity is shown in (c) and (f).

Table 2: Overall experiment results. The 95% confidence interval is represented after $\pm$ symbol. The improvement is calculated by comparing with baseline methods. Compression rate is calculated by how much the regularization reduces the number of non-zero-valued parameters while having an accuracy similar to L1 baseline. Note that when no L1/L2 regularization is imposed, dropout is still employed.

Model		CIFAR-100		CIFAR-10	
		Sparsity	Acc. (%)	Sparsity	Acc. (%)
VGG-16	No L1/L2	0	62.08 ± 0.81	0	90.80 ± 0.23
	L2 baseline	0	69.16 ± 0.46	0	92.42 ± 0.16
	L2 ours	0	71.01 ± 0.33	0	92.60 ± 0.16
	Abs. acc. improvement	-	+1.85 pp	-	+0.18 pp
	L1 baseline	0.269	66.94 ± 0.24	0.808	91.29 ± 0.16
	L1 ours	0.303	67.55 ± 0.12	0.845	91.55 ± 0.10
	Abs. acc. improvement	-	+0.61 pp	-	+0.26 pp
	L1 ours (sparse)	0.697	67.06 ± 0.62	0.926	91.38 ± 0.05
	Compression rate over L1 baseline	2.4×	+0.12 pp	2.6×	+0.09 pp
AlexNet	No L1/L2	0	43.09 ± 0.25	0	75.05 ± 0.20
	L2 baseline	0	46.91 ± 0.15	0	78.66 ± 0.17
	L2 ours	0	47.64 ± 0.33	0	78.65 ± 0.29
	Abs. acc. improvement	-	+0.73 pp	-	−0.01 pp
	L1 baseline	0.219	45.70 ± 0.10	0.766	76.87 ± 0.24
	L1 ours	0.632	47.48 ± 0.29	0.794	77.63 ± 0.34
	Abs. acc. improvement	-	+1.78 pp	-	+0.76 pp
	L1 ours (sparse)	0.814	45.77 ± 0.32	0.877	76.90 ± 0.22
	Compression rate over L1 baseline	4.2×	+0.07 pp	1.9×	+0.03 pp



(a) Average $|\frac{\partial \mathcal{L}}{\partial \mathbf{w}_i}|$ (close-up)

(b) Average Magnitude of Parameters

Figure 5: Experiment results by L2 regularization with VGG-16 on CIFAR-100. The results are from baseline method unless it is labeled as "ours".

We also show how gradients and weights change when our method and baselines are applied in Figure 5. As we hypothesized, our time-dependent λ schedule helps the model keep learning whereas the baselines fail to learn. When strong regularization is imposed with our method, the weights do not decrease for the first five epochs as supposed, and then the weights decrease. However, unlike the baseline methods, the weights stop decreasing after some time, resulting in successful learning.

4 Related Work

The related work is partially covered in Section 1, and we extend other related work here. It has been shown that L2 regularization is important for training DNNs (Krizhevsky et al., 2012; Deng et al., 2013). Although there has been a new regularization method such as dropout, and L2 regularization was shown to reduce the test error effectively when combined with dropout (Srivastava et al., 2014). Meanwhile, L1 regularization has also been used often in order to obtain sparse solutions. To reduce computation and power consumption, L1 regularization and its variations such as group sparsity regularization can be promising for deep neural networks (Wen et al., 2016; Scardapane et al., 2017; Yoon & Hwang, 2017). However, for both L1 and L2 regularization, the phenomenon that strong regularization fails has not been emphasized previously. Bergstra & Bengio (2012) showed that tuning hyper-parameters such as l2 regularization strength can be effectively done through random search instead of grid search, but it is not studied how strong regularization fails or can be achieved. Yosinski et al. (2015) visualized activations to understand deep neural networks and showed that strong L2 regularization fails to learn. However, it is still not shown how exactly strong regularization fails and how it can be achieved. To the best of our knowledge, there is no existing work that is dedicated to studying the phenomenon that strong regularization fails and to propose a method that can avoid the failure.

5 Discussion

In this work, we studied the problem of achieving strong regularization for deep neural networks. Strong regularization with gradient descent algorithm easily fails for deep neural networks, but few work addressed this phenomenon in detail. We provided investigation and analysis of the phenomenon, and we found that there is a strict tolerance level of regularization strength. To avoid this problem, we proposed a novel but simple method: a time-dependent regularization schedule. Our model does not require extra computations, and it is relatively easy to tune. We performed experiments with fine tuning of regularization strength. Evaluation results show that (1) our model indeed achieves strong regularization on deep neural networks, verifying our hypothesis that the model will keep learning once it follows a path with steep slope, (2) with strong regularization, our model obtains higher accuracy and sparsity, (3) even when the baseline regularizer does not fail, our model still outperforms it with a better early exploration, and (4) L1/L2 regularization is difficult to tune, and it may be more difficult for smaller networks, but it can yield great performance boost when tuned well.

There are limitations in this work. Our proposed method can be especially useful when strong regularization is desired. For example, deep learning projects that cannot afford a huge labeled data

set can be benefit from our method. However, strong regularization may not be necessary in other cases where the labeled data set is available or the networks do not contain many parameters. Our experiments were not performed on a bigger data set such as ImageNet data set, which is another limitation of this work. Since we needed to investigate the problem in fine detail, we had to try many different regularization strength with different models, which may take too much time for ImageNet.

Our work can be further extended in several ways. Since our model can achieve strong regularization, it will be interesting to see how the strongly regularized parameters perform if combined with pruning-related methods (Han et al., 2015; 2016). We applied our approach to only L1 and L2 regularizers, but applying it to other regularizers such as group sparsity regularizers will be promising. Lastly, our proposed method with a regularization strength schedule is very simple, so it can be easily extended to more complicated schedules. All these directions are left as our future work.

REFERENCES

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

Li Deng, Geoffrey Hinton, and Brian Kingsbury. New types of deep neural network learning for speech recognition and related applications: An overview. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pp. 8599–8603. IEEE, 2013.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems*, pp. 1135–1143, 2015.

Song Han, Jeff Pool, Sharan Narang, Huizi Mao, Shijian Tang, Erich Elsen, Bryan Catanzaro, John Tran, and William J Dally. Dsd: Regularizing deep neural networks with dense-sparse-dense training flow. *arXiv preprint arXiv:1607.04381*, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

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.

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

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.

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

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Neal Parikh, Stephen Boyd, et al. Proximal algorithms. *Foundations and Trends® in Optimization*, 1(3):127–239, 2014.

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.

Simone Scardapane, Danilo Comminiello, Amir Hussain, and Aurelio Uncini. Group sparse regularization for deep neural networks. *Neurocomputing*, 241:81–89, 2017.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

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.

Twan van Laarhoven. L2 regularization versus batch and weight normalization. *arXiv preprint arXiv:1706.05350*, 2017.

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.

Jaehong Yoon and Sung Ju Hwang. Combined group and exclusive sparsity for deep neural networks. In *International Conference on Machine Learning*, pp. 3958–3966, 2017.

Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.