Schedae Informaticae Vol. 27 (2018): 31–45
doi: 10.4467/20838476SI.18.003.10408

Gradient Regularization Improves Accuracy of Discriminative Models

Dániel Varga[1], Adrián Csiszárik[1,2], Zsolt Zombori[1,2]
1: Alfréd Rényi Institute of Mathematics, Hungarian Academy of Sciences
2: ELTE, Institute of Mathematics, Department of Computer Science
Budapest, Hungary
{daniel,csadrian,zombori}@renyi.hu

Abstract. Regularizing the gradient norm of the output of a neural network is a powerful technique, rediscovered several times. This paper presents evidence that gradient regularization can consistently improve classification accuracy on vision tasks, using modern deep neural networks, especially when the amount of training data is small. We introduce our regularizers as members of a broader class of Jacobian-based regularizers. We demonstrate empirically on real and synthetic data that the learning process leads to gradients controlled beyond the training points, and results in solutions that generalize well.

Keywords: neural network, generalization, gradient regularization, spectral norm, Frobenius norm

1. Introduction and Related Work

Regularizing the gradient norm of a neural network's output with respect to its inputs goes back to *Double Backpropagation* [2], with variants independently rediscovered several times [16, 11, 1, 4], most recently by the authors of this paper. Outside the domain of neural networks, *Sobolev regularization* [13] is essentially the same concept, a special case of the very general, classic method of approximating functions in Sobolev space [5]. Smoothing splines [19] are another important special case.

Most recent applications [3, 16, 11, 15] focus on robustness against adversarial sampling [18]. Here we argue that gradient regularization can be used for the more fundamental task of increasing classification accuracy, especially when the available

training set is small. Our work explores a broad class of of Jacobian-based regularizers, providing a unified framework for various gradient regularization approaches. After introducing the general framework, we focus on the two most promising variants: 1) classic *Double Backpropagation* [2], and 2) *SpectReg* which is our contribution.

Our experiments control gradients at labeled training points, but we note that SpectReg makes no use of labels and can be applied in a semi-supervised setting.

2. Analysis

We consider feed-forward classifier networks with a loss function $L(x, y, \Theta) = M(f(x, \Theta), y) = M(\text{softmax}(g(x, \Theta)), y)$, with input x, one-hot encoded desired output y, neural network f with a top softmax layer, network parameters Θ and categorical cross entropy function M. The inputs and outputs of the softmax layer are called *logits* and *probabilities*, respectively. The central object of our investigation is the Jacobian of the logits $J_g(x) = \frac{\partial}{\partial x} g(x)$ with respect to the inputs.

2.1. Gradient Regularizaton Schemes

Gradient regularization penalizes large changes in the output of some neural network layer, to enforce a smoothness prior. We get different variants depending on where the gradients are computed (logits, probabilities, loss term), with respect to what (inputs, hidden activations), what loss function is used to create a scalar loss. Some of these variants require the computation of a full Jacobian matrix, which can be made more efficient by the application of some projection, introducing new variants. Here we present the most promising variants:

- **Double Backpropagation (DoubleBack)** [2, 11] Penalize the squared $L2$ norm of the gradient of the original loss term with respect to the inputs.

  $$L_{DG}(x, y, \Theta) = L(x, y, \Theta) + \lambda \| (\frac{\partial}{\partial x} L(x, y, \Theta)) \|_2^2$$

  Although not obvious from its definition, DoubleBack can be interpreted as applying a particular projection to the Jacobian of the logits and regularizing it. We prove this in Subsection 2.3..

- **Jacobian Regularizer (JacReg)** [16] Penalize the squared Frobenius norm of the Jacobian of the softmax output (probabilities) with respect to the input.

  $$L_{JacReg}(x, y, \Theta) = L(x, y, \Theta) + \lambda \| J_f \|_F^2$$

  Symbolically computing the Jacobian is expensive as computation scales linearly with the number of output labels. To alleviate this, a layer-wise approximation is employed in [3] and [16].

- **Frobenius Regularizer (FrobReg)** (Our contribution, also introduced by [8].) Penalize the squared Frobenius norm of the Jacobian of the logits with respect to the input.

$$L_{FrobReg}(x, y, \Theta) = L(x, y, \Theta) + \lambda \|J_g\|_F^2$$

  FrobReg only differs from JacReg in that the Jacobian is computed on the logits instead of the probabilities. Computation is equally expensive, however, diminishing gradients due to the softmax transformation are less of a concern.

- **Spectral Regularization (SpectReg)** (Our contribution.) Apply a random projection to the Jacobian of the logits, and penalize its squared $L2$ norm:

$$L_{SpectReg}(x, y, \Theta) = L(x, y, \Theta) + \lambda \|P_{rnd}(J_g)\|_2^2$$

  where $P_{rnd}(J_g) = J_g^T r$ and $r \in \mathcal{N}(0, I_m)$.

  If the random projector of SpectReg is normalized onto the unit sphere (*spherical SpectReg*), the norm of the projection is a lower bound to the (hard to compute) spectral norm of the Jacobian, which motivates our naming. Furthermore, one can easily show that spherical SpectReg is an unbiased estimator of the squared Frobenius norm of the Jacobian, and so is SpectReg, up to constant scaling. We have not observed any empirical differences between the spherical and the unnormalized variants. The Frobenius norm is within a constant factor of the spectral norm, so it can be interpreted as a proxy when our goal is to enforce a Lipschitz property.

2.2. Starting from the linear case

It is instructive to first consider the toy edge case when the neural network consists of a single dense linear layer. Here the weight matrix and the Jacobian coincide. Thus, for this network, FrobReg is identical to L2 weight decay and SpectReg is an estimator for weight decay. The Frobenius norm is submultiplicative, and the gradient of the ReLU is upper bounded by 1. Thus, for a dense ReLU network the product of layer-wise weight norms is an upper bound for the FrobReg loss term. Applying the inequality of arithmetic and geometric means, we can see that the total weight norm can be used to upper bound the FrobReg loss term. Penalizing the Frobenius norm of the Jacobian seems to be closely connected to weight decay, however, the former is more targeted to the data distribution. The Frobenius norm also serves as a proxy for the spectral norm, as they are equivalent matrix norms. For multi-valued functions, calculating, or even approximating the spectral norm is infeasible. Hence, Frobenius norm regularization is a reasonable approach to enforce a Lipschitz-like property.

2.3. Choosing where the gradient is computed

Penalizing gradients computed at different vectors (logits, probabilities or loss) leads to different regularizing effects, worth comparing theoretically.

1. **logits vs. probabilities** The Jacobian matrix of the softmax transformation is $J_{softmax} = (p\mathbf{1}^T) \odot (I - p\mathbf{1}^T)^T$, where p is the probability vector and $\mathbf{1}$ denotes the all-one vector. Consequently:

$$J_f = J_{softmax} J_g = \left[(p\mathbf{1}^T) \odot (I - p\mathbf{1}^T)^T\right] J_g \tag{1}$$

   One can easily verify that as p tends to a one-hot vector, $J_{softmax}$ tends to the zero matrix. This results in a vanishing effect: once the network starts to converge, JacReg disappears, while FrobReg and SpectReg does not.

   JacReg does not punish big changes in the logits when they do not affect class probabilities. This can lead to less robust predictions. Indeed, as our experiments confirm, FrobReg and SpectReg consistently outperform JacReg.

2. **probabilities vs. loss** Since the Jacobian of cross-entropy is $J_{xent} = -\left[\frac{y}{p}\right]^T$:

$$J_{loss} = J_{xent} J_f = \left[-\frac{y}{p}\right]^T J_f \tag{2}$$

   Assuming that y is a one-hot vector, the gradient at the loss is obtained from the Jacobian at the probabilities by selecting row t of J_f corresponding to the true label and multiplying it with $-\frac{1}{p_t}$. Most importantly, we no longer optimize the full Jacobian J_f, only a single row.

3. **logits vs. loss** Composing the softmax and cross-entropy Jacobians, we obtain:

$$J_{loss} = J_{xent} J_{softmax} J_g = \left[-\frac{y}{p}\right]^T \left[(p\mathbf{1}^T) \odot (I - p\mathbf{1}^T)^T\right] J_g \tag{3}$$

$$J_{loss} = \left[-\frac{1}{p_t}\right] \left[(p_t\mathbf{1}^T) \odot (y^T - p^T)\right] J_g \tag{4}$$

$$J_{loss} = (p - y)^T J_g \tag{5}$$

   Equation 5 shows that calculating the gradient at the loss corresponds to applying a projection to the Jacobian of the logits, revealing a close connection between DoubleBack and SpectReg. DoubleBack applies a projection in a particular direction, while SpectReg selects a uniformly random direction, hence aims to control the gradients in all directions.

   As our analysis shows, the gradient norms computed after the softmax layer quickly vanish during training, which indicates that the weights of such regularizers (JacReg, DoubleBack) require careful tuning. Indeed, our experiments confirm this hypothesis. Regularization based on the logits (FrobReg, SpectReg) is significantly more robust to the choice of hyperparameters.

2.4. Computationally efficient regularization of the Jacobian

Some of the variants presented in Subsection 2.1. compute gradients on a vector (FrobReg, JacReg), while others first apply a projection (DoubleBack, SpectReg) and hence we compute the gradient of a single scalar. Due to the linarity of the gradient, both scenarios can be interpreted as regularizing the Jacobian matrix. If our regularizer is of the form $\|J_g^T w\|^2$, then

$$\|\frac{\partial}{\partial x}\langle g(x), w\rangle\| = \|(\frac{\partial}{\partial x}g(x))^T w\|$$

While in theory the projection should yield a negligible increase of computational burden, current tensor libraries all employ backwards automatic differentiation, which means that they do not reuse shared intermediate results when calculating the vector gradients, i.e. the rows of the Jacobian. Consequently, FrobReg/JacReg is vastly slower and vastly more memory-hungry than their peers that use projection, the performance ratio scaling with the number of class labels.

The following Claim is an easy consequence of the linearity of expectation:

Claim 1. *If R is a distribution of row vectors with covariance matrix I_m, then $E_{r\sim R}[\|rJ\|_2^2] = \|J\|_F^2$.*

Consequently, spherical SpectReg is an unbiased estimator of the squared Frobenius norm of the Jacobian and so is SpectReg, up to constant scaling.

2.5. Easy implementation

Calculating gradient regularization terms is made easy and fast by modern tensor libraries. Figure 1 shows the full Tensorflow implementation of the two regularizers that we focus on.

```
doubleback_loss = tf.reduce_sum(
    tf.square(tf.gradients(loss, [input])[0]))

projector = tf.random.normal((BATCH_SIZE, OUTPUT_DIM))
proj = tf.reduce_sum(
    logits * projector, axis=1)
spectreg_loss = tf.reduce_sum(
    tf.square(tf.gradients(proj, [input])[0]))
```

Figure 1. TensorFlow implementations for the DoubleBack and SpectReg regularizers

3. Related Work

The idea to control gradient norms with respect to the inputs first appeared in [2] who called it *Double Backpropagation*. However, due to slow hardware and a missing technical apparatus for symbolic differentiation at that time, they only evaluated the idea on networks with two hidden layers. The *TangentProp* variant was introduced in [14] where the magnitude of change is controlled only in some input space directions, corresponding to selected invariances. This, for example, can be used to promote rotational invariance by using the direction of infinitesimal rotation in pixel space.

Jacobian Regularization

The *Jacobian Regularizer* introduced by [16] works by regularizing the Frobenius norm of the Jacobian of the softmax output with respect to the input. For smaller networks the authors symbolically compute the Jacobian. For larger networks they employ a Frobenius norm based regularization for each layer individually, for performance reasons. Such layer-wise regularization of the Frobenius norm was also employed in [3].

Robustness to adversarial examples

The Double Backpropagation formula was rediscovered by [11] who named it *DataGrad*. However, the authors do not actually implement the formula, rather, a finite difference approximation is used: after finding adversarial samples by gradient descent, they penalize large changes in the loss function between the data point and its adversarially perturbed version. This approximation improves robustness to adversarial sampling. Recently, [15] use symbolically computed Double Backpropagation to improve model *interpretability* as well as robustness against adversarial noise.

Sobolev training

A generalized form of the Double Backpropagation idea is when an oracle gives information about the gradients (or even the Hessian) of the target function, and we incorporate this into a loss term. This is investigated in [1], focusing on *distillation* and *synthetic gradients*, applications where such an oracle is available. In contrast to this approach, we demonstrate the counter-intuitive fact that in the absence of such an oracle, simply pushing gradient norms toward zero at the data points can already

have a beneficial regularization effect.

Spectral norm regularization

Concurrent with our work [20] enforce smoothness by penalizing an upper bound to the spectral norm of the neural network at all points of the input space. The price of this global effect is that the upper bound is potentially very far from the actual spectral norm. Our approach, on the other hand, focuses on the data manifold and provides input specific regularization effect.

Sensitivity and generalization

Very recently, [10] perform a large scale empirical study that shows correlation between sensitivity to input perturbations and generalization. They find that for networks that generalize well, the Frobenius norm of the Jacobian tends to have smaller norm. Their findings strengthen the motivation for gradient regularizaton.

4. Experiments

We compare gradient regularization with various other methods in order to position this technique on the landscape. Our results show that SpectReg and DoubleBack consistently increase accuracy of baselines on a wide variety of classification tasks.

Our experiments use reduced training sets as we found that gradient regularization is particularly useful in such scenarios. By *small MNIST* and *small CIFAR-10/100* we refer to these datasets restricted to 200 randomly selected training points per class. [1] We also run experiments on TinyImageNet-200 [2], a standard reduced version of the ImageNet dataset containing 500 training images per class. We also use a synthetic dataset *SIN* generated from function $f(x) = \sin(5x)$, with 100 training points sampled uniformly from $[-1, 1]$ and added Gaussian noise ($\sigma = 0.1$). None of our experiments employ data augmentation, to force the model to generalize from limited data. Each dataset is trained on a baseline network that performs reasonably well on the full set, without any attempt to reach the state of the art.

- For MNIST, we implement a standard modern incarnation of the classic LeNet-5

[1] Note that our small datasets are not static: at the beginning of each experiment we randomly select a new subset of the training corpus, hence our methods cannot overfit to the selected points.

[2] https://tiny-imagenet.herokuapp.com

architecture [9], that has 61706 parameters and that achieves 99.1% test accuracy on the full (unaugmented) MNIST.

- For CIFAR-10/100, we use a residual network (ResNet) [6], consisting of three levels (with 48, 96 and 192 filters). This network has 2.5 million parameters and achieves 93.71% test accuracy on the full (augmented) CIFAR-10.
- For TinyImagenet-200, we use the ResNet-18 architecture that has been used for the ImageNet training in [6] with appropriate modification for 200 labels. The network has around 11.3 million parameters.
- For SIN, we use an MLP network with 5 dense layers of 64 neurons and ReLU nonlinearity, followed by a single linear neuron.

An individual experiment consists of 10 runs of the same setup, each time with a different randomly chosen training set. All reported numbers are the mean of these runs evaluated on 10000 test points, accompanied with the standard deviation in parentheses. Hyperparameters are tuned on 10000 validation points. The hyperparameters that are not explicitly discussed are all tuned for the baseline model and kept constant for the regularized versions.

Weight decay and gradient regularization

Weight decay is probably still the most widespread regularization method. On small MNIST, gradient regularization yields more accuracy gain than weight decay. For small CIFAR-10/100, weight decay is more important than gradient regularization, but the latter still brings significant benefit. In our exploratory experiments, DoubleBack slightly outperformed other variants on MNIST, however, we could not tune it on CIFAR to rival SpectReg. Overall, we found SpectReg to be the most robust regularizer and here we only present accuracy results for this variant, in Table 1.

Table 1. Gradient regularization alongside weight decay.

	Without weight decay		With weight decay		
Dataset	Baseline	SpectReg	Baseline	SpectReg	WD
small MNIST	97.25 (0.22)	97.69 (0.11)	97.40 (0.15)	**97.73** (0.13)	0.0005
small CIFAR-10	48.27 (0.82)	50.41 (0.65)	55.63 (2.06)	**59.24** (1.48)	0.003
small CIFAR-100	37.81 (0.33)	41.80 (0.70)	49.56 (2.96)	**52.49** (0.65)	0.003

Table 2. Comparison of Dropout and Batchnorm versus DoubleBack and SpectReg.

	Dataset	NoGR	SpectReg	DoubleBack
Baseline	small MNIST	96.99 (0.15)	97.59 (0.13)	97.56 (0.24)
Batchnorm	small MNIST	96.89 (0.23)	96.94 (0.27)	96.89 (0.22)
Dropout	small MNIST	97.29 (0.19)	97.65 (0.14)	**97.98** (0.12)

Gradient Regularization Compared with Dropout and Batch Normalization

On small MNIST, we find that both DoubleBack and SpectReg outperform both Dropout [17] and Batch Normalization [7], two well established techniques. We observe further accuracy gain when Dropout and DoubleBack/SpectReg is combined. Table 2 summarizes our results. While in all other experiments the MNIST baseline includes dropout, here the baseline is only regularized with weight decay.

Gradient Regularization versus Confidence Penalty

One recent regularization technique is *Confidence Penalty (CP)* [12]. It penalizes the negentropy of the softmax output, and brings small but consistent accuracy improvements on many tasks. We find that gradient regularization performs better than CP on small MNIST. They seem to perform orthogonal tasks and can be combined to obtain further improvements. Table 3 shows our results. We also make these comparisons on small CIFAR-10 and small CIFAR-100 in Table 4.

Table 3. Confidence Penalty versus SpectReg and DoubleBack on small MNIST.

	Dataset	Baseline	SpectReg	DoubleBack
No CP	small MNIST	97.39 (0.11)	97.79 (0.12)	97.89 (0.12)
CP	small MNIST	97.57 (0.15)	97.89 (0.12)	**98.07** (0.09)

Table 4. Comparing SpectReg, DoubleBack and CP on small CIFAR-10/100.

Dataset	Baseline	SpectReg	DoubleBack	CP
small CIFAR-10	55.63 (2.06)	**59.24** (1.48)	57.45 (1.79)	58.30 (1.79)
small CIFAR-100	49.56 (2.96)	**52.49** (0.65)	48.72 (2.84)	51.04 (1.29)

The effect of training set size

Regularizers are more useful for smaller training sets, however, as we show in Figure 2 Left they maintain a significant benefit even for as much as 20000 training points on MNIST. DoubleBack performs best for all sizes, followed by SpectReg.

Scaling with the number of labels

As Table 5 shows, gradient regularization works well even for the significantly more complex TinyImageNet-200 dataset. Note that in our various experiments the number of class labels range from 10 to 200, and the effect of gradient regularization does not fade with increased label count. That is, SpectReg is beneficial regardless of whether the number of matrix rows projected is 10 or 200.

Table 5. Results on the TinyImageNet-200 dataset.

	Top-1 accuracy		Top-5 accuracy	
Dataset	Baseline	SpectReg	Baseline	SpectReg
TinyImageNet-200	44.62 (0.58)	**50.76** (0.59)	70.20 (0.58)	**75.93** (0.40)

Table 6. Comparing SpectReg, FrobReg and JacReg on small MNIST.

Dataset	SpectReg	JacReg	FrobReg
small MNIST	**97.79%** (0.12)	97.63% (0.15)	97.76% (0.13)

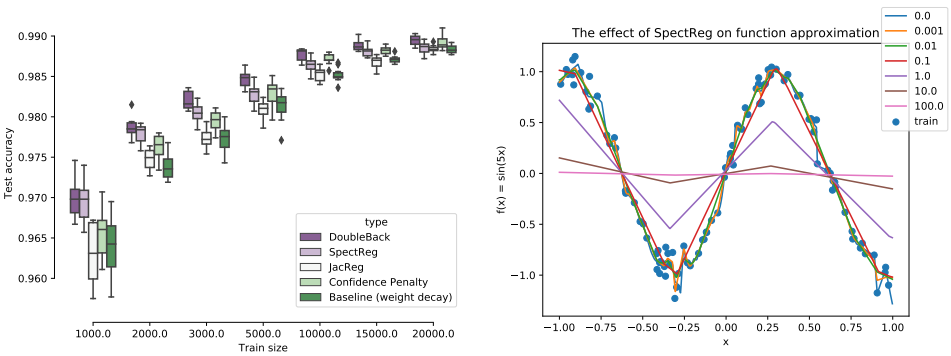

Figure 2. Left: Comparison of various regularization methods on Small MNIST using different training set sizes. DoubleBack performs best for all sizes. **Right:** Increasing the weight of the SpectReg regularizer on a small synthetic dataset. Although the gradient is controlled only in 100 training points, the whole manifold becomes smoother.

SpectReg vs. DoubleBack

Table 7 compares SpectReg and DoubleBack. While DoubleBack is consistently better on MNIST, it greatly degrades on the more complex CIFAR data. On CIFAR-

100, DoubleBack even reduces generalization. We hypothesize that this is because DoubleBack is applied after the softmax layer and the quick vanishing effect makes it very hard to properly tune the model. [3] In comparison, SpectReg requires minimal tuning and even a "wrong" weight yields close to optimal performance. Also note the large standard deviation of the DoubleBack results on CIFAR. This is possibly due to the fact that regularization is only applied to one particular projection of the Jacobian and random perturbations (like selecting a different subset of the data) can have larger effect. The robustness of SpectReg is what we believe makes it a better candidate in applications.

Table 7. Comparing SpectReg and DoubleBack.

Dataset	Baseline	SpectReg	DoubleBack
small MNIST	97.39 (0.11)	97.79 (0.12)	**97.89** (0.12)
small CIFAR-10	55.63 (2.06)	**59.24** (1.48)	57.45 (1.79)
small CIFAR-100	49.56 (2.96)	**52.49** (0.65)	48.72 (2.84)

Approximating the Frobenius norm of the Jacobian does not decrease accuracy

We compare SpectReg, JacReg and FrobReg. We can see from Table 6 (and especially from Figure 2 Left) that it is more beneficial to control the Jacobian on the logits (SpectReg, FrobReg), rather than on the probabilities (JacReg). We also conclude that minimizing a random projection of the Jacobian (SpectReg) does not lead to a loss in accuracy, compared with the full calculation of the Jacobian (FrobReg). We know that the SpectReg loss term is an unbiased estimator (up to a constant factor) of the FrobReg loss term, and these experiments indicate that the approximation of the expectation (FrobReg) with a single sample (single random projection applied by SpectReg) is sufficient.

Gradient Regularization's effect on the magnitude of network parameters

Figure 3 shows the effect of DoubleBack weight λ on accuracy, and on the squared L2 weight norm. There is a positive correlation between λ and accuracy until a phase transition point where accuracy collapses. For small MNIST, this phase transition point is around DoubleBack $\lambda = 20$, regardless of the amount of weight decay.

Without weight decay, DoubleBack acts as a kind of weight decay itself, inasmuch as increasing the DoubleBack weight decreases the weight loss of the trained network. For models with weight decay, the relationship is more complex: below the phase transition, DoubleBack counteracts weight decay, above the phase transition it reinforces its effect.

[3] Indeed, even on MNIST, we found DoubleBack to be extremely sensitive to its weight.

The fact that DoubleBack affects a global attribute of the network such as total weight loss supports the hypothesis that its effect is global, not limited to the immediate neighborhood of the training points where the gradient norm is regularized. In the next subsection we give a clear demonstration of this phenomenon on our synthetic task.

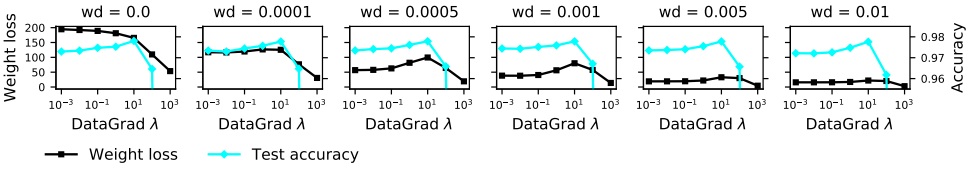

Figure 3. Black: weight loss, primary y-axis. Blue: accuracy, secondary y-axis. Each chart uses different weight decay. Increasing λ increases accuracy, until a phase transition where accuracy collapses. Below this transition point, the weight loss of the trained network increases with λ, except for the case of zero weight decay, where DoubleBack takes the role of weight decay.

Local gradient control does not lead to pathological gradient landscape

A reasonable objection to gradient regularization is that it controls the gradients only in the training points. A highly overparametrized network is capable of representing a "step function" that is extremely flat around the training points and contains unwanted sudden jumps elsewhere. Such a solution results in low gradient loss with high actual gradient norms in some points. However, all our experiments indicate that the learning process of neural networks struggles to find such pathological solutions and instead smoothens the function on its whole domain. We emphasize that in all of our experiments except for TinyImagenet, the datasets are so small that the models are inevitably overfitting, yielding close to 100% train accuracy. In such setting, it is remarkable that gradient control does not overfit.

Figure 2 Right demonstrates this phenomenon on the SIN dataset. Our network is highly overparametrized network that has the capacity to simultaneously achieve zero reconstruction loss and zero SpectReg loss on the training data. However, training converges to a smooth function instead.

While too strong regularization degrades the approximation, a well tuned SpectReg yields much better fit to our target function. Table 8 shows the mean squared error on the test set for various values of SpectReg λ. SpectReg can reduce the mean squared error of the baseline model by a factor of 10.

Table 8. Mean squared error (MSE) on SIN for various SpectReg weights. The optimal weight is 0.03 which yields a reduction in MSE by a factor of 10.

SpectReg λ	0	0.001	0.003	0.01	0.03	0.1	0.3	1	3
MSE (1e-5)	16.6	24.5	2.5	2.3	**1.6**	3.4	74.9	497.7	1264.4

5. Conclusion

Our paper presents evidence that gradient regularization can increase classification accuracy, especially for smaller training set sizes. We introduce *Spectral Regularization* and after comparing it to other gradient regularization schemes, we find that it is the most promising variant. Despite the fact that gradient control is applied only at the training points, we find that stochastic gradient descent converges to a solution where gradients are globally controlled. Even for very small training set sizes, the regularized models become smoother on the whole data manifold.

Below we list some of what we see as promising further research on gradient regularization:

- We considered gradients calculated with respect to inputs. In the future, we plan to investigate gradients calculated with respect to hidden activations.

- Our understanding of the counterintuitive interaction between gradient regularization and stochastic gradient descent is still limited. A full-blown theory may yet be out of sight, but it is reasonable to expect that well-chosen experiments on synthetic benchmarks will lead to important new insights.

Acknowledgement

The project was supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.3-VEKOP-16-2017-00002). Furthermore, it was supported by the Hungarian National Excellence Grant 2018-1.2.1-NKP-00008, the HU-MATHS-IN project (EFOP 3.6.2-16.), as well as the ERC grant agreement 617747 (FP7/2007-2013).

References

[1] Wojciech M. Czarnecki, Simon Osindero, Max Jaderberg, Grzegorz Swirszcz, and Razvan Pascanu. Sobolev training for neural networks. In *NIPS*, pages 4281–4290, 2017.

[2] H. Drucker and Y LeCun. Double backpropagation: Increasing generalization performance. In *Proceedings of the International Joint Conference on Neural Networks*, volume 2, pages 145–150, Seattle, WA, July 1991. IEEE Press.

[3] Shixiang Gu and Luca Rigazio. Towards deep neural network architectures robust to adversarial examples. *CoRR*, abs/1412.5068, 2014.

[4] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*. Curran Associates, Inc., December 2017. arxiv: 1704.00028.

[5] László Györfi, Michael Kohler, Adam Krzyzak, and Harro Walk. *A Distribution-Free Theory of Nonparametric Regression.* Springer series in statistics. Springer, 2002.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[7] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 448–456. JMLR.org, 2015.

[8] Daniel Jakubovitz and Raja Giryes. Improving DNN robustness to adversarial attacks using jacobian regularization. *CoRR*, abs/1803.08680, 2018.

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

[10] Roman Novak, Yasaman Bahri, Daniel A. Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Sensitivity and generalization in neural networks: an empirical study. In *International Conference on Learning Representations*, 2018.

[11] Alexander G. Ororbia II, Daniel Kifer, and C. Lee Giles. Unifying adversarial training algorithms with data gradient regularization. *Neural Computation*, 29(4):867–887, 2017.

[12] Gabriel Pereyra, George Tucker, Jan Chorowski, Lukasz Kaiser, and Geoffrey E. Hinton. Regularizing neural networks by penalizing confident output distributions. *CoRR*, abs/1701.06548, 2017.

[13] Lorenzo Rosasco, Silvia Villa, Sofia Mosci, Matteo Santoro, and Alessandro Verri. Nonparametric sparsity and regularization. *Journal of Machine Learning Research*, 14(1):1665–1714, 2013.

[14] Patrice Y. Simard, Bernard Victorri, Yann LeCun, and John S. Denker. Tangent prop - A formalism for specifying selected invariances in an adaptive network. In *NIPS*, pages 895–903. Morgan Kaufmann, 1991.

[15] A. Slavin Ross and F. Doshi-Velez. Improving the Adversarial Robustness and Interpretability of Deep Neural Networks by Regularizing their Input Gradients. *ArXiv e-prints*, November 2017.

[16] Jure Sokolic, Raja Giryes, Guillermo Sapiro, and Miguel R. D. Rodrigues. Robust large margin deep neural networks. *IEEE Trans. Signal Processing*, 65(16):4265–4280, 2017.

[17] 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*, 15:1929–1958, 2014.

[18] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2013.

[19] G. Wahba. *Spline Models for Observational Data*. Society for Industrial and Applied Mathematics, Philadelphia, 1990.

[20] Yuichi Yoshida and Takeru Miyato. Spectral norm regularization for improving the generalizability of deep learning. *CoRR*, abs/1705.10941, 2017.