L 1-norm double backpropagation adversarial defense

Ismaïla Seck, Gaëlle Loosli, Stephane Canu

L_1-norm double backpropagation adversarial defense

Ismaila Seck[1,2], Gaëlle Loosli[2,3] and Stéphane Canu[1]

1- Normandie Univ, INSA Rouen, UNIROUEN, UNIHAVRE, LITIS, France

2- UCA - LIMOS UMR 6158 CNRS
Clermont-Ferrand, France

3- PobRun, Brioude, France

Abstract. Adversarial examples are a challenging open problem for deep neural networks. We propose in this paper to add a penalization term that forces the decision function to be flat in some regions of the input space, such that it becomes, at least locally, less sensitive to attacks. Our proposition is theoretically motivated and shows on a first set of carefully conducted experiments that it behaves as expected when used alone, and seems promising when coupled with adversarial training.

1 Introduction

Deep learning algorithms have set the state-of-the-art in several domains among which image classification. However, [1] showed that it is possible, and relatively easy, to fool Deep Neural Networks by adding to the inputs a particular perturbation. This added perturbation may be such that the disturbed inputs and the original ones are very close according to some metrics, but are assigned to different classes. Several defense mechanisms have been introduced to prevent such a behavior, but their efficiency is limited to specific cases so that the general problem of preventing the existence of adversarial examples remains open.

One idea is to have a constant output over a region around known training points. For a differentiable function, that means an arbitrarily chosen norm of the output's gradient with respect to the input should be 0 or at least as small as possible over that region. Our claim is that by using adversarial training and by penalizing the gradient's norm of the output with respect to the input, the robustness of the model can be improved. The L_1-norm is chosen and this choice will be theoretically motivated by calculus.

2 Related work

The adversarial examples were first presented in [2], and the principle of adversarial training was introduced at the same time. Adversarial training consists in augmenting the dataset with potentially adversarial points. But it was impractical, since the method used to generate adversarial samples, L-BFGS, was too slow. Adversarial training became more convenient to use with the introduction of Fast Gradient Sign Method (FGSM) [1], which is much faster and generates adversarial examples with a good success rate. Moreover, using a first order expansion of Taylor series, adversarial training can be seen as an ℓ_1-norm penalty

of the derivative of the loss with respect to the inputs [3]. Note that *regularization functional which penalize derivatives of the resulting classifier function are not typically used in deep learning* [4].

The idea of penalizing the gradient of the output with respect to the input was introduced by [5] under the name *double backpropagation* and later used in [6] to find large connected regions of the error function called flat minima. The ultimate goal was the improvement of the generalization of their models, which differs slightly from our goal here. In *double backpropagation*, the ℓ_2-norm of the loss is penalized. Using the Energy loss function, it was stressed that the penalization of the loss would have little to no effect when the classification is good. To balance out that effect, the multiplicative parameter of the penalization ought to be large enough.

The difference between double backpropagation and our gradient penalization is that we penalize the ℓ_1-norm of the gradient of each output with respect to the input while, in backpropagation, the ℓ_2-norm of the loss is penalized. Hence, we should not have the problem that occurs when the penalization term is multiplied by a small error vector. Nevertheless, this might not be a problem when using a different loss function. Although empirical evidence show double backpropagation's efficiency to enhance generalization, it is insufficient to defend against adversarial examples. That is what [7] highlights saying *limiting sensitivity to infinitesimal perturbation [e.g., using double backpropagation 5] only provides constraints very near training examples, so it does not solve the adversarial perturbation problem.* But there are evidence that coupling that gradient penalty with adversarial training increases the robustness.

3 Defensive gradient penalty

It has been shown that that maliciously crafted examples can fool the deep learning classifiers and lead them to misclassify those examples with high confidence. In addition to the adversarial training, a gradient penalization is proposed here to make the models more robust. Let $\mathbf{x}$ be an input image stored in a vector belonging to the input space $\mathcal{X} = [0,1]^d$, where d is the dimension of $\mathbf{x}$ and let y be the target, a one-hot label, associated with $\mathbf{x}$. Let f, a differentiable function, $f : \mathcal{X} \to \mathbb{R}^c$ where c is the number of classes, such that the decision function $D(x) = argmax(f(x))$ represents our classifier. And let $\mathcal{L}(f(\mathbf{x}), \mathbf{y})$ be a common differentiable loss function.

For a given input $\mathbf{x}$, a perturbation direction $\mathbf{v}$ and a positive scalar ε, the first order expansion of the transfer function i-th component, f_i, of the neural network is:

$$f_i(\mathbf{x} + \varepsilon\mathbf{v}) = f_i(\mathbf{x}) + \varepsilon\mathbf{v}^{\mathrm{T}}\nabla_{\mathbf{x}} f_i(\mathbf{x}) + \circ(\varepsilon\|v\|). \tag{1}$$

Considering the FGSM attack [1] with $\mathbf{v} = \text{sign}(\nabla_{\mathbf{x}}\mathcal{L}(f(\mathbf{x}), \mathbf{y})$, we have $\mathbf{v}^{\mathrm{T}} = \{\pm 1\}^d$, $|f_i(\mathbf{x} + \varepsilon\mathbf{v}) - f_i(\mathbf{x})| \approx |\varepsilon\mathbf{v}^{\mathrm{T}}\nabla_{\mathbf{x}} f_i(\mathbf{x})| \leq \varepsilon\|\nabla_{\mathbf{x}} f_i(\mathbf{x})\|_1 = \varepsilon w^{\mathrm{T}}\nabla_{\mathbf{x}} f_i(\mathbf{x})$ for $w = \text{sign}(\nabla_{\mathbf{x}} f(\mathbf{x}))$. So that, in this case, finding weights that minimize sensitivity to infinitesimal perturbation of the input can be done by minimizing the ℓ_1-norm of the gradient of each component of the transfer function with respect to the

input. Our approach is therefore to penalize the ℓ_1-norm of the gradient of each coordinate $f_i(x)$ not only on the original point but also on the potentially adversarial point generated. If we manage to have 0 as the ℓ_1-norm of those gradients at two points that are very close, then the output of the classifier is almost constant along the segment joining those two points. Indeed we have:

$$0 \leq |f_i(\mathbf{x} + \varepsilon\mathbf{v}) - f_i(\mathbf{x})| \leq \varepsilon \sup_{t \in [0,\varepsilon]} \|\nabla_{\mathbf{x}} f_i(\mathbf{x} + t\mathbf{v})\|_1 \tag{2}$$

The regularization coefficient being small (ε has to be small for $\mathbf{x} + \varepsilon\mathbf{v}$ to be considered as an adversarial example) the regularization is not used at its full potential. In order to increase the regularization coefficient, an explicit penalization of the ℓ_1-norm is added to the loss function. Hence the analogy with [5], which penalize the ℓ_2-norm of the gradient of the loss while in this paper the sum of the ℓ_1-norm of the outputs $(f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_c(\mathbf{x}))$ with respect to the input is directly penalized. But, it is very hard to make derivatives very small using only the penalization of the gradient when penalizing on the training points only, and it is also ineffective against adversarial examples.

Better results are obtained when we penalize the gradients on the training set and also on the adversarial example near the training set. Doing so each training point is associated with an adversarial example, as in classical adversarial training, and we penalize the gradient on both these points. Let $\mathbf{x}$ be a point of the training set and $\mathbf{x}_{adv}$ the adversarial example computed from $\mathbf{x}$ using the FGSM. Let us assume that we train the classifier for long enough so that $\|\nabla_{\mathbf{x}} f_i(\mathbf{x})\|_1 \approx 0$ and $\|\nabla_{\mathbf{x}} f_i(\mathbf{x}_{adv})\|_1 \approx 0$ for all $i = 1, \ldots, c$, since those two points are close enough, the variation of f on the segment joining $\mathbf{x}$ and $\mathbf{y}$ is so small that the class does not change. The ideal training progress is presented in figure 1. See section 4.2 for comparison of the results of between classical adversarial training, and the new variant introduced in this paper. The loss function used is:

$$\begin{aligned}
\mathcal{L}_{gp}(\mathbf{x}, \mathbf{y}) &= \mathcal{L}(f(\mathbf{x}), \mathbf{y}) + \lambda \sum_{i=1}^{c} \|\nabla_{\mathbf{x}} f_i(\mathbf{x})\|_1 \\
&= \mathcal{L}(f(\mathbf{x}), \mathbf{y}) + \lambda \sum_{i=1}^{c} \sum_{j=1}^{d} |J_{ij}| \\
&= \mathcal{L}(f(\mathbf{x}), \mathbf{y}) + \lambda \|J\|_{1,1},
\end{aligned} \tag{3}$$

where d denotes the dimension of the input image, c, the number of neurons on the output layers of the neural network *i.e.* the number of classes, λ, the penalization parameter, $J_{ij} = \frac{\partial f_i(\mathbf{x})}{\partial x_j}$, the Jacobian matrix of f and $\|.\|_{1,1}$, the entry wise L_1-norm.

4 Experimental Results

All experiments here are built upon 3 models (referred as A,B and C for simplicity) that are described in table 1. The first experiments is our proof of concept. It shows that the penalization helps the defense. Since alone it's still not enough to propose a robust model, the second experiment explores the coupling with adversarial training.

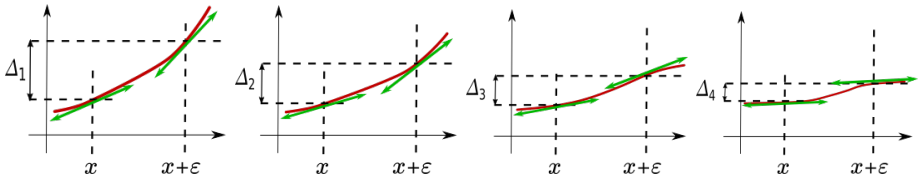

Fig. 1: Figure showing the ideal progression of the training for one training point x, and associated adversarial points. At first, the difference $\Delta = f(x+\varepsilon) - f(x)$ of ordinates and gradients norm at x and $x+\varepsilon$ are high. During the course of the training, Δ decreases, making the value at x and $x+\varepsilon$ closer. The gradients' norm also decrease. In the end, we have almost the same value for f at those points, and the gradients are nearly 0. In those conditions, we have $\Delta_1 > \Delta_2 > \Delta_3 > \Delta_4 \approx 0$, and the variation between x and $x+\varepsilon$ is then very small.

model A	model B	model C
conv(64,8,2)	conv(128,3,1)	FC(512)
conv(128,6,2)	conv(64,3,2)	FC(256)
conv(128,5,1)	FC(128)	FC(128)
FC(10)	FC(10)	FC(10)
710,218	1,460,938	567,434

Table 1: Description of the models used in this paper: conv(nf, k, s) represents a convolutional layer with nf filters, of size $k \times k$ applied with a stride s. FC(nn) represents a fully connected layer with nn neurons. All activations are ReLU except the output layer. The numbers in the last line represent the number of parameters.

4.1 Proof of concept

The goal of this experiment is to show that penalizing the gradient improves a lot the robustness of the model while keeping the efficiency on the clean data, and the more we penalize, the more the effect is visible. In our tests, we observe that we double (at least) the number of correctly classified adversarial examples.

We attack the model using the FGSM in a white-box setup in which the attacker has access to the parameters of the model and to the test set. The value $\varepsilon = 0.3$ is chosen as the intensity of the pixel wise perturbation allowed to the attacker as it is often the case for the data used in this experiment. Several values of λ are used, allowing to have a notion of the influence of that hyper-parameter on the performance on adversarial samples. The MNIST dataset and three different architectures are used. The training and testing split of Keras is used, and the training set is then split into two groups, one of 55000 used to train models and the remaining 5000 are used as a validation set. Models are trained until their accuracy on the validation set stop increasing (difference of accuracy between 2 epochs less than 0.0001) after 10 epochs, or until 100 epochs of training on the clean data is achieved. This process is repeated 10 times, the mean value and the standard deviation are recorded in table 2. The program always terminated before 100 hundred epochs were reached, around the 20^{th} epoch. The optimization method used is Adam with a learning rate of 0.001,

$\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$, a label smoothing of 0.1 was also used.

λ		50	25	10	1	0.1
A	clean	99.37 ± 0.04	99.36 ± 0.03	99.38 ± 0.04	99.35 ± 0.03	99.35 ± 0.06
	adv	45.09 ± 5.36	44.96 ± 4.44	46 ± 10.16	28.83 ± 9.34	22.64 ± 5.02
B	clean	98.99 ± 0.05	98.99 ± 0.05	98.98 ± 0.07	98.94 ± 0.07	98.90 ± 0.05
	adv	6.35 ± 3.258	4.69 ± 1.91	3.69 ± 0.73	3.00 ± 1.14	3.17 ± 0.82
C	clean	98.63 ± 0.11	98.67 ± 0.05	98.57 ± 0.06	98.55 ± 0.05	98.60 ± 0.08
	adv	29.92 ± 14.24	25.76 ± 11.96	18.24 ± 9.97	19.99 ± 5.63	13.66 ± 6.64

Table 2: Accuracy on clean test and adversarial test point of model. We observe that the more we penalize, the more robust the model becomes. However, while the gain is significant, it's clearly not enough to call it a robust model.

4.2 Coupling with adversarial training

ε		0.05		0.1		0.2		0.3	
		clean	adv	clean	adv	clean	adv	clean	adv
A	adv_train	99.36 ±0.05	98.57 ±0.11	99.34 ±0.06	97.50 ±0.26	99.27 ±0.03	96.39 ±0.37	99.30 ±0.06	95.72 ±0.19
	adv_train_gp	99.38 ±0.028	**98.73** ±0.072	99.35 ±0.035	97.45 ±0.30	99.25 ±0.061	96.63 ±0.30	99.26 ±0.06	**97.60** ±0.34
B	adv_train	98.94 ±0.079	98.37 ±0.48	99.07 ±0.03	98.22 ±0.61	98.96 ±0.05	98.55 ±0.26	98.85 ±0.05	98.67 ±0.07
	adv_train_gp	99.05 ±0.05	98.18 ±0.15	99.06 ±0.05	97,64 ±1.04	98.95 ±0.02	98.30 ±0.49	98.84 ±0.07	97.80 ±1.11
C	adv_train	98.98 ±0.06	96.26 ±0.22	98.68 ±0.06	93.56 ±0.47	98.59 ±0.07	93.24 ±0.48	98.48 ±0.1	95.52 ±0.74
	adv_train_gp	98.79 ±0.04	96.36 ±0.11	98.87 ±0.07	**94.56** ±0.16	98.69 ±0.09	93.73 ±0.75	98.42 ±0.08	**96.62** ±0.51

Table 3: Table showing the performance of models A, B and C, trained with adversarial training and adversarial training plus gradient penalty ($\lambda = 10$)[1]. We can see that, the adv_gp performs better for model A and $\varepsilon = 0.3$, and is sensibly equal for other values of ε. The large standard deviation values might be an indication that the training was still in progress when epoch 100^{th} epochs was reached.

The previous experiments show that our penalization has an interesting effect on the robustness of the models. Now we explore the coupling with adversarial training, and the hope is that the penalization has the same impact, in order to obtain more robust models than adversarial training alone. We train the same model using adversarial training with the classical loss function and with the gradient penalty in addition to that loss. We train for one hundred epochs. The adversarial training consists in adding adversarial examples to the original training set to make the models more robust. Typically, the FGSM is used to generate those adversarial examples due to its practicality. So it will be used in the following experiments with a clipping to make sure that adversarial examples remains in $\mathcal{X}$. During training, for each image of a batch, an adversarial image is generated and added to the batch with the same target as the original image associated. A label smoothing of 0.1 is also used for this experiment.

[1]The loss function used here is not exactly the one proposed in 3. Instead of $\|J\|_{1,1}$, $\sum_{j=1}^{d} |\sum_{i=1}^{c} \frac{\partial f_i(\mathbf{x})}{\partial x_j}|$ is used due to time constraints.

Table 3 shows results for $\lambda = 10$. It turns out that performances are not as impressive as they were without adversarial training, even though for some settings, the penalization helps. One reason could be that λ is not high enough. However for computing time reason, we could not conduct the same set of experiments with different values at the same level of care. Hence we propose some partial results to argue our point in table where we see that increasing λ provides better results. We note also that we increased the number of epoch up to 150.

λ	0	10	50	100	200	1000
clean	98.48 $\pm$0.10	98.42 $\pm$0.08	98.49 $\pm$0.06	98.59 $\pm$0.10	98.60 $\pm$0.11	98.62 $\pm$0.07
adv	95.52 $\pm$0.74	96.62 $\pm$0.51	**97.25** $\pm$0.05	96.88 $\pm$0.51	96.71 $\pm$0.19	94.65 $\pm$0.79

Table 4: For model C, we fixed $\varepsilon = 0.3$ and different λ, on the same setting as previous experiment. $\lambda = 0$ refers to the adversarial learning alone. We observe improvement when λ increases, until it fails when too high.

5 Conclusion and future work

In this paper we propose to improve deep neural networks' robustness to adversarial examples by adding a penalization term. We show that penalizing the gradients of the output with respect to input, or in other words the Jacobian, can have an effect defending against adversarial but does not suffice. Coupled with adversarial training to give L_1-norm double backpropopagation adversarial defense, we provide good hints that it is still a good approach. In future work we will extend the experimental part to more sets of parameters and other datasets.

References

[1] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

[2] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[3] Carl-Johann Simon-Gabriel, Yann Ollivier, Bernhard Schölkopf, Léon Bottou, and David Lopez-Paz. Adversarial vulnerability of neural networks increases with input dimension. *arXiv preprint arXiv:1802.01421*, 2018.

[4] Matthias Hein and Maksym Andriushchenko. Formal guarantees on the robustness of a classifier against adversarial manipulation. In *Advances in Neural Information Processing Systems*, pages 2266–2276, 2017.

[5] Harris Drucker and Yann Le Cun. Improving generalization performance using double backpropagation. *IEEE Transactions on Neural Networks*, 3(6):991–997, 1992.

[6] Sepp Hochreiter and Jürgen Schmidhuber. Simplifying neural nets by discovering flat minima. In *Advances in neural information processing systems*, pages 529–536, 1995.

[7] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 582–597. IEEE, 2016.