

# Unifying Adversarial Training Algorithms with Data Gradient Regularization

**Alexander G. Ororbia II**

*agol09@ist.psu.edu*

**Daniel Kifer**

*dkifer@cse.psu.edu*

**C. Lee Giles**

*giles@ist.psu.edu*

*Pennsylvania State University, University Park, PA 16802, U.S.A.*

Many previous proposals for adversarial training of deep neural nets have included directly modifying the gradient, training on a mix of original and adversarial examples, using contractive penalties, and approximately optimizing constrained adversarial objective functions. In this article, we show that these proposals are actually all instances of optimizing a general, regularized objective we call DataGrad. Our proposed DataGrad framework, which can be viewed as a deep extension of the layerwise contractive autoencoder penalty, cleanly simplifies prior work and easily allows extensions such as adversarial training with multitask cues. In our experiments, we find that the deep gradient regularization of DataGrad (which also has L1 and L2 flavors of regularization) outperforms alternative forms of regularization, including classical L1, L2, and multitask, on both the original data set and adversarial sets. Furthermore, we find that combining multitask optimization with DataGrad adversarial training results in the most robust performance.

## 1 Introduction ---

Deep neural architectures are highly effective at a vast array of tasks, both supervised and unsupervised. However, recently, it has been shown that deep architectures are sensitive to certain kinds of perturbations of the input, which can range from being barely perceptible to quite noticeable (even semirandom noise), as in Nguyen, Yosinski, and Clune (2014). Samples containing this type of noise, called adversarial examples (Szegedy et al., 2013), can cause a trained network to confidently misclassify its input. While a variety of ways can be used to generate adversarial samples, the fastest and most effective approaches in the literature are based on the idea of using backpropagation to acquire the derivative of the loss with respect to an input image (i.e., the gradient) and adding a small multiple of the gradient to the image.

Earlier work suggested adding a regularization penalty on the deep gradient (Goodfellow, Shlens, & Szegedy, 2014; Gu & Rigazio, 2014) but had difficulty in computing the derivative (with respect to the weights) of the gradient, which is necessary for gradient-descent optimization algorithms. Instead, approximations were used. One was a shallow layerwise gradient penalty (Gu & Rigazio, 2014), which had also been used for regularizing contractive autoencoders (Rifai et al., 2011). Meanwhile, Lyu, Huang, and Liang (2015) presented a heuristic algorithm for this objective.

Here we provide an efficient, deterministic backpropagation style algorithm for training with a wide variety of gradient penalties. The resulting algorithm has the potential for unifying existing approaches for adversarial training. In particular, it helps explain some of the newer approaches to adversarial training (Miyato, Maeda, Koyama, Nakae, & Ishii, 2015; Huang, Xu, Schuurmans, & Szepesvari, 2015). These approaches set up an adversarial objective as a constrained optimization problem and then approximate and simplify it using properties that hold for optimal solutions of unconstrained problems. The algorithms then developed approximate optimization (when compared to our algorithms) and can be viewed as regularizations of this deep gradient.

## 2 The DataGrad Framework

---

Given a set of loss functions  $\mathcal{L}_0, \mathcal{L}_1, \dots, \mathcal{L}_m$  and regularizers  $\mathcal{R}_1, \dots, \mathcal{R}_m$ , consider

$$\begin{aligned} \mathcal{L}_{DG}(t, \mathbf{d}, \Theta) = & \lambda_0 \mathcal{L}_0(t, \mathbf{d}, \Theta) + \lambda_1 \mathcal{R}_1(\mathcal{J}_{\mathcal{L}_1}(t, \mathbf{d}, \Theta)) + \dots \\ & + \lambda_m \mathcal{R}_m(\mathcal{J}_{\mathcal{L}_m}(t, \mathbf{d}, \Theta)), \end{aligned}$$

where  $\mathbf{d} = (d_1, d_2, \dots, d_k)$  is a data sample,  $t$  is its corresponding label/target, and  $\Theta = \{W_1, W_2, \dots, W_K\}$  represents the parameters of a  $K$  layer neural network.<sup>1</sup> We use  $\mathcal{J}_{\mathcal{L}_i}$  to denote the gradient of  $\mathcal{L}_i$  (the gradient of  $\mathcal{L}_i$  with respect to  $\mathbf{d}$ ).  $\lambda_0, \lambda_1, \dots, \lambda_m$  are the weight coefficients of the terms in the DataGrad loss function. Close to our work, Lyu et al. (2015) present a heuristic way to optimize a special case of this objective. By directly providing an algorithm, our analysis can explain what their algorithm optimizes.

We denote the entire data set as  $\mathcal{D} = \{(\mathbf{d}^{(1)}, t^{(1)}), \dots, (\mathbf{d}^{(n)}, t^{(n)})\}$ . Following the framework of empirical risk minimization with stochastic gradient descent, the goal is to minimize the objective function,

---

<sup>1</sup>The loss  $\mathcal{L}_0$  could be a single objective (as in minimizing negative log likelihood for a classification task) or an objective combined with an auxiliary task, as in multitask learning. In the latter setting,  $\mathcal{R}_1$  could be the regularizer for the main task, and  $\mathcal{R}_2$  could be an optional regularizer on the secondary task, especially if the model needs to be robust with respect to the secondary task as well.

$\sum_{i=1}^n \mathcal{L}_{DG}(t^{(i)}, \mathbf{d}^{(i)}, \Theta)$ , by iterating the following parameter updates (here,  $w_{ij}^\ell$  is the component of  $\Theta$  representing the weight of the incoming edge to node  $i$  of layer  $\ell$  from node  $j$  of layer  $\ell - 1$ ):

$$w_{ij}^\ell \leftarrow w_{ij}^\ell - \eta \lambda_0 \frac{\partial}{\partial w_{ij}^{(\ell)}} [\mathcal{L}_0(t, \mathbf{d}, \Theta)] - \eta \sum_{r=1}^m \lambda_r \frac{\partial}{\partial w_{ij}^{(\ell)}} [\mathcal{R}_r(\mathcal{J}_{\mathcal{L}_r}(t, \mathbf{d}, \Theta))], \tag{2.1}$$

where  $\eta$  is the step-size coefficient.

**2.1. The Derivation.** The first update term of equation 1.1,  $\frac{\partial}{\partial w_{ij}^{(\ell)}} [\mathcal{L}_0(t, \mathbf{d}, \Theta)]$ , is provided by standard backpropagation. For the remaining terms, since the gradient of the loss also depends on the current weights  $\Theta$ , we see that

$$\begin{aligned} \frac{\partial \mathcal{R}_r(\mathcal{J}_{\mathcal{L}_r}(t, \mathbf{d}, \Theta))}{\partial w_{ij}^{(\ell)}} &= \frac{\partial \mathcal{R}_r(\frac{\partial \mathcal{L}_r}{\partial a_1}, \dots, \frac{\partial \mathcal{L}_r}{\partial a_k})}{\partial w_{ij}^{(\ell)}} \\ &= \sum_{s=1}^k \frac{\partial \mathcal{R}_r(a_1, \dots, a_k)}{\partial a_s} \frac{\partial^2 \mathcal{L}_r(t, \mathbf{d}, \Theta)}{\partial w_{ij}^{(\ell)} \partial d_s}, \end{aligned} \tag{2.2}$$

where  $a_s$  is a variable that takes the current value of  $\frac{\partial \mathcal{L}_r}{\partial a_k}$ . It turns out that these mixed partial derivatives (with respect to weights and to data) have structural similarities to the Hessian (since derivatives with respect to the data are computed almost exactly the same way as the derivatives with respect to the lowest layer weights). Since exact computation of the Hessian is slow (Bishop, 1992), we would expect that the computation of this matrix of partial derivatives would also be slow. However, it turns out that we do not need to compute the full matrix; we only need this matrix times a vector, and hence we can use ideas reminiscent of fast Hessian multiplication algorithms (Pearlmutter, 1994). At points of continuous differentiability, we have

$$\begin{aligned} \sum_{s=1}^k \frac{\partial \mathcal{R}_r(a_1, \dots, a_k)}{\partial a_s} \frac{\partial^2 \mathcal{L}_r(t, \mathbf{d}, \Theta)}{\partial w_{ij}^{(\ell)} \partial d_s} &= \sum_{s=1}^k \frac{\partial \mathcal{R}_r(a_1, \dots, a_k)}{\partial a_s} \frac{\partial^2 \mathcal{L}_r(t, \mathbf{d}, \Theta)}{\partial d_s \partial w_{ij}^{(\ell)}} \\ &= \frac{\partial^2 \mathcal{L}(t, \mathbf{d} + \phi \mathbf{y}, \Theta)}{\partial \phi \partial w_{ij}^\ell} \end{aligned} \tag{2.3}$$

evaluated at the point  $\phi = 0$  and direction  $\mathbf{y} = \left( \frac{\partial \mathcal{R}_r(a_1, \dots, a_k)}{\partial a_1}, \dots, \frac{\partial \mathcal{R}_r(a_1, \dots, a_k)}{\partial a_k} \right)^2$ . The outer directional derivative with respect to the scalar  $\phi$  can be computed using finite differences. Thus, equations 2.2 and 2.3 mean that we can compute the term  $\frac{\partial}{\partial w_{ij}^{(\ell)}} [\mathcal{R}_r(\mathcal{J}_{\mathcal{L}_r}(t, \mathbf{d}, \Theta))]$  from the stochastic gradient descent update equation, equation 2.1, as follows:

1. Use standard backpropagation to simultaneously compute the vector derivatives  $\frac{\partial \mathcal{L}_r(t, \mathbf{d}, \Theta)}{\partial \Theta}$  and  $\frac{\partial \mathcal{L}_r(t, \mathbf{d}, \Theta)}{\partial \mathbf{d}}$  (note that the latter corresponds to the vector  $(a_1, \dots, a_s)$  in our derivation).
2. Analytically determine the gradient of  $\mathcal{R}_r$  with respect to its immediate inputs. For example, if  $\mathcal{R}_r$  is the  $L_2$  penalty  $\mathcal{R}_r(x_1, \dots, x_s) = |x_1|^2 + \dots + |x_s|^2$ , then the immediate gradient would be  $(2x_1, \dots, 2x_s)$ , and if  $\mathcal{R}_r$  is the  $L_1$  penalty, the immediate gradient would be  $(\text{sign}(x_1), \dots, \text{sign}(x_s))$ .
3. Evaluate the immediate gradient of  $\mathcal{R}_r$  at the vector  $\frac{\partial \mathcal{L}_r(t, \mathbf{d}, \Theta)}{\partial \mathbf{d}}$ . This corresponds to the adversarial direction, denoted by  $\mathbf{y}$  in our derivation.
4. Form the adversarial example  $\hat{\mathbf{d}} = \mathbf{d} + \phi \mathbf{y}$ , where  $\mathbf{y}$  is the result of the previous step and  $\phi$  is a small constant.
5. Use a second backpropagation pass (with  $\hat{\mathbf{d}}$  as input) to compute  $\frac{\partial \mathcal{L}_r(t, \hat{\mathbf{d}}, \Theta)}{\partial \Theta}$ , and then return the finite difference  $\left( \frac{\partial \mathcal{L}_r(t, \hat{\mathbf{d}}, \Theta)}{\partial \Theta} - \frac{\partial \mathcal{L}_r(t, \mathbf{d}, \Theta)}{\partial \Theta} \right) / \phi$ .

**2.2. The High-Level View: Putting It All Together.** At a high level, the loss  $\mathcal{L}_r$  and regularizer  $\mathcal{R}_r$  together serve to define an adversarial noise vector  $\mathbf{y}$  and adversarial example  $\hat{\mathbf{d}} = \mathbf{d} + \phi \mathbf{y}$  (where  $\phi$  is a small constant), as explained in section 2.1. Different choices of  $\mathcal{L}_r$  and  $\mathcal{R}_r$  result in different types of adversarial examples. For example, setting  $\mathcal{R}_r$  to be the  $L_1$  penalty, the resulting adversarial example is the same as that generated by the fast gradient sign method of Goodfellow et al. (2014).

Putting together the components of our finite differences algorithm, the stochastic gradient descent update equation becomes

$$w_{ij}^\ell \leftarrow w_{ij}^\ell - \eta \lambda_0 \frac{\partial \mathcal{L}_0(t, \mathbf{d}, \Theta)}{\partial w_{ij}^{(\ell)}} - \eta \sum_{r=1}^m \frac{\lambda_r}{\phi} \left( \frac{\partial \mathcal{L}_r(t, \mathbf{x}, \Theta)}{\partial w_{ij}^{(\ell)}} \Big|_{\mathbf{x}=\mathbf{x}_r} - \frac{\partial \mathcal{L}_r(t, \mathbf{d}, \Theta)}{\partial w_{ij}^{(\ell)}} \right)$$

<sup>2</sup>Note that equation 2.3 follows from the chain rule.

$$\begin{aligned}
 &= w_{ij}^{(\ell)} - \eta \left( \lambda_0 - \sum_r \frac{\lambda_r}{\phi} \right) \frac{\partial \mathcal{L}_0(t, \mathbf{d}, \Theta)}{\partial w_{ij}^{(\ell)}} \\
 &\quad - \eta \sum_{r=1}^m \frac{\lambda_r}{\phi} \frac{\partial \mathcal{L}_r(t, \mathbf{x}, \Theta)}{\partial w_{ij}^{(\ell)}} \Big|_{\mathbf{x}=\mathbf{x}_r}, \tag{2.4}
 \end{aligned}$$

where  $\mathbf{x}_r$  is the adversarial example of  $\mathbf{d}$  resulting from regularizer  $\mathcal{R}_r$  in conjunction with loss  $\mathcal{L}_r$ , and the notation  $\frac{\partial \mathcal{L}_r(t, \mathbf{x}, \Theta)}{\partial w_{ij}^{(\ell)}} \Big|_{\mathbf{x}=\mathbf{x}_r}$  here specifically means to compute the derivative using backpropagation with  $\mathbf{x}_r$  as an input. In other words,  $\mathbf{x}_r$  is not to be treated as a function of  $\Theta$  (and its components  $w_{ij}^{(\ell)}$ ) when computing this partial derivative.

**2.3. How Prior Works Are Instances of Datagrad.** Since the recent discovery of adversarial samples (Szegedy et al., 2013), a variety of remedies have been proposed to make neural architectures robust to this problem. A straightforward solution is to simply add adversarial examples during each training round of stochastic gradient descent (Szegedy et al., 2013). This is exactly what equation 2.4 specifies, so that a post hoc solution can be justified as a regularization of the data gradient. Subsequent work (Goodfellow et al., 2014) introduced the objective function  $\sum_d \alpha \mathcal{L}(t, \mathbf{d}, \Theta) + (1 - \alpha) \mathcal{L}(t, \hat{\mathbf{d}}, \Theta)$ , where  $\hat{\mathbf{d}}$  is the adversarial version of input  $d$ . A gradient-based method would need to compute the derivative with respect to  $w_{ij}^{(\ell)}$ , which is  $\alpha \frac{\partial \mathcal{L}(t, d, \Theta)}{\partial w_{ij}^{(\ell)}} + (1 - \alpha) \frac{\partial \mathcal{L}(t, \hat{\mathbf{d}}, \Theta)}{\partial w_{ij}^{(\ell)}} + (1 - \alpha) \frac{\partial \mathcal{L}(t, \hat{\mathbf{d}}, \Theta)}{\partial \hat{\mathbf{d}}} \cdot \frac{d\hat{\mathbf{d}}}{dw_{ij}^{(\ell)}}$ , since the construction of  $\hat{\mathbf{d}}$  depends on  $w_{ij}^{(\ell)}$ . Their work approximates the optimization by ignoring the third term, as it is difficult to compute. This approximation then results in an updated equation having the form of equation 2.4 and actually optimizes the DataGrad objective. Nkland (2015) presents a variant where the deep network is trained using backpropagation only on adversarial examples (rather than a mix of adversarial and original examples). Equation 2.4 shows that this method optimizes the DataGrad objective with  $r = 1$  and  $\lambda_0$  and  $\lambda_1$  chosen so that the  $\frac{\partial \mathcal{L}_0(t, \mathbf{d}, \Theta)}{\partial w_{ij}^{(\ell)}}$  term is eliminated.

Both Huang et al. (2015) and Miyato et al. (2015) propose optimizing constrained objective functions that can be put in the form  $\min_{\Theta} \sum_{\mathbf{d}} \max_{g(r) \leq c} f(t, \mathbf{d}, r, \Theta)$ , where  $r$  represents adversarial noise and the constraint  $g(r) \leq c$  puts a bound on the size of the noise. Letting  $r^*(\mathbf{d}, \Theta)$  be the (constrained) optimal value of  $r$  for each  $\mathbf{d}$  and setting of  $\Theta$ , this is the same as the objective  $\min_{\Theta} \sum_{\mathbf{d}} f(t, \mathbf{d}, r^*(\mathbf{d}, \Theta), \Theta)$ . The derivative of any term in the summation with respect to  $w_{ij}^{(\ell)}$  is then equal to

$$\frac{\partial f(t, \mathbf{d}, r, \Theta)}{\partial w_{ij}^{(\ell)}} \Big|_{r=r^*(\mathbf{d}, \Theta)} + \frac{\partial f(t, \mathbf{d}, r, \Theta)}{\partial r} \Big|_{r=r^*(\mathbf{d}, \Theta)} \cdot \frac{\partial r^*(\mathbf{d}, \Theta)}{\partial w_{ij}^{(\ell)}}. \quad (2.5)$$

Now, if  $r^*(\mathbf{d}, \Theta)$  were an unconstrained maximum value of  $r$ , then  $\frac{\partial f(t, \mathbf{d}, r, \Theta)}{\partial r} \Big|_{r=r^*(\mathbf{d}, \Theta)}$  would equal 0, and the second term of equation 2.5 would disappear. However, since  $r^*$  is a constrained optimum and the constraint is active, the second term would generally be nonzero. Since the derivative of the constrained optimum is difficult to compute, Huang et al. (2015) and Miyato et al. (2015) opt to approximate or simplify the derivative, making the second term disappear (as it would in the unconstrained case). Comparing the remaining term to equation 2.4 shows that they are optimizing the DataGrad objective with  $r = 1$  and  $\lambda_0$  and  $\lambda_1$  carefully chosen to eliminate the  $\frac{\partial \mathcal{L}_0(t, \mathbf{d}, \Theta)}{\partial w_{ij}^{(\ell)}}$  term.

In an approach that ends up closely related to ours, Lyu et al. (2015) consider the objective  $\min_{\theta} \max_{r: \|r\|_p \leq \sigma} \mathcal{L}(x + r; \theta)$  and a linearized inner version  $\max_{r: \|r\|_p \leq \sigma} \mathcal{L}(x) + \nabla_x \mathcal{L}^T r$ . They iteratively select  $r$  by optimizing the latter and  $\theta$  by backpropagation on the former (with  $r$  fixed). Since the  $\theta$  update is not directly minimizing the linearized objective, Lyu et al. (2015) claimed the procedure was only an approximation of what we call the DataGrad objective. However, their method devolves to training on adversarial examples, so as before, equation 2.4 shows they are actually optimizing the DataGrad objective but with  $r = 1$  and  $\lambda_0$  and  $\lambda_1$  carefully chosen to eliminate the  $\frac{\partial \mathcal{L}_0(t, \mathbf{d}, \Theta)}{\partial w_{ij}^{(\ell)}}$  term.

Finally, Gu and Rigazio (2014) penalize the Frobenius norm of the deep gradient. However, they do this with a shallow layer-wise approximation. Specifically, they note that shallow contractive autoencoders optimize the same objective for shallow (one-layer) networks and that the gradient of the gradient can be computed analytically in those cases. Thus, they apply this penalty layer by layer (hence, it is a penalty on the derivative of each layer with respect to its immediate inputs) and use this penalty as an approximation to regularizing the deep gradient. Since DataGrad does regularize the deep gradient, the work of Gu and Rigazio (2014) can also be viewed as an approximation to DataGrad.

Thus, DataGrad provides a unifying view of previously proposed optimizations for training deep architectures that are resilient to adversarial noise.

### 3 Experimental Results

---

Given that we have shown that previous approaches are instances of the general DataGrad framework, it is not our intention to replicate prior work. Rather, we intend to test the effectiveness of our finite difference approximation and show that one can flexibly use DataGrad in other scenarios,

such as adding multitask cues within the adversarial framework. To test the proposed DataGrad framework, we conduct experiments using the permutation-invariant MNIST data set<sup>3</sup> of 60,000 training samples and 10,000 testing samples. A validation subset of 10,000 samples (randomly sampled without replacement from the training split) was used for tuning architecture metaparameters via a coarse grid search. Image features were gray-scale pixel values, which we normalized to the range of  $[0, 1]$ . We find that turning our attention first to an image classification problem like MNIST is appropriate since the adversarial problem was first presented in the context of computer vision problems. Investigation of our framework's usefulness in domains such as text is left for future work.

In this study, we experiment with two concrete instantiations of the DataGrad framework: DataGrad-L1 (*DGL1*) and DataGrad-L2 (*DGL2*). By setting  $\lambda_0 = 1$ , letting  $\lambda_1$  freely vary as a metaparameter and  $\lambda_j = 0$  for  $j > 1$ , and  $\mathcal{L}_0 = \mathcal{L}_1$ , choosing  $\mathcal{R}_1$  to be the L1 penalty results in *DGL1* while choosing the L2 penalty yields *DGL2*. As a result, DataGrad becomes a regularization algorithm on either the  $L_1$  or  $L_2$  norm of the gradient of the loss  $\mathcal{L}_0$ . In this setup, DataGrad requires two forward passes and two backward passes to perform a weight update.

We are interested in evaluating how DataGrad compares to conventional and nonconventional forms of regularization. Beyond traditional  $L_1$  and  $L_2$  regularization of the network parameters ( $L1$  and  $L2$ , respectively), we also experimented with the regularizing effect that multitask learning (MT) has on parameter learning in the interest of testing whether having an auxiliary objective could introduce any robustness to adversarial samples in addition to improved generalization. In order to do so, we designed a dual-task rectifier network with two disjoint sets of output units, each connected to the penultimate hidden layer by a separate set of parameters (i.e.,  $U_0$  for task 0,  $U_1$  for task 1). The left-most branch is trained to predict one of the 10 original target digit labels associated with an image (as in the original MNIST task setup), which corresponds to loss  $\mathcal{L}_0$ , while the right-most branch is trained to predict one of five artificially constructed categories pertaining to the discretized degree of rotation of the image, which corresponds to loss  $\mathcal{L}_1$ .<sup>4</sup> The multiobjective optimization problem for this setup then becomes

---

<sup>3</sup><http://yann.lecun.com/exdb/mnist/>.

<sup>4</sup>This auxiliary task consists of essentially predicting whether a given sample has been artificially rotated  $0^\circ$ ,  $15^\circ$  to the left,  $30^\circ$  to the left,  $15^\circ$  to the right, or  $30^\circ$  to the right. To automatically generate these auxiliary labels, we artificially rotate each sample in the training set five times (once for each category) and record that rotation as a label. This could be viewed as a form of data set expansion, which generally leads to improved generalization ability. However, since we assign an artificial label from a different task to accompany each sample, we use the expanded training set to create a different optimization problem.

$$\mathcal{L}_{DG}(t, \mathbf{d}, \Theta) = \mathcal{L}_0(t, \mathbf{d}, \Theta) + \gamma \mathcal{L}_1(t, \mathbf{d}, \Theta) + \lambda_1 \mathcal{R}_1(\mathcal{J}_{\mathcal{L}_0}(t, \mathbf{d}, \Theta)), \quad (3.1)$$

where  $\gamma$  is a coefficient that controls the influence of the auxiliary objective  $\mathcal{L}_1$  on the overall parameter optimization problem. Note that we have extended equation 3.1 to include a DataGrad term, which may be of either L1 form, *MT-DGL1*, or L2 form, *MT-DGL2*. All regularized architectures are compared against the baseline sparse rectifier network, *Rect*.

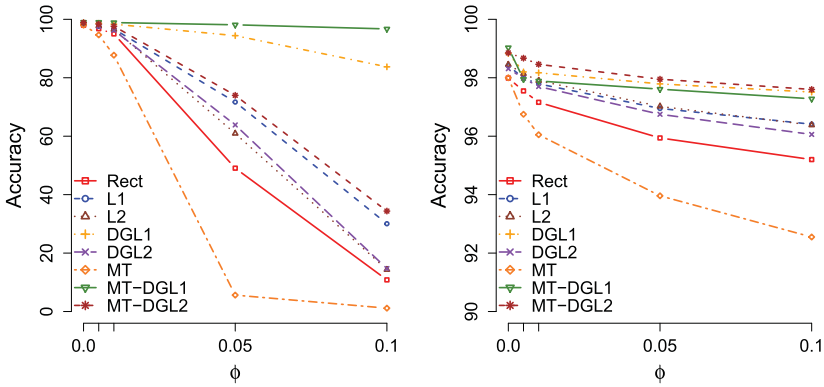
We implemented several deep sparse rectifier architectures (Glorot, Bordes, & Bengio, 2011) three hidden layers, each with 784 latent variables, with parameters were initialized following the scheme of (He, Zhang, Ren, & Sun, 2015), which were all to be trained in a gradient descent framework under the various regularization schemes described earlier. Minibatches of size 100 were used for calculating each parameter update. Hyperparameters and ranges searched included the  $\lambda_1 = [0.0001, 0.1]$  and  $\phi = [0.005, 0.1]$  coefficients for controlling DataGrad, the  $L1 = [0.0001, 0.01]$  and  $L2 = [0.0001, 0.01]$  penalty coefficients (which would simply appear as  $L1 \lambda$  and  $L2 \lambda$ , as in the appendix) for controlling the classical regularization terms, the  $\gamma = [0.25, 0.75]$  auxiliary objective weight, and the gradient descent step-size  $\alpha = [0.001, 0.2]$ . We did not use any additional gradient descent heuristics (e.g., momentum, adaptive learning rates, dropout) for simplicity, since we are interested in investigating the effect that the regularizers have on model robustness to adversarial samples.

To evaluate these architectures in the adversarial setting, we conduct a series of experiments where each trained model plays the role of “attacker.” An adversarial test set of 10,000 samples is generated from the attacking model with backpropagation, using the derivative of the loss with respect to the model’s inputs, followed by the application of the appropriate regularizer function (either  $\mathcal{R}_1$  or  $\mathcal{R}_2$ ) to create the noise. The amount of noise applied is controlled by  $\phi$ , which we varied along the values  $\{0.0, 0.005, 0.01, 0.05, 0.1\}$ , which corresponds to maximal pixel gains of  $\{0, \sim 1, \sim 3, \sim 12, \sim 25\}$  (0 would be equivalent to using the original test set). Generalization performances reported in all figures in this section are of the architectures that achieved best performance on the validation subset (consult the appendix for a full treatment of performance across a range of  $\phi$  and  $\lambda$  values).

We observe in Figures 1, 2, and 3 that a DataGrad-regularized architecture outperforms the nonregularized baseline as well as alternatively regularized ones. Note that the accuracy in Figure 1 drops only to as low as 92% in the worst case, meaning that  $\mathcal{R}_2$  samples seem to cause only minimal damage and should be much less of a concern than  $\mathcal{R}_1$  samples (which would be akin to generating noise via the fast gradient sign method).<sup>5</sup> With

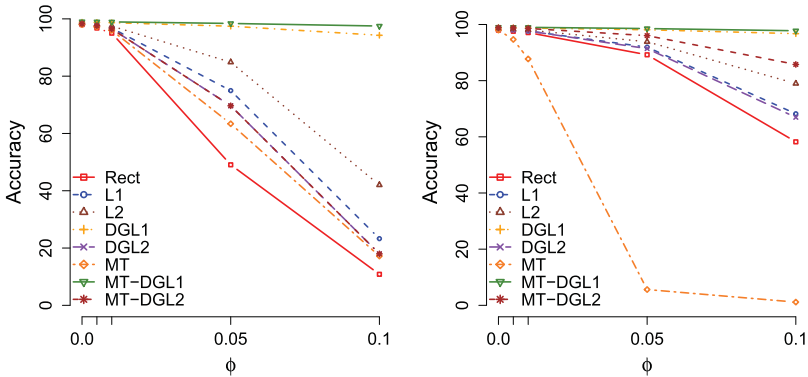
<sup>5</sup>Note that in the case of  $\mathcal{R}_2$  noise, DataGrad L2 yields slightly more robust performance in this scenario, closely followed by DataGrad L1, which makes intuitive sense.





(a) Performance on L1 adversarial samples. (b) Performance on L2 adversarial samples.

Figure 1: Model performance when each model is its own adversary. Note that  $\phi$  on the  $x$ -axis indicates degree of  $(\mathcal{R}_1$  or  $\mathcal{R}_2$ ) noise used to create adversarial samples. Terms in the legend refer to specific architectures (e.g., L2 refers to the L2-regularized network). Note that the model most susceptible to adversarial samples is the one trained under the multitask objective, while, interestingly, the most robust one is the multitask model trained with a DataGrad term.



(a) Simple rectifier network is the adversary. (b) Multi-task rectifier network is the adversary.

Figure 2: Model performance when the adversarial architecture is the same, either simple (*Rect*) or multitask (*MT*). Note that  $\phi$  on the  $x$ -axis indicates degree of  $\mathcal{R}_1$  noise applied to create adversarial samples. We observe that the DataGrad-L1 regularized models, dual or single task, are the most robust to attacks generated by other models of various levels of sophistication (such as a simple, standard rectifier network versus a dual-task rectifier network).
















$\phi =$	<b>0.0</b>	<b>0.005</b>	<b>0.01</b>	<b>0.05</b>	<b>0.1</b>
<i>Rect-MLP</i>	97.99%	96.80%	95.01%	49.06%	10.83%
					
$\phi =$	<b>0.0</b>	<b>0.005</b>	<b>0.01</b>	<b>0.05</b>	<b>0.1</b>
<i>Rect-L1</i>	98.41%	97.62%	96.68%	71.69%	30.05%
					
$\phi =$	<b>0.0</b>	<b>0.005</b>	<b>0.01</b>	<b>0.05</b>	<b>0.1</b>
<i>Rect-DG</i>	98.83%	98.63%	98.34%	94.41%	83.74%
					

Figure 3: Adversarial test set accuracy ( $\phi = 0$  corresponds to original test split) and samples generated from a deep sparse rectifier network in the case of (starting from top of diagram to bottom) (1) no regularization, (2) L1 regularization, and (3) L1 DataGrad regularization. The measures reported here are when each model is used to attack itself (akin to the malicious user using the exact same architecture to generate samples).

respect to using only an auxiliary objective to regularize the model (*MT*), in both Figures 1 and 2, we often see that a dual-task model performs the worst when adversarial noise is introduced, surprisingly even more so than the simple rectifier baseline. Figure 2 shows that when the nonregularized multitask architecture is attacked by itself, its error can drop as low as nearly 1%. However, when a DataGrad term is added to the multitask objective, we achieve nearly no loss in classification performance. This means that a multitask, DataGrad-regularized rectifier network appears to be quite robust to adversarial samples (of either  $\mathcal{R}_1$  or  $\mathcal{R}_2$  form) generated from itself or other perceptron architectures (*DGL1* more so than *DGL2*).

Classical  $L_1$  and  $L_2$  regularizers appear to mitigate some of the damage in some instances, but seemingly afford at best only modest robustness to adversarial perturbation. In contrast, the proposed *DGL1* and *DGL2* regularizers appear to yield a significant reduction in error on all of the various adversarial test sets, the improvement clearer as  $\phi$  is increased (as

evidenced in Figure 3). The visualization of some adversarial samples in Figure 3 demonstrates that even when more noise is applied to generate stronger adversarials, the samples themselves are still quite recognizable to the human eye. However, a neural architecture, such as a deep rectifier network, is sensitive to adversarial noise and incorrectly classifies these images. In addition to robustness against adversarial samples, we also observe improved classification error on the original test set when using DataGrad or multitask DataGrad, the *DGL1* and *MT-DGL1* variants offering the lowest error of all. (For further experimental results exploring the performance and sensitivity of DataGrad to its metaparameters, including when other architectures are the adversary, see the appendix.)

#### 4 Conclusion

---

We have shown how previous proposals can be viewed as instances of a simple, general framework and provide an efficient, deterministic adversarial training procedure, DataGrad. The simplicity of the framework allows easy extensions, such as adding multitask cues as another signal to be combined with adversarial training. Empirically, we found that general DataGrad regularization not only significantly reduces error (especially when combined with a multitask learning objective) in classifying adversarial samples but also improves generalization. We postulate that a reason for this is that adversarial samples generated during the DataGrad learning phase potentially cover more of the underlying data manifold (yielding benefits similar to data set expansion).

Since DataGrad is effectively a deep data-driven penalty, it may be used in tandem with most training objective functions, whether supervised, unsupervised (Bengio, Lamblin, Popovici, & Larochelle, 2007), or hybrid (Ororbia II, Reitter, Wu, & Giles, 2015). Future work entails further improving the efficiency of the proposed DataGrad backpropagation procedure and investigating our procedure in a wider variety of settings.

#### Appendix: Detailed Results

---

In this appendix, to augment the experimental results presented in section 3, we present the generalization performances of the regularized (and non-regularized models) under various settings of their key hyperparameters. This particularly applies to  $\lambda$  and  $\phi$  (when applicable). All model performances reported are those with the learning rate  $\alpha$  and auxiliary objective weight  $\gamma$  metaparameters fixed at the value that yielded the best validation set performance. Furthermore, each table represents a different adversarial scenario, where a different architecture was selected to be the generator of adversarial examples. In each table, two lines are in bold: one for the single-task architecture and one for the multitask model that achieves the most robust performance across all noise values of  $\mathcal{R}_1$ . We do not report

Table 1: Comparative Results Where All Models Are Attacked by a Simple Rectifier Network, *Rect*, Using Laplacian ( $\mathcal{R}_1$ ) Adversarial Noise.

Model	$\phi = 0.0$	$\phi = 0.005$	$\phi = 0.01$	$\phi = 0.05$	$\phi = 0.1$
DGL1 $\lambda = 0.0001$ $\phi = 0.01$	98.25	97.38	96.14	59.74	14.20
DGL1 $\lambda = 0.0001$ $\phi = 0.05$	98.57	98.17	97.76	89.18	59.98
DGL1 $\lambda = 0.0001$ $\phi = 0.1$	98.47	98.13	97.75	90.00	70.03
DGL1 $\lambda = 0.001$ $\phi = 0.01$	98.03	97.26	96.05	62.77	14.82
DGL1 $\lambda = 0.001$ $\phi = 0.05$	98.70	98.43	98.08	92.97	75.65
DGL1 $\lambda = 0.001$ $\phi = 0.1$	98.62	98.38	98.11	93.74	83.06
DGL1 $\lambda = 0.01$ $\phi = 0.01$	98.36	97.96	97.35	88.03	55.16
DGL1 $\lambda = 0.01$ $\phi = 0.05$	98.62	98.50	98.37	95.16	85.78
DGL1 $\lambda = 0.01$ $\phi = 0.1$	<b>98.83</b>	<b>98.77</b>	<b>98.67</b>	<b>97.44</b>	<b>94.27</b>
DGL2 $\lambda = 0.0001$ $\phi = 0.01$	97.93	96.95	95.60	53.29	11.90
DGL2 $\lambda = 0.0001$ $\phi = 0.05$	98.29	97.38	96.06	60.37	13.68
DGL2 $\lambda = 0.0001$ $\phi = 0.1$	98.02	96.97	95.66	53.58	12.26
DGL2 $\lambda = 0.001$ $\phi = 0.01$	97.97	97.00	95.64	54.55	12.07
DGL2 $\lambda = 0.001$ $\phi = 0.05$	98.20	97.08	95.63	55.43	12.53
DGL2 $\lambda = 0.001$ $\phi = 0.1$	98.26	97.42	96.17	60.86	13.18
DGL2 $\lambda = 0.01$ $\phi = 0.01$	98.20	97.49	96.60	72.71	19.84
DGL2 $\lambda = 0.01$ $\phi = 0.05$	98.31	97.60	96.50	69.66	17.97
DGL2 $\lambda = 0.01$ $\phi = 0.1$	98.30	97.54	96.40	68.05	17.61
L1 $\lambda = 0.0001$	98.15	97.48	96.14	59.65	13.19
L1 $\lambda = 0.001$	98.41	97.69	96.77	74.96	23.28
L1 $\lambda = 0.01$	97.73	97.38	97.12	91.08	74.35
L1 $\lambda = 0.1$	93.90	93.38	92.91	86.62	72.01
L2 $\lambda = 0.0001$	98.00	97.05	95.86	57.90	13.93
L2 $\lambda = 0.001$	97.88	96.84	95.39	53.58	12.41
L2 $\lambda = 0.01$	98.45	98.00	97.48	84.87	42.00
L2 $\lambda = 0.1$	98.12	97.85	97.53	91.29	69.87
Rect	97.99	96.80	95.01	49.06	10.83
MT-DGL2 $\lambda = 0.001$ $\phi = 0.01$	98.38	98.00	97.49	81.37	33.88
MT-DGL2 $\lambda = 0.001$ $\phi = 0.05$	98.50	98.06	97.44	81.52	33.55
MT-DGL2 $\lambda = 0.001$ $\phi = 0.1$	98.35	97.87	97.24	79.36	33.47
MT-DGL2 $\lambda = 0.01$ $\phi = 0.01$	98.57	98.26	97.83	89.19	52.96
MT-DGL2 $\lambda = 0.01$ $\phi = 0.05$	98.65	98.26	97.80	89.69	55.09
MT-DGL2 $\lambda = 0.01$ $\phi = 0.1$	98.59	98.31	97.98	90.03	56.43
MT-DGL2 $\lambda = 0.1$ $\phi = 0.01$	98.85	98.74	98.53	95.75	84.42
MT-DGL2 $\lambda = 0.1$ $\phi = 0.05$	98.76	98.59	98.39	95.16	82.86
MT-DGL2 $\lambda = 0.1$ $\phi = 0.1$	98.70	98.54	98.39	95.37	83.70
MT-DGL1 $\lambda = 0.001$ $\phi = 0.01$	98.58	98.38	98.04	90.46	58.06
MT-DGL1 $\lambda = 0.001$ $\phi = 0.05$	98.77	98.66	98.51	96.08	87.14
MT-DGL1 $\lambda = 0.001$ $\phi = 0.1$	98.63	98.47	98.24	93.36	76.71
MT-DGL1 $\lambda = 0.01$ $\phi = 0.01$	98.82	98.69	98.40	95.79	85.41
MT-DGL1 $\lambda = 0.01$ $\phi = 0.05$	98.91	98.82	98.70	97.16	92.72
MT-DGL1 $\lambda = 0.01$ $\phi = 0.1$	98.99	98.94	98.84	97.44	93.79
MT-DGL1 $\lambda = 0.1$ $\phi = 0.01$	98.63	98.50	98.38	97.04	93.79
MT-DGL1 $\lambda = 0.1$ $\phi = 0.05$	98.99	98.97	98.90	98.25	96.68
MT-DGL1 $\lambda = 0.1$ $\phi = 0.1$	<b>99.03</b>	<b>98.98</b>	<b>98.95</b>	<b>98.40</b>	<b>97.50</b>
MT	98.00	97.20	95.99	63.38	17.20

Table 2: Comparative Results Where All Models Are Attacked by the L1-Regularized Rectifier Network,  $L1$ , Using Laplacian ( $\mathcal{R}_1$ ) Adversarial Noise.

Model	$\phi = 0.0$	$\phi = 0.005$	$\phi = 0.01$	$\phi = 0.05$	$\phi = 0.1$
<i>DGL1</i> $\lambda = 0.0001$ $\phi = 0.01$	98.25	97.31	95.96	56.68	23.68
<i>DGL1</i> $\lambda = 0.0001$ $\phi = 0.05$	98.57	98.15	97.68	88.75	58.32
<i>DGL1</i> $\lambda = 0.0001$ $\phi = 0.1$	98.47	98.11	97.75	89.65	68.44
<i>DGL1</i> $\lambda = 0.001$ $\phi = 0.01$	98.03	97.27	96.12	65.06	26.64
<i>DGL1</i> $\lambda = 0.001$ $\phi = 0.05$	98.70	98.41	98.06	92.78	74.48
<i>DGL1</i> $\lambda = 0.001$ $\phi = 0.1$	98.62	98.37	98.12	93.52	82.12
<i>DGL1</i> $\lambda = 0.01$ $\phi = 0.01$	98.36	97.93	97.34	87.04	54.26
<i>DGL1</i> $\lambda = 0.01$ $\phi = 0.05$	98.62	98.51	98.36	95.00	85.33
<i>DGL1</i> $\lambda = 0.01$ $\phi = 0.1$	<b>98.83</b>	<b>98.77</b>	<b>98.68</b>	<b>97.47</b>	<b>94.20</b>
<i>DGL2</i> $\lambda = 0.0001$ $\phi = 0.01$	97.93	97.00	95.72	56.55	23.07
<i>DGL2</i> $\lambda = 0.0001$ $\phi = 0.05$	98.29	97.32	95.94	57.76	23.86
<i>DGL2</i> $\lambda = 0.0001$ $\phi = 0.1$	98.02	97.03	95.74	57.57	23.96
<i>DGL2</i> $\lambda = 0.001$ $\phi = 0.01$	97.97	96.98	95.70	57.95	23.37
<i>DGL2</i> $\lambda = 0.001$ $\phi = 0.05$	98.20	97.14	95.71	58.54	24.01
<i>DGL2</i> $\lambda = 0.001$ $\phi = 0.1$	98.26	97.36	96.08	57.30	23.43
<i>DGL2</i> $\lambda = 0.01$ $\phi = 0.01$	98.20	97.43	96.48	69.58	28.98
<i>DGL2</i> $\lambda = 0.01$ $\phi = 0.05$	98.31	97.57	96.41	66.05	26.99
<i>DGL2</i> $\lambda = 0.01$ $\phi = 0.1$	98.30	97.51	96.32	65.38	26.63
<i>L1</i> $\lambda = 0.0001$	98.15	97.24	95.61	51.91	21.44
<i>L1</i> $\lambda = 0.001$	98.41	97.62	96.68	71.69	30.05
<i>L1</i> $\lambda = 0.01$	97.73	97.39	97.08	91.00	73.68
<i>L1</i> $\lambda = 0.1$	93.90	93.38	92.90	86.96	73.77
<i>L2</i> $\lambda = 0.0001$	98.00	97.03	95.72	56.20	24.11
<i>L2</i> $\lambda = 0.001$	97.88	96.89	95.50	57.57	23.54
<i>L2</i> $\lambda = 0.01$	98.45	97.98	97.43	83.01	42.66
<i>L2</i> $\lambda = 0.1$	98.12	97.83	97.52	90.66	68.11
<i>Rect</i>	97.99	96.98	95.69	58.53	23.58
<i>MT-DGL2</i> $\lambda = 0.001$ $\phi = 0.01$	98.38	97.99	97.47	80.50	39.72
<i>MT-DGL2</i> $\lambda = 0.001$ $\phi = 0.05$	98.50	98.06	97.44	80.28	39.47
<i>MT-DGL2</i> $\lambda = 0.001$ $\phi = 0.1$	98.35	97.89	97.27	78.97	39.08
<i>MT-DGL2</i> $\lambda = 0.01$ $\phi = 0.01$	98.57	98.28	97.84	88.89	55.29
<i>MT-DGL2</i> $\lambda = 0.01$ $\phi = 0.05$	98.65	98.23	97.81	89.46	56.76
<i>MT-DGL2</i> $\lambda = 0.01$ $\phi = 0.1$	98.59	98.35	97.98	89.58	58.33
<i>MT-DGL2</i> $\lambda = 0.1$ $\phi = 0.01$	98.85	98.74	98.57	95.67	84.08
<i>MT-DGL2</i> $\lambda = 0.1$ $\phi = 0.05$	98.76	98.57	98.36	95.12	81.97
<i>MT-DGL2</i> $\lambda = 0.1$ $\phi = 0.1$	98.70	98.55	98.38	95.30	82.72
<i>MT-DGL1</i> $\lambda = 0.001$ $\phi = 0.01$	98.58	98.36	98.06	90.06	59.26
<i>MT-DGL1</i> $\lambda = 0.001$ $\phi = 0.05$	98.77	98.64	98.48	95.92	86.91
<i>MT-DGL1</i> $\lambda = 0.001$ $\phi = 0.1$	98.63	98.47	98.27	93.21	76.36
<i>MT-DGL1</i> $\lambda = 0.01$ $\phi = 0.01$	98.82	98.66	98.40	95.73	85.10
<i>MT-DGL1</i> $\lambda = 0.01$ $\phi = 0.05$	98.91	98.79	98.70	97.18	92.77
<i>MT-DGL1</i> $\lambda = 0.01$ $\phi = 0.1$	98.99	98.94	98.78	97.45	93.90
<i>MT-DGL1</i> $\lambda = 0.1$ $\phi = 0.01$	98.63	98.51	98.40	96.99	93.65
<i>MT-DGL1</i> $\lambda = 0.1$ $\phi = 0.05$	98.99	98.96	98.89	98.26	96.68
<i>MT-DGL1</i> $\lambda = 0.1$ $\phi = 0.1$	<b>99.03</b>	<b>98.97</b>	<b>98.93</b>	<b>98.37</b>	<b>97.33</b>
<i>MT</i>	98.00	97.18	95.86	63.03	25.51

Table 3: Comparative Results Where All Models Are Attacked by the L2-Regularized Rectifier Network, L2, Using Laplacian ( $\mathcal{R}_1$ ) Adversarial Noise.

Model	$\phi = 0.0$	$\phi = 0.005$	$\phi = 0.01$	$\phi = 0.05$	$\phi = 0.1$
DGL1 $\lambda = 0.0001 \phi = 0.01$	98.25	97.70	96.81	75.66	25.26
DGL1 $\lambda = 0.0001 \phi = 0.05$	98.57	98.28	97.99	93.04	73.97
DGL1 $\lambda = 0.0001 \phi = 0.1$	98.47	98.26	97.92	93.39	77.97
DGL1 $\lambda = 0.001 \phi = 0.01$	98.03	97.48	96.94	81.26	32.74
DGL1 $\lambda = 0.001 \phi = 0.05$	98.70	98.48	98.22	94.82	82.96
DGL1 $\lambda = 0.001 \phi = 0.1$	98.62	98.44	98.27	95.04	86.12
DGL1 $\lambda = 0.01 \phi = 0.01$	98.36	98.04	97.52	90.49	63.92
DGL1 $\lambda = 0.01 \phi = 0.05$	98.62	98.51	98.42	95.57	87.67
DGL1 $\lambda = 0.01 \phi = 0.1$	<b>98.83</b>	<b>98.77</b>	<b>98.67</b>	<b>97.44</b>	<b>94.27</b>
DGL2 $\lambda = 0.0001 \phi = 0.01$	97.93	97.28	96.50	76.93	25.90
DGL2 $\lambda = 0.0001 \phi = 0.05$	98.29	97.68	96.75	75.97	24.43
DGL2 $\lambda = 0.0001 \phi = 0.1$	98.02	97.35	96.57	76.87	26.25
DGL2 $\lambda = 0.001 \phi = 0.01$	97.97	97.38	96.46	77.19	26.88
DGL2 $\lambda = 0.001 \phi = 0.05$	98.20	97.52	96.66	77.90	26.53
DGL2 $\lambda = 0.001 \phi = 0.1$	98.26	97.72	96.89	77.18	25.03
DGL2 $\lambda = 0.01 \phi = 0.01$	98.20	97.76	97.02	83.13	36.57
DGL2 $\lambda = 0.01 \phi = 0.05$	98.31	97.78	97.14	82.35	34.70
DGL2 $\lambda = 0.01 \phi = 0.1$	98.30	97.77	97.17	82.02	33.80
L1 $\lambda = 0.0001$	98.15	97.63	96.91	75.97	24.07
L1 $\lambda = 0.001$	98.41	97.71	96.97	76.47	24.02
L1 $\lambda = 0.01$	97.73	97.32	96.97	89.07	64.45
L1 $\lambda = 0.1$	93.90	93.40	92.83	86.17	68.98
L2 $\lambda = 0.0001$	98.00	97.37	96.59	76.52	26.14
L2 $\lambda = 0.001$	97.88	97.22	96.44	76.19	25.73
L2 $\lambda = 0.01$	98.45	97.74	96.52	60.93	14.34
L2 $\lambda = 0.1$	98.12	97.81	97.19	86.46	47.90
Rect	97.99	97.32	96.57	77.87	26.99
MT-DGL2 $\lambda = 0.001 \phi = 0.01$	98.38	98.07	97.63	86.65	45.44
MT-DGL2 $\lambda = 0.001 \phi = 0.05$	98.50	98.15	97.69	86.91	44.76
MT-DGL2 $\lambda = 0.001 \phi = 0.1$	98.35	97.99	97.58	86.04	43.86
MT-DGL2 $\lambda = 0.01 \phi = 0.01$	98.57	98.33	97.96	91.69	63.27
MT-DGL2 $\lambda = 0.01 \phi = 0.05$	98.65	98.33	98.01	92.24	65.58
MT-DGL2 $\lambda = 0.01 \phi = 0.1$	98.59	98.41	98.10	92.37	66.75
MT-DGL2 $\lambda = 0.1 \phi = 0.01$	98.85	98.76	98.59	96.03	85.26
MT-DGL2 $\lambda = 0.1 \phi = 0.05$	98.76	98.64	98.43	95.71	84.42
MT-DGL2 $\lambda = 0.1 \phi = 0.1$	98.70	98.56	98.42	95.78	85.20
MT-DGL1 $\lambda = 0.001 \phi = 0.01$	98.58	98.41	98.16	92.21	65.98
MT-DGL1 $\lambda = 0.001 \phi = 0.05$	98.77	98.65	98.53	96.28	88.39
MT-DGL1 $\lambda = 0.001 \phi = 0.1$	98.63	98.48	98.33	94.49	79.56
MT-DGL1 $\lambda = 0.01 \phi = 0.01$	98.82	98.69	98.45	96.13	86.22
MT-DGL1 $\lambda = 0.01 \phi = 0.05$	98.91	98.82	98.71	97.26	92.70
MT-DGL1 $\lambda = 0.01 \phi = 0.1$	98.99	98.94	98.81	97.43	94.04
MT-DGL1 $\lambda = 0.1 \phi = 0.01$	98.63	98.51	98.41	96.98	93.27
MT-DGL1 $\lambda = 0.1 \phi = 0.05$	98.99	98.96	98.90	98.24	96.58
MT-DGL1 $\lambda = 0.1 \phi = 0.1$	<b>99.03</b>	<b>98.98</b>	<b>98.91</b>	<b>98.36</b>	<b>97.37</b>
MT	98.00	97.37	96.59	74.09	23.44

Table 4: Comparative Results Where All Models Are Attacked by the DataGrad-L1-Regularized Rectifier Network, *DGL1*, Using Laplacian ( $\mathcal{R}_1$ ) Adversarial Noise.

Model	$\phi = 0.0$	$\phi = 0.005$	$\phi = 0.01$	$\phi = 0.05$	$\phi = 0.1$
<i>DGL1</i> $\lambda = 0.0001$ $\phi = 0.01$	98.25	98.05	97.71	93.34	78.95
<i>DGL1</i> $\lambda = 0.0001$ $\phi = 0.05$	98.57	98.31	98.14	94.85	83.99
<i>DGL1</i> $\lambda = 0.0001$ $\phi = 0.1$	98.47	98.27	98.05	94.89	84.31
<i>DGL1</i> $\lambda = 0.001$ $\phi = 0.01$	98.03	97.75	97.42	92.87	78.07
<i>DGL1</i> $\lambda = 0.001$ $\phi = 0.05$	98.70	98.47	98.21	94.83	84.16
<i>DGL1</i> $\lambda = 0.001$ $\phi = 0.1$	<b>98.62</b>	<b>98.42</b>	<b>98.21</b>	<b>94.73</b>	<b>84.87</b>
<i>DGL1</i> $\lambda = 0.01$ $\phi = 0.01$	98.36	98.15	97.83	93.55	80.92
<i>DGL1</i> $\lambda = 0.01$ $\phi = 0.05$	98.62	98.49	98.33	94.34	82.77
<i>DGL1</i> $\lambda = 0.01$ $\phi = 0.1$	98.83	98.63	98.34	94.41	83.74
<i>DGL2</i> $\lambda = 0.0001$ $\phi = 0.01$	97.93	97.71	97.31	92.55	77.17
<i>DGL2</i> $\lambda = 0.0001$ $\phi = 0.05$	98.29	97.99	97.72	93.20	78.77
<i>DGL2</i> $\lambda = 0.0001$ $\phi = 0.1$	98.02	97.73	97.39	92.77	77.69
<i>DGL2</i> $\lambda = 0.001$ $\phi = 0.01$	97.97	97.64	97.37	92.60	76.74
<i>DGL2</i> $\lambda = 0.001$ $\phi = 0.05$	98.20	97.95	97.52	92.97	77.97
<i>DGL2</i> $\lambda = 0.001$ $\phi = 0.1$	98.26	98.00	97.73	93.44	78.81
<i>DGL2</i> $\lambda = 0.01$ $\phi = 0.01$	98.20	97.95	97.65	93.26	79.59
<i>DGL2</i> $\lambda = 0.01$ $\phi = 0.05$	98.31	98.07	97.69	93.14	78.58
<i>DGL2</i> $\lambda = 0.01$ $\phi = 0.1$	98.30	98.02	97.69	92.98	77.97
<i>L1</i> $\lambda = 0.0001$	98.15	97.89	97.61	93.22	78.80
<i>L1</i> $\lambda = 0.001$	98.41	98.10	97.83	94.07	81.43
<i>L1</i> $\lambda = 0.01$	97.73	97.46	97.22	93.56	83.95
<i>L1</i> $\lambda = 0.1$	93.90	93.56	93.29	89.81	81.29
<i>L2</i> $\lambda = 0.0001$	98.00	97.76	97.41	92.57	77.57
<i>L2</i> $\lambda = 0.001$	97.88	97.59	97.28	92.37	76.68
<i>L2</i> $\lambda = 0.01$	98.45	98.24	98.03	94.26	82.19
<i>L2</i> $\lambda = 0.1$	98.12	97.96	97.75	94.44	84.35
<i>Rect</i>	97.99	97.68	97.34	92.89	78.44
<i>MT-DGL2</i> $\lambda = 0.001$ $\phi = 0.01$	98.38	98.15	97.94	93.69	79.16
<i>MT-DGL2</i> $\lambda = 0.001$ $\phi = 0.05$	98.50	98.24	98.00	93.77	79.74
<i>MT-DGL2</i> $\lambda = 0.001$ $\phi = 0.1$	98.35	98.10	97.87	93.41	77.41
<i>MT-DGL2</i> $\lambda = 0.01$ $\phi = 0.01$	98.57	98.38	98.16	94.37	81.79
<i>MT-DGL2</i> $\lambda = 0.01$ $\phi = 0.05$	98.65	98.40	98.13	94.53	81.87
<i>MT-DGL2</i> $\lambda = 0.01$ $\phi = 0.1$	98.59	98.45	98.19	94.50	82.22
<i>MT-DGL2</i> $\lambda = 0.1$ $\phi = 0.01$	98.85	98.75	98.58	96.10	86.59
<i>MT-DGL2</i> $\lambda = 0.1$ $\phi = 0.05$	98.76	98.58	98.35	95.58	85.48
<i>MT-DGL2</i> $\lambda = 0.1$ $\phi = 0.1$	98.70	98.54	98.42	95.72	85.46
<i>MT-DGL1</i> $\lambda = 0.001$ $\phi = 0.01$	98.58	98.50	98.25	94.80	83.03
<i>MT-DGL1</i> $\lambda = 0.001$ $\phi = 0.05$	98.77	98.64	98.47	95.99	88.08
<i>MT-DGL1</i> $\lambda = 0.001$ $\phi = 0.1$	98.63	98.48	98.32	95.33	85.11
<i>MT-DGL1</i> $\lambda = 0.01$ $\phi = 0.01$	98.82	98.66	98.43	95.68	86.38
<i>MT-DGL1</i> $\lambda = 0.01$ $\phi = 0.05$	98.91	98.79	98.59	96.61	90.05
<i>MT-DGL1</i> $\lambda = 0.01$ $\phi = 0.1$	98.99	98.91	98.77	96.67	90.82
<i>MT-DGL1</i> $\lambda = 0.1$ $\phi = 0.01$	98.63	98.50	98.32	96.43	91.62
<i>MT-DGL1</i> $\lambda = 0.1$ $\phi = 0.05$	98.99	98.92	98.86	97.90	95.35
<i>MT-DGL1</i> $\lambda = 0.1$ $\phi = 0.1$	<b>99.03</b>	<b>98.98</b>	<b>98.87</b>	<b>98.08</b>	<b>96.42</b>
<i>MT</i>	98.00	97.65	97.30	92.13	75.10

Table 5: Comparative Results Where All Models Are Attacked by the DataGrad-L2-Regularized Rectifier Network, *DGL2*, Using Laplacian ( $\mathcal{R}_1$ ) Adversarial Noise.

Model	$\phi = 0.0$	$\phi = 0.005$	$\phi = 0.01$	$\phi = 0.05$	$\phi = 0.1$
<i>DGL1</i> $\lambda = 0.0001$ $\phi = 0.01$	98.25	97.41	96.16	60.96	14.33
<i>DGL1</i> $\lambda = 0.0001$ $\phi = 0.05$	98.57	98.13	97.67	88.27	54.88
<i>DGL1</i> $\lambda = 0.0001$ $\phi = 0.1$	98.47	98.12	97.66	89.06	65.93
<i>DGL1</i> $\lambda = 0.001$ $\phi = 0.01$	98.03	97.28	96.12	67.04	16.81
<i>DGL1</i> $\lambda = 0.001$ $\phi = 0.05$	98.70	98.40	97.95	92.00	70.31
<i>DGL1</i> $\lambda = 0.001$ $\phi = 0.1$	98.62	98.36	98.07	92.92	79.93
<i>DGL1</i> $\lambda = 0.01$ $\phi = 0.01$	98.36	97.91	97.28	86.01	48.12
<i>DGL1</i> $\lambda = 0.01$ $\phi = 0.05$	98.62	98.50	98.35	94.81	83.50
<i>DGL1</i> $\lambda = 0.01$ $\phi = 0.1$	<b>98.83</b>	<b>98.77</b>	<b>98.67</b>	<b>97.28</b>	<b>93.75</b>
<i>DGL2</i> $\lambda = 0.0001$ $\phi = 0.01$	97.93	97.05	95.73	59.03	13.69
<i>DGL2</i> $\lambda = 0.0001$ $\phi = 0.05$	98.29	97.38	96.06	60.67	14.03
<i>DGL2</i> $\lambda = 0.0001$ $\phi = 0.1$	98.02	97.11	95.77	60.25	14.48
<i>DGL2</i> $\lambda = 0.001$ $\phi = 0.01$	97.97	97.04	95.85	60.31	14.08
<i>DGL2</i> $\lambda = 0.001$ $\phi = 0.05$	98.20	97.17	95.90	61.10	14.52
<i>DGL2</i> $\lambda = 0.001$ $\phi = 0.1$	98.26	97.43	96.16	60.24	13.34
<i>DGL2</i> $\lambda = 0.01$ $\phi = 0.01$	98.20	97.41	96.43	69.33	17.98
<i>DGL2</i> $\lambda = 0.01$ $\phi = 0.05$	98.31	97.55	96.26	63.85	14.71
<i>DGL2</i> $\lambda = 0.01$ $\phi = 0.1$	98.30	97.34	95.85	57.83	13.00
<i>L1</i> $\lambda = 0.0001$	98.15	97.48	96.18	61.12	13.89
<i>L1</i> $\lambda = 0.001$	98.41	97.66	96.82	75.51	23.36
<i>L1</i> $\lambda = 0.01$	97.73	97.34	97.10	91.06	73.55
<i>L1</i> $\lambda = 0.1$	93.90	93.38	92.94	86.84	72.99
<i>L2</i> $\lambda = 0.0001$	98.00	97.05	95.82	59.47	14.10
<i>L2</i> $\lambda = 0.001$	97.88	96.93	95.69	59.73	14.67
<i>L2</i> $\lambda = 0.01$	98.45	98.02	97.51	84.91	42.24
<i>L2</i> $\lambda = 0.1$	98.12	97.84	97.53	91.02	68.14
<i>Rect</i>	97.99	97.04	95.78	61.98	15.02
<i>MT-DGL2</i> $\lambda = 0.001$ $\phi = 0.01$	98.38	97.98	97.42	80.81	32.25
<i>MT-DGL2</i> $\lambda = 0.001$ $\phi = 0.05$	98.50	98.07	97.41	80.76	32.50
<i>MT-DGL2</i> $\lambda = 0.001$ $\phi = 0.1$	98.35	97.90	97.30	79.30	32.06
<i>MT-DGL2</i> $\lambda = 0.01$ $\phi = 0.01$	98.57	98.26	97.83	88.36	50.17
<i>MT-DGL2</i> $\lambda = 0.01$ $\phi = 0.05$	98.65	98.24	97.77	88.79	51.86
<i>MT-DGL2</i> $\lambda = 0.01$ $\phi = 0.1$	98.59	98.31	97.93	89.13	53.25
<i>MT-DGL2</i> $\lambda = 0.1$ $\phi = 0.01$	98.85	98.76	98.53	95.51	82.51
<i>MT-DGL2</i> $\lambda = 0.1$ $\phi = 0.05$	98.76	98.58	98.33	94.74	80.04
<i>MT-DGL2</i> $\lambda = 0.1$ $\phi = 0.1$	98.70	98.52	98.34	95.08	81.28
<i>MT-DGL1</i> $\lambda = 0.001$ $\phi = 0.01$	98.58	98.36	98.06	89.82	55.87
<i>MT-DGL1</i> $\lambda = 0.001$ $\phi = 0.05$	98.77	98.65	98.49	95.86	85.98
<i>MT-DGL1</i> $\lambda = 0.001$ $\phi = 0.1$	98.63	98.46	98.24	92.57	74.01
<i>MT-DGL1</i> $\lambda = 0.01$ $\phi = 0.01$	98.82	98.65	98.41	95.46	83.91
<i>MT-DGL1</i> $\lambda = 0.01$ $\phi = 0.05$	98.91	98.79	98.66	96.99	92.26
<i>MT-DGL1</i> $\lambda = 0.01$ $\phi = 0.1$	98.99	98.94	98.80	97.18	93.14
<i>MT-DGL1</i> $\lambda = 0.1$ $\phi = 0.01$	98.63	98.50	98.37	97.04	93.33
<i>MT-DGL1</i> $\lambda = 0.1$ $\phi = 0.05$	98.99	98.95	98.89	98.19	96.31
<i>MT-DGL1</i> $\lambda = 0.1$ $\phi = 0.1$	<b>99.03</b>	<b>98.98</b>	<b>98.94</b>	<b>98.31</b>	<b>97.36</b>
<i>MT</i>	98.00	97.22	96.04	64.16	17.38



Table 6: Comparative Results Where All Models Are Attacked by the Multitask Rectifier Network, *MT*, Using Laplacian ( $\mathcal{R}_1$ ) Adversarial Noise Driven by the Target Task (Digit Recognition).

Model	$\phi = 0.0$	$\phi = 0.005$	$\phi = 0.01$	$\phi = 0.05$	$\phi = 0.1$
<i>DGL1</i> $\lambda = 0.0001$ $\phi = 0.01$	98.25	97.95	97.50	89.19	57.88
<i>DGL1</i> $\lambda = 0.0001$ $\phi = 0.05$	98.57	98.38	98.16	95.99	89.33
<i>DGL1</i> $\lambda = 0.0001$ $\phi = 0.1$	98.47	98.28	98.19	96.15	89.95
<i>DGL1</i> $\lambda = 0.001$ $\phi = 0.01$	98.03	97.71	97.36	90.86	65.55
<i>DGL1</i> $\lambda = 0.001$ $\phi = 0.05$	98.70	98.56	98.41	96.74	92.11
<i>DGL1</i> $\lambda = 0.001$ $\phi = 0.1$	98.62	98.52	98.39	96.94	93.02
<i>DGL1</i> $\lambda = 0.01$ $\phi = 0.01$	98.36	98.18	97.99	95.13	86.87
<i>DGL1</i> $\lambda = 0.01$ $\phi = 0.05$	98.62	98.54	98.49	97.09	94.23
<i>DGL1</i> $\lambda = 0.01$ $\phi = 0.1$	<b>98.83</b>	<b>98.80</b>	<b>98.75</b>	<b>98.12</b>	<b>96.73</b>
<i>DGL2</i> $\lambda = 0.0001$ $\phi = 0.01$	97.93	97.50	97.13	88.86	56.48
<i>DGL2</i> $\lambda = 0.0001$ $\phi = 0.05$	98.29	97.91	97.51	89.36	57.68
<i>DGL2</i> $\lambda = 0.0001$ $\phi = 0.1$	98.02	97.59	97.16	88.85	56.84
<i>DGL2</i> $\lambda = 0.001$ $\phi = 0.01$	97.97	97.52	97.17	89.20	58.11
<i>DGL2</i> $\lambda = 0.001$ $\phi = 0.05$	98.20	97.88	97.28	89.41	58.29
<i>DGL2</i> $\lambda = 0.001$ $\phi = 0.1$	98.26	97.90	97.43	89.71	58.21
<i>DGL2</i> $\lambda = 0.01$ $\phi = 0.01$	98.20	97.92	97.55	91.95	70.65
<i>DGL2</i> $\lambda = 0.01$ $\phi = 0.05$	98.31	98.02	97.69	91.41	66.94
<i>DGL2</i> $\lambda = 0.01$ $\phi = 0.1$	98.30	98.00	97.58	91.21	65.30
<i>L1</i> $\lambda = 0.0001$	98.15	97.84	97.44	89.63	58.24
<i>L1</i> $\lambda = 0.001$	98.41	98.12	97.68	91.93	68.21
<i>L1</i> $\lambda = 0.01$	97.73	97.54	97.37	95.01	89.36
<i>L1</i> $\lambda = 0.1$	93.90	93.61	93.40	90.66	85.01
<i>L2</i> $\lambda = 0.0001$	98.00	97.62	97.16	88.81	56.66
<i>L2</i> $\lambda = 0.001$	97.88	97.49	97.09	88.85	57.26
<i>L2</i> $\lambda = 0.01$	98.45	98.22	98.02	93.85	78.97
<i>L2</i> $\lambda = 0.1$	98.12	98.02	97.84	95.29	88.35
<i>Rect</i>	97.99	97.59	97.12	89.21	58.22
<i>MT-DGL2</i> $\lambda = 0.001$ $\phi = 0.01$	98.38	97.75	96.73	58.65	10.89
<i>MT-DGL2</i> $\lambda = 0.001$ $\phi = 0.05$	98.50	97.73	96.75	58.44	10.43
<i>MT-DGL2</i> $\lambda = 0.001$ $\phi = 0.1$	98.35	97.66	96.42	56.42	10.20
<i>MT-DGL2</i> $\lambda = 0.01$ $\phi = 0.01$	98.57	98.11	97.62	81.13	29.40
<i>MT-DGL2</i> $\lambda = 0.01$ $\phi = 0.05$	98.65	98.16	97.62	83.11	31.86
<i>MT-DGL2</i> $\lambda = 0.01$ $\phi = 0.1$	98.59	98.25	97.73	83.52	33.65
<i>MT-DGL2</i> $\lambda = 0.1$ $\phi = 0.01$	98.85	98.77	98.59	96.03	85.77
<i>MT-DGL2</i> $\lambda = 0.1$ $\phi = 0.05$	98.76	98.57	98.37	95.17	81.90
<i>MT-DGL2</i> $\lambda = 0.1$ $\phi = 0.1$	98.70	98.53	98.39	95.22	83.77
<i>MT-DGL1</i> $\lambda = 0.001$ $\phi = 0.01$	98.58	98.32	97.93	87.90	46.80
<i>MT-DGL1</i> $\lambda = 0.001$ $\phi = 0.05$	98.77	98.67	98.50	96.32	89.44
<i>MT-DGL1</i> $\lambda = 0.001$ $\phi = 0.1$	98.63	98.48	98.23	93.22	76.42
<i>MT-DGL1</i> $\lambda = 0.01$ $\phi = 0.01$	98.82	98.71	98.47	96.00	86.98
<i>MT-DGL1</i> $\lambda = 0.01$ $\phi = 0.05$	98.91	98.82	98.73	97.60	94.22
<i>MT-DGL1</i> $\lambda = 0.01$ $\phi = 0.1$	98.99	98.95	98.84	97.84	95.56
<i>MT-DGL1</i> $\lambda = 0.1$ $\phi = 0.01$	98.63	98.54	98.46	97.42	95.18
<i>MT-DGL1</i> $\lambda = 0.1$ $\phi = 0.05$	98.99	98.96	98.92	98.42	97.39
<i>MT-DGL1</i> $\lambda = 0.1$ $\phi = 0.1$	<b>99.03</b>	<b>99.00</b>	<b>98.98</b>	<b>98.55</b>	<b>97.77</b>
<i>MT</i>	98.00	94.67	87.73	5.62	1.15

Table 7: Comparative Results Where All Models Are Attacked by the DataGrad-L1 Regularized Multitask Rectifier Network, *MT-DGL1*, Using Laplacian ( $\mathcal{R}_1$ ) Adversarial Noise Driven by the Target Task (Digit Recognition).

Model	$\phi = 0.0$	$\phi = 0.005$	$\phi = 0.01$	$\phi = 0.05$	$\phi = 0.1$
<i>DGL1</i> $\lambda = 0.0001$ $\phi = 0.01$	98.25	98.10	97.94	95.63	89.74
<i>DGL1</i> $\lambda = 0.0001$ $\phi = 0.05$	98.57	98.43	98.27	96.89	92.92
<i>DGL1</i> $\lambda = 0.0001$ $\phi = 0.1$	98.47	98.33	98.24	96.70	92.79
<i>DGL1</i> $\lambda = 0.001$ $\phi = 0.01$	98.03	97.91	97.66	95.47	89.90
<i>DGL1</i> $\lambda = 0.001$ $\phi = 0.05$	98.70	98.57	98.42	97.01	93.27
<i>DGL1</i> $\lambda = 0.001$ $\phi = 0.1$	98.62	98.50	98.41	96.93	93.22
<i>DGL1</i> $\lambda = 0.01$ $\phi = 0.01$	98.36	98.20	98.07	96.01	91.09
<i>DGL1</i> $\lambda = 0.01$ $\phi = 0.05$	98.62	98.52	98.49	96.88	93.56
<i>DGL1</i> $\lambda = 0.01$ $\phi = 0.1$	<b>98.83</b>	<b>98.77</b>	<b>98.68</b>	<b>97.72</b>	<b>95.26</b>
<i>DGL2</i> $\lambda = 0.0001$ $\phi = 0.01$	97.93	97.79	97.57	95.33	89.18
<i>DGL2</i> $\lambda = 0.0001$ $\phi = 0.05$	98.29	98.15	97.91	95.69	90.02
<i>DGL2</i> $\lambda = 0.0001$ $\phi = 0.1$	98.02	97.85	97.67	95.22	89.50
<i>DGL2</i> $\lambda = 0.001$ $\phi = 0.01$	97.97	97.73	97.52	95.12	89.20
<i>DGL2</i> $\lambda = 0.001$ $\phi = 0.05$	98.20	98.06	97.78	95.31	89.61
<i>DGL2</i> $\lambda = 0.001$ $\phi = 0.1$	98.26	98.11	98.00	95.79	90.29
<i>DGL2</i> $\lambda = 0.01$ $\phi = 0.01$	98.20	98.02	97.84	95.81	90.29
<i>DGL2</i> $\lambda = 0.01$ $\phi = 0.05$	98.31	98.14	97.99	95.82	90.24
<i>DGL2</i> $\lambda = 0.01$ $\phi = 0.1$	98.30	98.14	97.93	95.61	89.86
<i>L1</i> $\lambda = 0.0001$	98.15	98.01	97.89	95.51	90.19
<i>L1</i> $\lambda = 0.001$	98.41	98.26	98.01	95.93	90.82
<i>L1</i> $\lambda = 0.01$	97.73	97.50	97.36	95.25	90.77
<i>L1</i> $\lambda = 0.1$	93.90	93.73	93.53	91.61	86.95
<i>L2</i> $\lambda = 0.0001$	98.00	97.85	97.66	95.34	89.38
<i>L2</i> $\lambda = 0.001$	97.88	97.71	97.51	95.02	89.23
<i>L2</i> $\lambda = 0.01$	98.45	98.30	98.14	96.19	91.15
<i>L2</i> $\lambda = 0.1$	98.12	98.04	97.92	96.12	91.69
<i>Rect</i>	97.99	97.83	97.63	95.27	89.83
<i>MT-DGL2</i> $\lambda = 0.001$ $\phi = 0.01$	98.38	98.14	97.90	93.12	80.36
<i>MT-DGL2</i> $\lambda = 0.001$ $\phi = 0.05$	98.50	98.26	97.96	93.43	80.28
<i>MT-DGL2</i> $\lambda = 0.001$ $\phi = 0.1$	98.35	98.11	97.86	93.00	79.57
<i>MT-DGL2</i> $\lambda = 0.01$ $\phi = 0.01$	98.57	98.35	98.08	93.71	81.96
<i>MT-DGL2</i> $\lambda = 0.01$ $\phi = 0.05$	98.65	98.38	98.05	93.77	82.71
<i>MT-DGL2</i> $\lambda = 0.01$ $\phi = 0.1$	98.59	98.39	98.16	93.88	82.13
<i>MT-DGL2</i> $\lambda = 0.1$ $\phi = 0.01$	98.85	98.73	98.53	95.33	85.69
<i>MT-DGL2</i> $\lambda = 0.1$ $\phi = 0.05$	98.76	98.53	98.32	94.72	84.69
<i>MT-DGL2</i> $\lambda = 0.1$ $\phi = 0.1$	98.70	98.53	98.38	94.66	84.27
<i>MT-DGL1</i> $\lambda = 0.001$ $\phi = 0.01$	98.58	98.45	98.21	93.97	82.65
<i>MT-DGL1</i> $\lambda = 0.001$ $\phi = 0.05$	98.77	98.62	98.44	95.40	86.81
<i>MT-DGL1</i> $\lambda = 0.001$ $\phi = 0.1$	98.63	98.46	98.28	94.07	84.26
<i>MT-DGL1</i> $\lambda = 0.01$ $\phi = 0.01$	98.82	98.64	98.33	94.84	85.10
<i>MT-DGL1</i> $\lambda = 0.01$ $\phi = 0.05$	98.91	98.75	98.55	95.86	88.21
<i>MT-DGL1</i> $\lambda = 0.01$ $\phi = 0.1$	98.99	98.55	97.96	93.23	86.97
<i>MT-DGL1</i> $\lambda = 0.1$ $\phi = 0.01$	98.63	98.51	98.35	96.64	93.01
<i>MT-DGL1</i> $\lambda = 0.1$ $\phi = 0.05$	98.99	98.95	98.85	98.05	95.82
<i>MT-DGL1</i> $\lambda = 0.1$ $\phi = 0.1$	<b>99.03</b>	<b>99.00</b>	<b>98.88</b>	<b>98.12</b>	<b>96.71</b>
<i>MT</i>	98.00	97.68	97.40	92.48	79.59

Table 8: Comparative Results Where All Models Are Attacked by the DataGrad-L2 Regularized Multitask Rectifier Network, *MT-DGL2*, Using Laplacian ( $\mathcal{R}_1$ ) Adversarial Noise Driven by the Target Task (Digit Recognition).

Model	$\phi = 0.0$	$\phi = 0.005$	$\phi = 0.01$	$\phi = 0.05$	$\phi = 0.1$
<i>DGL1</i> $\lambda = 0.0001$ $\phi = 0.01$	98.25	98.01	97.78	93.31	78.21
<i>DGL1</i> $\lambda = 0.0001$ $\phi = 0.05$	98.57	98.34	98.21	95.82	88.17
<i>DGL1</i> $\lambda = 0.0001$ $\phi = 0.1$	98.47	98.28	98.16	95.72	88.42
<i>DGL1</i> $\lambda = 0.001$ $\phi = 0.01$	98.03	97.81	97.48	93.29	79.74
<i>DGL1</i> $\lambda = 0.001$ $\phi = 0.05$	98.70	98.56	98.38	96.25	89.84
<i>DGL1</i> $\lambda = 0.001$ $\phi = 0.1$	98.62	98.48	98.37	96.32	90.73
<i>DGL1</i> $\lambda = 0.01$ $\phi = 0.01$	98.36	98.16	97.91	94.65	84.87
<i>DGL1</i> $\lambda = 0.01$ $\phi = 0.05$	98.62	98.50	98.44	96.49	91.22
<i>DGL1</i> $\lambda = 0.01$ $\phi = 0.1$	<b>98.83</b>	<b>98.77</b>	<b>98.67</b>	<b>97.66</b>	<b>94.71</b>
<i>DGL2</i> $\lambda = 0.0001$ $\phi = 0.01$	97.93	97.68	97.34	92.68	77.64
<i>DGL2</i> $\lambda = 0.0001$ $\phi = 0.05$	98.29	98.07	97.68	93.07	78.21
<i>DGL2</i> $\lambda = 0.0001$ $\phi = 0.1$	98.02	97.76	97.39	92.76	78.23
<i>DGL2</i> $\lambda = 0.001$ $\phi = 0.01$	97.97	97.64	97.38	92.88	77.47
<i>DGL2</i> $\lambda = 0.001$ $\phi = 0.05$	98.20	97.97	97.51	92.93	78.46
<i>DGL2</i> $\lambda = 0.001$ $\phi = 0.1$	98.26	98.05	97.72	93.33	78.70
<i>DGL2</i> $\lambda = 0.01$ $\phi = 0.01$	98.20	97.94	97.71	93.61	80.60
<i>DGL2</i> $\lambda = 0.01$ $\phi = 0.05$	98.31	98.10	97.79	93.53	80.19
<i>DGL2</i> $\lambda = 0.01$ $\phi = 0.1$	98.30	98.03	97.79	93.31	79.36
<i>L1</i> $\lambda = 0.0001$	98.15	97.94	97.73	93.21	78.52
<i>L1</i> $\lambda = 0.001$	98.41	98.14	97.83	94.35	81.61
<i>L1</i> $\lambda = 0.01$	97.73	97.46	97.31	94.35	86.50
<i>L1</i> $\lambda = 0.1$	93.90	93.59	93.41	90.34	83.60
<i>L2</i> $\lambda = 0.0001$	98.00	97.75	97.38	92.44	77.56
<i>L2</i> $\lambda = 0.001$	97.88	97.57	97.28	92.46	77.88
<i>L2</i> $\lambda = 0.01$	98.45	98.24	98.07	94.72	83.30
<i>L2</i> $\lambda = 0.1$	98.12	98.01	97.86	95.11	86.89
<i>Rect</i>	97.99	97.71	97.43	92.92	78.48
<i>MT-DGL2</i> $\lambda = 0.001$ $\phi = 0.01$	98.38	97.93	97.33	80.38	43.44
<i>MT-DGL2</i> $\lambda = 0.001$ $\phi = 0.05$	98.50	98.02	97.31	80.83	43.89
<i>MT-DGL2</i> $\lambda = 0.001$ $\phi = 0.1$	98.35	97.87	97.10	80.19	43.70
<i>MT-DGL2</i> $\lambda = 0.01$ $\phi = 0.01$	98.57	98.15	97.61	83.33	47.05
<i>MT-DGL2</i> $\lambda = 0.01$ $\phi = 0.05$	98.65	98.14	97.63	84.72	49.28
<i>MT-DGL2</i> $\lambda = 0.01$ $\phi = 0.1$	98.59	98.25	97.66	84.46	49.16
<i>MT-DGL2</i> $\lambda = 0.1$ $\phi = 0.01$	98.85	98.36	97.63	73.98	34.38
<i>MT-DGL2</i> $\lambda = 0.1$ $\phi = 0.05$	98.76	98.42	98.04	89.93	61.15
<i>MT-DGL2</i> $\lambda = 0.1$ $\phi = 0.1$	98.70	98.45	98.16	90.77	63.43
<i>MT-DGL1</i> $\lambda = 0.001$ $\phi = 0.01$	98.58	98.27	97.79	85.03	50.55
<i>MT-DGL1</i> $\lambda = 0.001$ $\phi = 0.05$	98.77	98.56	98.34	92.90	74.70
<i>MT-DGL1</i> $\lambda = 0.001$ $\phi = 0.1$	98.63	98.36	97.95	88.59	65.79
<i>MT-DGL1</i> $\lambda = 0.01$ $\phi = 0.01$	98.82	98.52	98.23	92.12	70.56
<i>MT-DGL1</i> $\lambda = 0.01$ $\phi = 0.05$	98.91	98.76	98.54	95.67	85.40
<i>MT-DGL1</i> $\lambda = 0.01$ $\phi = 0.1$	98.99	98.89	98.73	95.84	89.01
<i>MT-DGL1</i> $\lambda = 0.1$ $\phi = 0.01$	98.63	98.49	98.30	96.38	91.47
<i>MT-DGL1</i> $\lambda = 0.1$ $\phi = 0.05$	98.99	98.94	98.88	97.93	95.46
<i>MT-DGL1</i> $\lambda = 0.1$ $\phi = 0.1$	<b>99.03</b>	<b>99.00</b>	<b>98.90</b>	<b>98.11</b>	<b>96.59</b>
<i>MT</i>	98.00	97.43	96.80	82.60	49.99

the same adversarial scenarios for  $\mathcal{R}_2$  noise, as we found that it had little impact on model generalization ability (as we noted in section 3).

One key observation to take from this set of experiments on the MNIST data set is that DataGrad, particularly the L1 form, achieves the greatest level of robustness to adversarial samples in all settings when the  $\lambda$  and  $\phi$  are relatively higher. This is especially so when a DataGrad (L1) term is combined with the multitask objective. Note that this appears to be true no matter the adversary (even a DataGrad- or multitask-regularized one).

Perhaps even further performance improvement could be obtained if one added another DataGrad term to the auxiliary objective. In particular, this would apply to the rarer setting when one would also desire additional adversarial robustness with respect to the auxiliary objective. Tables 1 to 8, contain the full adversarial setting results.

### Acknowledgments

---

This work was supported by NSF grant CCF-1317560 and an NVIDIA hardware grant.

### References

---

- Bengio, Y., Lamblin, P., Popovici, D., & Larochelle, H. (2007). Greedy layer-wise training of deep networks. In B. Scholkopf, J. Platt, & T. Hoffman (Eds.), *Advances in neural information processing systems*, 19. Cambridge, MA: MIT Press.
- Bishop, C. M. (1992). Exact computation of the Hessian matrix for the multi-layer perceptron. *Neural Computation*, 4(4) 494–501.
- Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier networks. In *JMLR W & CP*, 15, 315–323.
- Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014). *Explaining and harnessing adversarial examples*. <http://arxiv.org/abs/1412.6572>
- Gu, S., & Rigazio, L. (2014). *Towards deep neural network architectures robust to adversarial examples*. <http://arxiv.org/abs/1412.5068>
- He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proc. IEEE Conference on Computer Vision* (pp. 1026–1034). Piscataway, NJ: IEEE.
- Huang, R., Xu, B., Schuurmans, D., & Szepesvari, C. (2015). *Learning with a strong adversary*. <http://arxiv.org/abs/1511.03034>
- Lyu, C., Huang, K., & Liang, H.-N. (2015). *A unified gradient regularization family for adversarial examples*. <http://arxiv.org/pdf/1511.06385.pdf>
- Miyato, T., Maeda, S.-I., Koyama, M., Nakae, K., & Ishii, S. (2015). *Distributional smoothing with virtual adversarial training*. <http://arxiv.org/abs/1507.00677>
- Nguyen, A., Yosinski, J., & Clune, J. (2014). *Deep neural networks are easily fooled: High confidence predictions for unrecognizable images*. <http://arxiv.org/abs/1412.1897>
- Nkland, A. (2015). *Improving back-propagation by adding an adversarial gradient*. <http://arxiv.org/abs/1510.04189>

- Ororbia II, A. G., Reitter, D., Wu, J., & Giles, C. L. (2015). Online learning of deep hybrid architectures for semi-supervised categorization. In *Lecture Notes in Computer Science: Vol. 9284. Machine Learning and Knowledge Discovery in Databases*. New York: Springer.
- Pearlmutter, B. A. (1994). Fast exact multiplication by the Hessian. *Neural Computation*, 6(1), 147–160.
- Rifai, S., Pascal, V., Muller, X., Glorot, X., & Bengio, Y. (2011). Contractive auto-encoders: Explicit invariance during feature extraction. In *Proc. of the 28th International Conference on Machine Learning (ICML-11)* (pp. 833–840). Madison: Omni Press.
- Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2013). *Intriguing properties of neural networks*. <http://arxiv.org/abs/1312.6199>

---

Received February 9, 2016; accepted October 22, 2016.