A. Brief Primer on Control Theory

Stability and Robustness of nonlinear systems have been studied in the field of control theory for more than a century. A nonlinear system is said to be stable, if given an input signal to the system, it can produce outputs that are expected by the user. A nonlinear system is said to be stable and robust, if given an input signal to the system and in the presence of external noise and disturbance, it can produce outputs that are expected by the user. In our work, we focus on bounded-input-bounded-output (BIBO) stability of nonlinear systems. Definition 1 in Preliminaries provides the mathematical definition for incremental BIBO stability. The Definition shows that the difference between two incremental outputs of the system should be bounded by the difference between their respective incremental inputs times γ. γ is a positive constant (gain) indicating the extent of boundedness property of the system. A small γ points to a more stable and robust system behavior. Our goal is to design a stable system that produces bounded outputs for the bounded signals it receives. In other words, we want to design a stable system that produces outputs close to the desired behavior defined by the user for the inputs from the domain of possible inputs. For instance, a DNN trained on the Imagenet data set is stable, if it produces desired classification decisions for any input image similar to the images in Imagenet data set. In our work, we focus on both stability and robustness. A robust stable DNN should be able to produce outputs close to the desired behavior defined by the user for the inputs from the domain possible inputs in cases where some amount noise has been added to the inputs. In our case, this noise is added by the adversary to the inputs.

A nonlinear system, given an input, state and output, is mathematically defined as follows,

$$H : \begin{cases} \dot{x} = f(x, u) \\ y = h(x, u), \end{cases}$$

where $x \in X \subseteq R^n$, $u \in U \subseteq R^m$, and $y \in Y \subseteq R^k$ are respectively the state, input and output of the system, and X, U and Y are the local state, input and output sub-spaces around the current operating points. The nonlinear mappings f and h model the relationship among the input signal u, the internal states of the system x and the output signal y. For a DNN trained on the ImageNet data set, U represents the images in the train and test data sets, Y represents the domain of possible class decisions, and X represents the domain of possible weight and bias values that could be assigned to the DNN's weights and biases. Further, x, u and y represents the possible realizations of these domains. As an example, x can represent the values assigned to the weights and biases of the DNN after training and convergence.

$\dot{x} = f(x, u)$ represents the dynamic behavior of the DNN during training, where u represents the training inputs to the DNN, and f models the updates to the states of the DNN during the t training iterations. This is also called the transient behavior of the system and $\dot{x}$ is a derivative taken over training iterations modeling the fact that the weights and biases are changing during training. In control theory, the transient behavior of a system is defined as the system's behavior before it reaches the stable equilibrium (steady-state). Lyapunov theory provides the foundation for defining stability and robustness for nonlinear systems solely based on their input-ouput behavior. This means that we can use Lyapunov results to define stability and robustness criteria for the system using the desired steady-state behavior $y = h(x, u_l) = h_l(\{W, B\}, u)$. This in return will determine the properties that the states (weights and biases) of the DNN should exhibit after the training is over.

Lyapunov theory treats the input-output relationship as an energy based concept and proves that if the input-output behavior of a nonlinear systems meets the relationship given in Definition 2 then the system is stable and robust. Lyapunov theory states that a stable and robust nonlinear system dissipate energy and as a results they have to be dissipative with respective to the relationship given in Definition 2.

In our work, we treat each layer inside a DNN as a nonlinear system, and use the relationship given in Definition 2, to characterize the conditions which the weights and biases of the layer should meet for that layer to be stable and robust. We make sure that these conditions are met during training so that once the training is over, the layer is stable and robust. Further, we define conditions under which the entire cascade of the systems (the entire DNN) is stable and robust and we make sure that these condition are met after the training as well. As a result, we can provide a set of design options for spectral regularization of the weights for each layer inside the DNN, and bound the response of each layer and eventually the response of the entire network against adversarial attacks and the adversarial noise added to the inputs. Given our results, we can train DNNs that outperform other state-of-the-art robust solutions in the current literature.

B. Proof of Theorem 2

Given Definition 2, we can show that for the two incremental inputs to the layer l, i.e., $u_{l,1} = u_l$ and $u_{l,2} = u_l + \Delta_l$ we have,

$$\begin{aligned} &\omega(u_l + \Delta_l - u_l, h(W_l[u_l + \Delta_l] + b_l) - h(W_l u_l + b_l)) \\ &= \Delta_l^T \Lambda_l W_l \Delta_l - \Delta_l^T (\nu_l I_l) \Delta_l - \Delta_l^T W_l^T \Lambda_l^T (\delta_l I_l) \Lambda_l W_l \Delta_l \end{aligned} \tag{1}$$

where,

$$\Lambda_l = \begin{bmatrix} 1 & 0 & \dots & \dots & \dots & \\ 0 & \ddots & \ddots & & & \vdots \\ \vdots & \ddots & 1 & 0 & & \vdots \\ \vdots & & 0 & a & \ddots & \\ \vdots & & & \ddots & \ddots & 0 \\ & \dots & \dots & & 0 & a \end{bmatrix}.$$

and I_l is a diagonal identity matrix of size layer l. Further for a layer to be instantaneously IIFOFP with some ν_l and a positive δ_l, one needs to show the following,

$$\begin{aligned} 0 \leq\ & [u_l + \Delta_l - u_l]^T [h_l(u_l + \Delta_l) - h_l(u_l)] \\ & - \delta_l [h_l(u_l + \Delta_l) - h_l(u_l)]^T [h_l(u_l + \Delta_l) - h_l(u_l)] \\ & - \nu_l [u_l + \Delta_l - u_l]^T [u_l + \Delta_l - u_l] \end{aligned}$$

Given (1) and for $\delta_l > 0$, the above can be represented as,

$$0 \leq (\frac{1}{2\delta_l} + |\nu_l|)||\Delta_l||_2^2 - \frac{\delta_l}{2}||\Lambda_l W_l \Delta_l||_2^2 \tag{2}$$

Further, the following properties hold, $||\Lambda_l W_l \Delta_l||_2^2 < ||\Lambda_l||_2^2 ||W_l||_2^2 ||\Delta_l||_2^2$, and since $0 < a < 1$, we have $||\Lambda_l||_2^2 < \lambda_{max}(\Lambda_l^T \Lambda_l) \leq 1$ where $\lambda_{max}(.)$ stands for the largest eigenvalue (singular value) of a matrix. Also we can show that $\rho(W_l) < ||W_l||_2^2 < \rho(W_l) + \sigma$ where $\rho(W_l)$ is the spectral radius of W_l and σ is a small positive number. Simplifying (2) further we have,

$$\begin{aligned} 0 & \leq (\frac{1}{2\delta_l} + |\nu_l|)||\Delta_l||_2^2 - \frac{\delta_l}{2}||\Lambda_l W_l \Delta_l||_2^2 \\ & \leq (\frac{1}{2\delta_l} + |\nu_l|)||\Delta_l||_2^2 - \frac{\delta_l}{2}||\Lambda_l||_2^2 ||W_l||_2^2 ||\Delta_l||_2^2 \\ & \leq (\frac{1}{2\delta_l} + |\nu_l|)||\Delta_l||_2^2 - \frac{\delta_l}{2}[(\rho(W_l) + \sigma)||\Delta_l||_2^2] \\ & \approx [\frac{1}{2\delta_l} + |\nu_l| - \frac{\delta_l}{2}\rho(W_l)]||\Delta_l||_2^2 \end{aligned} \tag{3}$$

For the relation (3) to be positive, the term inside the bracket needs to be positive. This gives us a measure for spectral regularization of the weights between each two hidden layers of a DNN. For layer l we have,

$$\rho(W_l) \leq \frac{1}{\delta_l^2} + \frac{2|\nu_l|}{\delta_l},$$

which proves the theorem.

C. Robustness analysis of a convolutional layer

The transformations before the activation functions at the convolutional layers are linear, and the same isomorphism as for the linear layers can be exploited for the convolutional layers to have the same final relationships as given in Theorem 2. More specifically, we can follow the steps given in [16] to define the transformation occurring at a convolutional layer l for output feature i with any padding and stride design as,

$$\phi_{l,i}^{conv}(u_{l,i}) = \sum_{j=1}^{M_{l-1}} f_{j,i} * u_{l,j,i} + b_{l,i}$$

Each $f_{j,i}$ is a filter applied to the input feature and each $u_{l,j,i}$ is an input feature map from the previous layer. $b_{l,i}$ is an appropriately shaped biased tensor adding the same value to every element resulting from the convolutions. M_{l-1} is the number of feature maps in the previous layer. One can represent the above relation as a matrix-vector multiplication by defining the serialized version of the input, $U_{l,i} = [u_{l,1,i}, ..., u_{l,M_{l-1},i}]$ and then representing the filter coefficients in the form of a doubly block circulant matrix [30]. Specifically, if $F_{j,i}$ is a matrix that encompasses convolution of $f_{j,i}$ with the j-th feature map in a vector form, then to represent convolutions associated with different input feature maps and the same output feature map i.e., $f_{j,i}$'s over M_{l-1} input features, one can horizontally concatenate the filter matrices to define $F_i = [F_{1,i}, F_{2,i}, ..., F_{M_{l-1},i}]$. Then the complete transformation performed by a convolutional layer to generate M_l output feature maps can be represented as,

$$h_l(U_l) = h_l(WU_l + B_l)$$

where,

$$W = \begin{bmatrix} F_{1,1} & \dots & F_{M_{l-1},1} \\ \vdots & \ddots & \vdots \\ F_{1,M_l} & \dots & F_{M_{l-1},M_l} \end{bmatrix}$$

and where vector B_l is the larger version of b_i^l's for all input feature maps and $U_l = [U_{l,1}, ..., U_{l,M_{l-1}}]$. Consequently, the spectral norm of W should meet the conditions given in Theorem 2 for the layer l to be IIFOFP and IIFG stable with bounded incremental outputs. We use the power iteration method to estimate the spectral norm of the weight matrix at a specific layer during training as proposed in [11]. Further, the pooling layers inside a DNN do not affect the conic behavior of the sub-systems given the properties of conic systems as described in [38]. More specifically, depending on how the pooling layer is designed, an adjustment is made to the ℓ_2 norm of the incremental output changes for the sub-system containing the pooling layer. Max (average) pooling decreases the norm of the changes and as a a result the conditions given in (2) is still met after the pooling layer.

D. Robustness analysis of a ResNet building block

The following represents the input-output mapping of a building block l for incremental inputs u_{l2}, u_{l1} and outputs y_{l2}, y_{l1} in a ResNet layer [17], where $y_{li} = u_{li} + \mathcal{F}(u_{li}, \{W_l\})$,

$$\begin{aligned}
M_l &= (u_{l2} - u_{l1})^T(y_{l2} - y_{l1}) - \delta_l(y_{l2} - y_{l1})^T(y_{l2} - y_{l1}) - \nu_l(u_{l2} - u_{l1})^T(u_{l2} - u_{l1}) \\
&= (u_{l2} - u_{l1})^T(u_{l2} + \mathcal{F}_2 - u_{l1} - \mathcal{F}_1) - \delta_l(u_{l2} + \mathcal{F}_2 - u_{l1} - \mathcal{F}_1)^T(u_{l2} + \mathcal{F}_2 - u_{l1} - \mathcal{F}_1) \\
&- \nu_l(u_{l2} - u_{l1})^T(u_{l2} - u_{l1}) \\
&= (u_{l2} - u_{l1})^T(\mathcal{F}_2 - \mathcal{F}_1) - \delta_l(u_{l2} + \mathcal{F}_2 - u_{l1} - \mathcal{F}_1)^T(u_{l2} + \mathcal{F}_2 - u_{l1} - \mathcal{F}_1) \\
&+ (1 - \nu_l)(u_{l2} - u_{l1})^T(u_{l2} - u_{l1}) \\
&= (1 - 2\delta_l)(u_{l2} - u_{l1})^T(\mathcal{F}_2 - \mathcal{F}_1) - \delta_l(\mathcal{F}_2 - \mathcal{F}_1)^T(\mathcal{F}_2 - \mathcal{F}_1) \\
&- (\nu_l + \delta_l - 1)(u_{l2} - u_{l1})^T(u_{l2} - u_{l1}) \\
&= (u_{l2} - u_{l1})^T(\mathcal{F}_2 - \mathcal{F}_1) - \frac{\delta_l}{1 - 2\delta_l}(\mathcal{F}_2 - \mathcal{F}_1)^T(\mathcal{F}_2 - \mathcal{F}_1) - \frac{\nu_l + \delta_l - 1}{1 - 2\delta_l}(u_{l2} - u_{l1})^T(u_{l2} - u_{l1})
\end{aligned}$$

where we have replaced $\mathcal{F}_i(u_{li}, \{W_l\})$ with $\mathcal{F}_i$ for simplicity. The above relation tells us that the feed-forward connection from the input to the output degrades the robustness of the ResNet block. This is because for a ResNet block l to be stable and robust, the following should hold for the Lyapunov parameters of the block excluding the feed-forward connection: $0 < \delta_l < \frac{1}{2}$ and $\nu_l + \delta_l > 1$ where $\delta_l \times \nu_l < 0.25$. So that the entire block can have the robustness properties: $\delta_l' = \frac{\nu_l + \delta_l - 1}{1 - 2\delta_l}$ and $\nu_l' = \frac{\nu_l + \delta_l - 1}{1 - 2\delta_l}$. This means that to maintain robustness and stability for a ResNet block, one needs to enforce stricter conditions on the Lyapunov design hyper-parameters of the sub-layers inside the block and consequently the spectral norm of the weights during the training of the DNN. In a sense, this means that a ResNet block is less robust against adversarial attacks in comparison to a simple feedforward or convolutional layer. This is expected because the output of a ResNet block consists of the block's output and the input signal fed into the block. This means that if the input signal is perturbed by adversarial noise, then under this architecture, the adversarial noise can easily propagate to the output of the block and throughout the DNN. The negative effects of feed-forward connections on robustness of nonlinear systems have been explored in control theory [22].

E. Proof of Theorem 3

The input-output mapping of each layer can be represented as $M_l = (u_{l2} - u_{l1})^T(y_{l2} - y_{l1}) - \delta_l(y_{l2} - y_{l1})^T(y_{l2} - y_{l1}) - \nu_l(u_{l2} - u_{l1})^T(u_{l2} - u_{l1}) > 0$ for layers $l = 1, .., N$, one needs to show that,

$$0 \leq \sum_{l=1}^{n} M_l \leq (u_2 - u_1)^T(y_2 - y_1) - \delta(y_2 - y_1)^T(y_2 - y_1) - \nu(u_{i2} - u_1)^T(u_2 - u_1).$$

The summation $\sum_{l=1}^{n} M_l$ is positive if the sub-layers are trained according to Theorem 2 and as a result one can show that the above relation is equivalent to,

$$\sum_{l=1}^{n} M_l - (u_2 - u_1)^T(y_2 - y_1) + \delta(y_2 - y_1)^T(y_2 - y_1) + \nu(u_2 - u_1)^T(u_2 - u_1) \leq 0. \tag{4}$$

We can define the following matrices,

$$A_1 = \begin{bmatrix} -\nu_1 & 0 & \dots & 0 & -\frac{1}{2} \\ \frac{1}{2} & -\nu_2 & \ddots & & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & \frac{1}{2} & -\nu_n & 0 \\ 0 & \dots & 0 & \frac{1}{2} & \delta \end{bmatrix},$$

and,

$$A_2 = \begin{bmatrix} \nu & 0 & \dots & 0 & -\frac{1}{2} \\ \frac{1}{2} & -\delta_1 & \ddots & & 0 \\ 0 & \frac{1}{2} & -\delta_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \frac{1}{2} & -\delta_n \end{bmatrix}.$$

As a result, (4) may be represented as,

$$[u^T y^T](A_1^T + A_2)[u^T y^T]^T = [u^T y^T]A[u^T y^T]^T,$$

where,

$$A = A_1^T + A_2 = \begin{bmatrix} \nu - \nu_1 & \frac{1}{2} & 0 & \dots & -\frac{1}{2} \\ \frac{1}{2} & -\delta_1 - \nu_2 & \frac{1}{2} & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & \frac{1}{2} & -\delta_{n-1} - \nu_n & \frac{1}{2} \\ -\frac{1}{2} & 0 & \dots & \frac{1}{2} & \delta - \delta_n \end{bmatrix}.$$

According to Corollary 1, if $-A$ is quasi-dominant, then$-A$ is positive definite, which means that A is negative definite and we have $[u^T y^T]A[u^T y^T]^T \leq 0$ and the relation given in (4) is met. As a result the DNN is instantaneously IIFOFP with indices δ and ν. For this to hold, the only condition is for the hyper-parameters to be selected such that the matrix $-A$ is quasi-dominant. For the case that we are interested in, we need the hyper-parameters to be selected such that $\delta > 0$, $\nu > 0$ where $\delta_n > \delta > 0$ and $\nu_1 > \nu > 0$, $\delta_i > 0$ for $i = 1, ..., n$ and ν_i for $i = 1, ..., n$ are selected such that the matrix $-A$ is quasi-dominant.

F. An example on how the Lyapunov parameters are selected

Here, we provide an example of parameter selection for the fully connected DNN used in some of our experiments with the global Lyapunov property of $\delta = 1.0$, $\nu = 0.24$. For a 3 layer fully connected forward-net, the following Lyapunov hyper-parameters should be selected: δ_1, ν_1 $(\rho(W_1))$, δ_2, ν_2 $(\rho(W_2))$, δ_3, ν_3 $(\rho(W_3))$, which then determine the global Lyapunov properties: δ, ν. The following conditions should be met for the matrix $-A$ to be quasi-dominant: $\nu < \nu_1$, $\delta < \delta_3$, $\delta_1 + \nu_2 > 1$, $\delta_2 + \nu_3 > 1$. The matrix $-A$ is:

$$-A = \begin{bmatrix} \nu_1 - \nu & -\frac{1}{2} & 0 & \frac{1}{2} \\ -\frac{1}{2} & \delta_1 + \nu_2 & -\frac{1}{2} & 0 \\ 0 & -\frac{1}{2} & \delta_2 + \nu_3 & -\frac{1}{2} \\ \frac{1}{2} & 0 & -\frac{1}{2} & \delta_3 - \delta \end{bmatrix}.$$

For $\delta = 1.0$, we have a range of choices for δ_3 ($\delta < \delta_3$), we select $\delta_3 = 1.08$ which is a robust choice due to its relatively large value (Subsection 4.3).This gives us the condition $\nu_3 < 0.25/1.08 = 0.231$, where $\nu_3 = 0.23$ is selected to meet the condition. As a result, the allowed range for spectral regularization for the last layer is $\rho(W_3) < \frac{1}{\delta_3^2} + \frac{2|\nu_3|}{\delta_3} = 1.283$. In a similar manner for $\nu = 0.24$, we have a range of choices for ν_1 ($\nu < \nu_1$), we select $\nu_1 = 0.27$. This gives us the condition $\delta_1 < 0.25/0.27 = 0.925$, where $\delta_1 = 0.92$ is selected to meet the condition. As a result, the allowed range for spectral regularization of the first layer becomes $\rho(W_1) < \frac{1}{\delta_1^2} + \frac{2|\nu_1|}{\delta_1} = 1.768$. Lastly, the values of ν_2 and δ_2 should be selected such that $\nu_2 > 0.08$ and $\delta_2 > 0.769$, we select $\delta_2 = 0.78$ which gives us $\nu_2 = 0.32$ and the spectral regularization range for the second layer becomes, $\rho(W_2) < \frac{1}{\delta_2^2} + \frac{2|\nu_2|}{\delta_2} = 2.464$. For our experiment, we use cross-validation to select the following spectral regularization set $[1.76, 2.46, 1.01]$. The matrix $-A$ becomes,

$$-A = \begin{bmatrix} 0.03 & -\frac{1}{2} & 0 & \frac{1}{2} \\ -\frac{1}{2} & 1.24 & -\frac{1}{2} & 0 \\ 0 & -\frac{1}{2} & 1.11 & -\frac{1}{2} \\ \frac{1}{2} & 0 & -\frac{1}{2} & 0.08 \end{bmatrix},$$

which is a quasi-dominant (diagonally dominant) matrix and positive definite with the spectral norm of 1.791 and Frobenius norm of 2.185.

G. Proof of Corollary 2

Given Theorem 3, the definitions for the following vectors, $\Delta_n = y_2 - y_1$ representing the changes in the output of the DNN, $\Delta_1 = u_1 + \Delta_1 - u_1$ representing the changes in the input of the DNN injected by the attacker and according to Theorem 2, we have,

$$0 \le \Delta_1^T \Delta_n - \Delta_n^T \delta I \Delta_n - \Delta_1^T \nu I \Delta_1$$

Given that $\delta > 0$ and $\nu > 0$, the above can be represented as,

$$0 \le (\frac{1}{2\delta} + \nu)||\Delta_1||_2^2 - \frac{\delta}{2}||\Delta_n||_2^2$$

Finally, if we move the appropriate terms to the left side of the above inequalities we have,

$$||\Delta_n||_2^2 \le (\frac{1}{\delta^2} + \frac{2\nu}{\delta})||\Delta_1||_2^2.$$

H. Experiment design and hyper-parameter details

This appendix includes all hyper-parameters used in the experiments. Please note that, each layer has their own δ and ν associated with them. The pair (δ, ν) then determine the level of spectral regularization, given in parenthesis, enforced at the layer during the training of network. The δ's and ν's are selected according to the conditions presented in Definition 2, Theorem 1, Theorem 2, Theorem 3 and Corollary 1. The selections of the pairs (δ, ν) for the layers then lead to a global Lyapunov pair (δ, ν) for the entire network which then later is used to present a specific network in the tables.

H.1. The Lyapunov design parameters for Forward-Net (trained on the MNIST dataset)

Table 4: The hyper-parameters for the fully connected Forward-Net (3 layers of sizes [50, 20, 10]) used in the experiments

Forward-Net	Layer 1 (linear)	Layer 2 (linear)	Layer 3 (linear)	Global Lyapunov Property
Design Parameters	δ, ν (spect. norm reg.)	δ, ν (spect. norm reg.)	δ, ν (spect. norm reg.)	δ, ν
	$\delta = 0.86$, $\nu = 0.29$ (1.83)	$\delta = 0.90$, $\nu = 0.27$ (1.83)	$\delta = 0.90$, $\nu = 0.27$ (1.83)	$\delta = 0.89$, $\nu = 0.28$
	$\delta = 0.92$, $\nu = 0.27$ (1.76)	$\delta = 0.98$, $\nu = 0.25$ (1.55)	$\delta = 0.96$, $\nu = 0.26$ (1.62)	$\delta = 0.95$, $\nu = 0.26$
	$\delta = 0.92$, $\nu = 0.27$ (1.76)	$\delta = 0.78$, $\nu = 0.32$ (2.46)	$\delta = 1.08$, $\nu = 0.23$ (1.01)	$\delta = 1.0$, $\nu = 0.24$
	N/A	N/A	N/A	None (Baseline Training with Freb. ℓ_2 reg., $\lambda = 0.01$)
	N/A	N/A	N/A	None (Baseline Training with Freb. ℓ_2 reg., $\lambda = 0.05$)
	N/A	N/A	N/A	None (Baseline Training with Freb. ℓ_2 reg., $\lambda = 0.1$)

H.2. The Lyapunov design parameters for AlexNet (trained on the CIFAR-10 dataset)

Table 5: The hyper-parameters for the AlexNet architecture used in the experiments

AlexNet:	Layer 1 (conv.)	Layer 2 (conv.)	Layer 3 (conv.)	Layer 4 (linear)	Layer 5 (linear)	Global Lyapunov Property
Design Parameters	δ, ν (spect. norm reg.)	δ, ν (spect. norm reg.)	δ, ν (spect. norm reg.)	δ, ν (spect. norm reg.)	δ, ν (spect. norm reg.)	δ, ν
	$\delta = 0.74$, $\nu = 0.33$ (2.52)	$\delta = 0.74$, $\nu = 0.33$ (2.52)	$\delta = 0.74$, $\nu = 0.33$ (2.52)	$\delta = 0.74$, $\nu = 0.33$ (2.52)	$\delta = 0.86$, $\nu = 0.29$ (2.02)	$\delta = 0.85$, $\nu = 0.29$
	$\delta = 0.86$, $\nu = 0.29$ (2.02)	$\delta = 0.86$, $\nu = 0.29$ (2.02)	$\delta = 0.86$, $\nu = 0.29$ (2.02)	$\delta = 0.86$, $\nu = 0.29$ (2.02)	$\delta = 0.92$, $\nu = 0.27$ (1.76)	$\delta = 0.90$, $\nu = 0.27$
	$\delta = 0.92$, $\nu = 0.27$ (1.76)	$\delta = 0.78$, $\nu = 0.32$ (2.46)	$\delta = 0.78$, $\nu = 0.32$ (2.46)	$\delta = 0.78$, $\nu = 0.32$ (2.46)	$\delta = 1.08$, $\nu = 0.23$ (1.01)	$\delta = 1.07$, $\nu = 0.22$
	N/A	N/A	N/A	N/A	N/A	None (Baseline Training)
	N/A	N/A	N/A	N/A	N/A	None (Baseline Training with Freb. ℓ_2 reg., $\lambda = 0.01$)
	N/A, N/A (1.0)	N/A, N/A (1.0)	N/A, N/A (1.0)	N/A, N/A (1.0)	N/A, N/A (1.0)	None (Training with spectral reg. <1 for each layer)

H.3. The Lyapunov design parameters for the ResNet architecture (trained on the SVHN dataset)

Table 6: The hyper-parameters for the ResNet architecture (conv. layer, 4 blocks of size 2, linear layer) used in the experiments

ResNet	Layer 1 (conv.)	Block 1	Block 2	Block 3	Block 4	Layer 5 (linear)	Global Lyapunov Property
Design Parameters	δ, ν	δ, ν δ_1, ν_1, δ_2, ν_2 (spect. norm reg.), (spect. norm reg.)	δ, ν, δ_1, ν_1, δ_2, ν_2 (spect. norm reg.), (spect. norm reg.)	δ, ν, δ_1, ν_1, δ_2, ν_2 (spect. norm reg.), (spect. norm reg.)	δ, ν δ_1, ν_1, δ_2, ν_2 (spect. norm reg.), (spect. norm reg.)	δ, ν (spect. norm reg.)	δ, ν
	$\delta = 0.84$, $\nu = 0.29$ (2.08)	$\delta = 1.16$, $\nu = 0.20$ $\delta_1 = 0.351$, $\nu_1 = 0.711$, $\delta_2 = 0.351$, $\nu_2 = 0.711$ (2.06), (2.06)	$\delta = 1.16$, $\nu = 0.20$ $\delta_1 = 0.464$, $\nu_1 = 0.540$, $\delta_2 = 0.464$, $\nu_2 = 0.540$ (2.06), (2.06)	$\delta = 1.16$, $\nu = 0.20$ $\delta_1 = 0.351$, $\nu_1 = 0.711$, $\delta_2 = 0.351$, $\nu_2 = 0.711$ (2.06), (2.06)	$\delta = 1.16$, $\nu = 0.20$ $\delta_1 = 0.351$, $\nu_1 = 0.711$, $\delta_2 = 0.351$, $\nu_2 = 0.711$ (2.06), (2.06)	$\delta = 0.83$, $\nu = 0.30$ (1.80)	$\delta = 0.80$, $\nu = 0.28$
	$\delta = 0.9$, $\nu = 0.27$ (1.76)	$\delta = 1.47$, $\nu = 0.17$ $\delta_1 = 0.37$, $\nu_1 = 0.66$, $\delta_2 = 0.37$, $\nu_2 = 0.66$ (2.48), (2.48)	$\delta = 1.47$, $\nu = 0.17$ $\delta_1 = 0.37$, $\nu_1 = 0.66$, $\delta_2 = 0.37$, $\nu_2 = 0.66$ (2.48), (2.48)	$\delta = 1.47$, $\nu = 0.17$ $\delta_1 = 0.37$, $\nu_1 = 0.66$, $\delta_2 = 0.37$, $\nu_2 = 0.66$ (2.48), (2.48)	$\delta = 1.47$, $\nu = 0.17$ $\delta_1 = 0.37$, $\nu_1 = 0.66$, $\delta_2 = 0.37$, $\nu_2 = 0.66$ (2.48), (2.48)	$\delta = 0.92$, $\nu = 0.27$ (1.01)	$\delta = 0.91$, $\nu = 0.26$
	N/A	N/A	N/A	N/A	N/A	N/A	None (Baseline Training)
	N/A	N/A	N/A	N/A	N/A	N/A	None Baseline Network with Freb., ℓ_2 reg., $\lambda = 0.05$

H.4. The Lyapunov design parameters for the ResNet50 architecture (trained on the ImageNet dataset)

Table 7: The hyper-parameters for the ResNet50 architecture (conv. layer, 4 blocks of sizes $[3, 4, 6, 3]$, linear layer) used in the experiments

ResNet	Layer 1 (conv.)	Block 1	Block 2	Block 3	Block 4	Layer 5 (linear)	Global Lyapunov Property
Design Parameters	δ, ν	δ, ν $\delta_l, \nu_l, l = 1, ..., 3$ $3\times$(spect. norm reg.)	$\delta, \nu,$ $\delta_l, \nu_l, l = 1, ..., 4$ $4\times$(spect. norm reg.)	$\delta, \nu,$ $\delta_l, \nu_l, l = 1, ..., 6$ $6\times$(spect. norm reg.)	δ, ν $\delta_l, \nu_l, l = 1, ..., 3$ $3\times$(spect. norm reg.)	δ, ν (spect. norm reg.)	δ, ν
	$\delta = 0.87, \nu = 0.29$ (1.60)	$\delta = 1.16, \nu = 0.20$ $\delta_l = 0.351, \nu_l = 0.711, l = 1, ..., 3$ $3 \times (1.61)$	$\delta = 1.16, \nu = 0.20$ $\delta_l = 0.351, \nu_l = 0.711, l = 1, ..., 4$ $4 \times (1.61)$	$\delta = 1.16, \nu = 0.20$ $\delta_l = 0.351, \nu_l = 0.711, l = 1, ..., 3$ $6 \times (1.61)$	$\delta = 1.16, \nu = 0.20$ $\delta_l = 0.351, \nu_l = 0.711, l = 1, ..., 3$ $3 \times (1.61)$	$\delta = 0.88, \nu = 0.28$ (1.60)	$\delta = 0.89, \nu = 0.28$
	$\delta = 0.93, \nu = 0.27$ (2.0)	$\delta = 1.16, \nu = 0.20$ $\delta_l = 0.351, \nu_l = 0.711, l = 1, ..., 3$ $3 \times (2.06)$	$\delta = 1.16, \nu = 0.20$ $\delta_l = 0.351, \nu_l = 0.711, l = 1, ..., 4$ $4 \times (2.06)$	$\delta = 1.16, \nu = 0.20$ $\delta_l = 0.351, \nu_l = 0.711, l = 1, ..., 6$ $6 \times (2.06)$	$\delta = 1.16, \nu = 0.20$ $\delta_l = 0.351, \nu_l = 0.711, l = 1, ..., 3$ $3 \times (2.06)$	$\delta = 0.94, \nu = 0.26$ (2.0)	$\delta = 0.95, \nu = 0.26$
	$\delta = 0.99, \nu = 0.25$ (2.52)	$\delta = 1.47, \nu = 0.17$ $\delta_l = 0.37, \nu_l = 0.66, l = 1, ..., 3$ $3 \times (2.48)$	$\delta = 1.47, \nu = 0.17$ $\delta_l = 0.37, \nu_l = 0.66, l = 1, ..., 4$ $4 \times (2.48)$	$\delta = 1.47, \nu = 0.17$ $\delta_l = 0.37, \nu_l = 0.66, l = 1, ..., 6$ $6 \times (2.48)$	$\delta = 1.47, \nu = 0.17$ $\delta_l = 0.37, \nu_l = 0.66, l = 1, ..., 3$ $3 \times (2.48)$	$\delta = 0.99, \nu = 0.24$ (2.52)	$\delta = 1.0, \nu = 0.24$
	N/A	N/A	N/A	N/A	N/A	N/A	None (Baseline Training)
	N/A	N/A	N/A	N/A	N/A	N/A	None Baseline Network with Freb., ℓ_2 reg., $\lambda = 0.01$

I. Experiment results

This appendix includes the experiment results for all the above architectures against FGM and Iterative PGD attacks. A pair of results are presented per DNN for the cases where a DNN was trained with or without adversarial training.

I.1. The experiment results for the Forward-Net architecture (trained on the MNIST dataset)

Table 8: Experiment results for the FGM attack

Network Type	Type of Attack	Type of Training	Accuracy on the Adversarial Test Dataset (Attack Strength ϵ)		
			$\epsilon = 0.1$	$\epsilon = 0.2$	$\epsilon = 0.3$
$\delta = 0.89$, $\nu = 0.28$	FGM		**0.966**	**0.953**	0.934
$\delta = 0.95$, $\nu = 0.26$	FGM		0.962	0.951	**0.935**
$\delta = 1.0$, $\nu = 0.24$	FGM		0.948	0.939	0.927
Baseline Network with Freb. ℓ_2 reg., $\lambda = 0.01$	FGM		0.927	0.915	0.902
Baseline Training with Freb. ℓ_2 reg., $\lambda = 0.05$	FGM		0.870	0.859	0.840
Baseline Training with Freb. ℓ_2 reg., $\lambda = 0.1$	FGM		0.816	0.809	0.797
$\delta = 0.89$, $\nu = 0.28$	FGM	with ℓ_2 FGM Adv. Training	**0.971**	**0.965**	**0.954**
$\delta = 0.95$, $\nu = 0.26$	FGM	with ℓ_2 FGM Adv. Training	0.966	0.956	0.950
$\delta = 1.0$, $\nu = 0.24$	FGM	with ℓ_2 FGM Adv. Training	0.952	0.941	0.935
Baseline Network with Freb. ℓ_2 reg., $\lambda = 0.01$	FGM	with ℓ_2 FGM Adv. Training	0.908	0.898	0.885
Baseline Network with Freb. ℓ_2 reg., $\lambda = 0.05$	FGM	with ℓ_2 FGM Adv. Training	0.822	0.805	0.793
Baseline Network with Freb. ℓ_2 reg., $\lambda = 0.1$	FGM	with ℓ_2 FGM Training	0.592	0.593	0.576

Table 9: Experiment results for the Iterative PGD attack (k=100)

Network Type	Type of Attack	Type of Training	Accuracy on the Adversarial Test Dataset (Attack Strength ϵ)		
			$\epsilon = 0.1$	$\epsilon = 0.2$	$\epsilon = 0.3$
$\delta = 0.89, \nu = 0.28$	PGD		**0.966**	**0.953**	0.933
$\delta = 0.95, \nu = 0.26$	PGD		0.962	0.951	**0.934**
$\delta = 1.0, \nu = 0.24$	PGD		0.948	0.939	0.926
Baseline Network with Freb. ℓ_2 reg., $\lambda = 0.01$	PGD		0.927	0.916	0.901
Baseline Training with Freb. ℓ_2 reg., $\lambda = 0.05$	PGD		0.869	0.858	0.839
Baseline Training with Freb. ℓ_2 reg., $\lambda = 0.1$	PGD		0.815	0.809	0.796
$\delta = 0.89, \nu = 0.28$	PGD	with ℓ_2 PGD Adv. Training	**0.970**	**0.963**	**0.954**
$\delta = 0.95, \nu = 0.26$	PGD	with ℓ_2 PGD Adv. Training	0.962	0.956	0.946
$\delta = 1.0, \nu = 0.24$	PGD	with ℓ_2 PGD Adv. Training	0.950	0.943	0.933
Baseline Network with Freb. ℓ_2 reg., $\lambda = 0.01$	PGD	with ℓ_2 PGD Adv. Training	0.639	0.633	0.625
Baseline Network with Freb. ℓ_2 reg., $\lambda = 0.05$	PGD	with ℓ_2 PGD Adv. Training	0.113	0.113	0.113
Baseline Network with Freb. ℓ_2 reg., $\lambda = 0.1$	PGD	with ℓ_2 PGD Adv. Training	0.113	0.113	0.113

I.2. The experiment results for the AlexNet architecture (trained on the CIFAR-10 dataset)

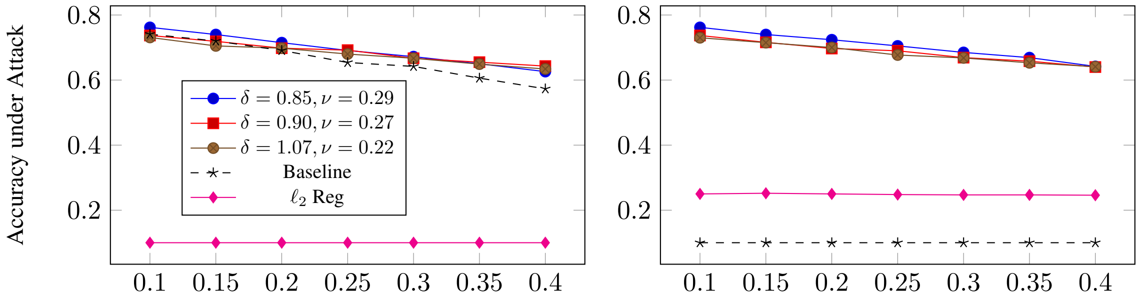

Figure 4: Accuracy of model under PGD attack with adversarial training (left) and under FGM attack with adversarial training (right) using AlexNet on CIFAR10. Plots share the same legend, x-axis indicates the power of the attack ϵ.

Table 10: Experiment results for the FGM attack

Network Type	Type of Attack	Type of Training	Accuracy on the Adversarial Test Dataset (Attack Strength ϵ)						
			$\epsilon = 0.10$	$\epsilon = 0.15$	$\epsilon = 0.20$	$\epsilon = 0.25$	$\epsilon = 0.30$	$\epsilon = 0.35$	$\epsilon = 0.40$
$\delta = 0.85$, $\nu = 0.29$	FGM		**0.745**	**0.708**	0.659	0.612	0.565	0.515	0.469
$\delta = 0.90$, $\nu = 0.27$	FGM		0.733	0.705	**0.678**	**0.644**	**0.609**	0.573	0.536
$\delta = 1.07$, $\nu = 0.22$	FGM		0.721	0.698	0.669	0.639	0.606	**0.575**	**0.539**
Baseline Network	FGM		0.729	0.680	0.620	0.563	0.508	0.458	0.424
Baseline Training with Freb. ℓ_2 reg., $\lambda = 0.01$	FGM		0.409	0.407	0.405	0.404	0.397	0.394	0.391
$\delta = 0.85$, $\nu = 0.29$	FGM	with ℓ_2 FGM Adv. Training	**0.762**	**0.740**	**0.724**	**0.705**	**0.685**	**0.669**	**0.642**
$\delta = 0.90$, $\nu = 0.27$	FGM	with ℓ_2 FGM Adv. Training	0.737	0.716	0.697	0.690	0.669	0.658	0.640
$\delta = 1.07$, $\nu = 0.22$	FGM	with ℓ_2 FGM Adv. Training	0.730	0.715	0.700	0.677	0.668	0.653	0.641
Baseline Network	FGM	with ℓ_2 FGM Adv. Training	0.100	0.100	0.100	0.100	0.100	0.100	0.100
Baseline Training with Freb. ℓ_2 reg., $\lambda = 0.01$	FGM	with ℓ_2 FGM Adv. Training	0.250	0.252	0.250	0.248	0.247	0.2478	0.246

Table 11: Experiment results for the Iterative PGD attack (k=100)

Network Type	Type of Attack	Type of Training	Accuracy on the Adversarial Test Dataset (Attack Strength ϵ)						
			$\epsilon = 0.10$	$\epsilon = 0.15$	$\epsilon = 0.20$	$\epsilon = 0.25$	$\epsilon = 0.30$	$\epsilon = 0.35$	$\epsilon = 0.40$
$\delta = 0.85$, $\nu = 0.29$	PGD (k=100)		**0.743**	0.700	0.645	0.590	0.530	0.473	0.415
$\delta = 0.90$, $\nu = 0.27$	PGD (k=100)		0.731	**0.702**	**0.672**	**0.633**	**0.591**	0.545	0.504
$\delta = 1.07$, $\nu = 0.22$	PGD (k=100)		0.720	0.696	0.662	0.626	0.590	**0.547**	**0.510**
Baseline Network	PGD (k=100)		0.730	0.672	0.608	0.541	0.478	0.419	0.367
Baseline Training with Freb. ℓ_2 reg., $\lambda = 0.01$	PGD (k=100)		0.409	0.407	0.405	0.402	0.396	0.393	0.390
$\delta = 0.85$, $\nu = 0.29$	PGD (k=100)	with ℓ_2 PGD Adv. Training	**0.762**	**0.740**	**0.715**	0.691	**0.672**	0.650	0.626
$\delta = 0.90$, $\nu = 0.27$	PGD (k=100)	with ℓ_2 PGD Adv. Training	0.737	0.719	0.698	**0.692**	0.667	**0.655**	**0.643**
$\delta = 1.07$, $\nu = 0.22$	PGD (k=100)	with ℓ_2 PGD Adv. Training	0.731	0.705	0.698	0.680	0.667	0.649	0.634
Baseline Network	PGD (k=100)	with ℓ_2 PGD Adv. Training	0.741	0.720	0.691	0.654	0.642	0.606	0.573
Baseline Training with Freb. ℓ_2 reg., $\lambda = 0.01$	PGD (k=100)	with ℓ_2 PGD Adv. Training	0.100	0.100	0.100	0.100	0.100	0.100	0.100

I.3. The experiment results for the ResNet architecture (trained on the SVHN dataset)

Table 12: Experiment results for the Iterative PGD attack (k=100)

Network Type	Type of Attack	Type of Training	Accuracy on the Adversarial Test Dataset (Attack Strength ϵ)						
			$\epsilon = 0.10$	$\epsilon = 0.15$	$\epsilon = 0.20$	$\epsilon = 0.25$	$\epsilon = 0.30$	$\epsilon = 0.35$	$\epsilon = 0.40$
$\delta = 0.80$, $\nu = 0.28$	PGD (k=100)		**0.869**	**0.814**	**0.746**	**0.674**	**0.607**	**0.545**	0.486
$\delta = 0.91$, $\nu = 0.26$	PGD (k=100)		0.862	0.808	0.742	**0.674**	0.606	0.544	**0.488**
Baseline Network	PGD (k=100)		0.850	0.783	0.711	0.631	0.565	0.493	0.431
Baseline Training with Freb. ℓ_2 reg., $\lambda = 0.01$	PGD (k=100)		0.850	0.783	0.719	0.640	0.571	0.509	0.455

J. Robustness performance against the Carlini & Wagner (C&W) attack

We tried our approach against the C&W attack. The parameters were chosen according to the tutorials presented in [27]. This appendix includes the results for this experiment.

Table 13: C&W attack parameters: $\{$learning rate: 0.001, initial const: 0.01, max iter: 500$\}$

Network Type/Dataset AlexNet/Cifar-10	Accuracy on the adversarial dataset	Network Type/Dataset ForwardNet/MNIST	Accuracy on the adversarial dataset
$\delta = 0.86$, $\nu = 0.29$	0.302	$\delta = 0.89$, $\nu = 0.28$	0.375
$\delta = 0.90$, $\nu = 0.27$	**0.335**	$\delta = 0.95$, $\nu = 0.26$	0.466
$\delta = 1.07$, $\nu = 0.22$	0.332	$\delta = 1.0$, $\nu = 0.24$	**0.470**
Baseline Network	0.265	Baseline Network	0.382
Baseline Training with Freb. ℓ_2 reg., $\lambda = 0.01$	0.157	Baseline Training with Freb. ℓ_2 reg., $\lambda = 0.05$	0.318
Baseline Training with spec. reg. $<$1 for all layers	0.281		

Table 14: C&W attack parameters: $\{$learning rate: 0.0001, initial const: 0, max iter: 500$\}$

Network Type/Dataset AlexNet/Cifar-10	Accuracy on the adversarial dataset	Network Type/Dataset ForwardNet/MNIST	Accuracy on the adversarial dataset
$\delta = 0.86$, $\nu = 0.29$	**0.787**	$\delta = 0.89$, $\nu = 0.28$	0.970
$\delta = 0.90$, $\nu = 0.27$	0.755	$\delta = 0.95$, $\nu = 0.26$	**0.975**
$\delta = 1.07$, $\nu = 0.22$	0.740	$\delta = 1.0$, $\nu = 0.24$	0.951
Baseline Network	0.717	Baseline Network	0.934
Baseline Training with Freb. ℓ_2 reg., $\lambda = 0.01$	0.407	Baseline Training with Freb. ℓ_2 reg., $\lambda = 0.05$	0.872
Baseline Training with spec. reg. $<$1 for all layers	0.441		

K. Robustness performance comparison with prior spectral normalization approaches

This appendix details a set of experiments performed to provide a robustness comparison between our proposed approach and the works given in [11], [28], [37] and [8]. [28], [11] propose training networks with the same spectral regularization enforced across the entire network where $\rho(W_l) \leq \beta$ for $l = 1, ..., n$ and β is a constant. We select 3 values of β from their papers: $\beta = 1.0, 1.6, 2.0$. The 2 works given in [37] and [8] may be seen as subsets of the works given in [28] and [11], where $\rho(W_l) \leq \beta$ for $l = 1, ..., n$ and $\beta = 1.0$ are selected. All the aforementioned works are special cases of our proposed Lyapunov robust solution. Our experiments confirm that our approach provides 3 main benefits that the other works do not provide. 1- Our work provides the theory, reasoning and interpretation behind why spectral regularization enhances robustness, and in particular how the spectral regularization hyper-parameters for each layer should be selected. 2- Our work provides a higher level of flexibility and freedom in selecting the hyper-parameter for training based the interpretations and reasoning behind our theory and work. 3- Our proposed approach produces trained DNNs that are more robust and perform better than the other works in this area.

Table 15: The hyper-parameters for the AlexNet architecture used in the experiments

AlexNet:	Layer 1 (conv.)	Layer 2 (conv.)	Layer 3 (conv.)	Layer 4 (linear)	Layer 5 (linear)	Global Lyapunov Property
Design Parameters	δ, ν (spect. norm reg.)	δ, ν (spect. norm reg.)	δ, ν (spect. norm reg.)	δ, ν (spect. norm reg.)	δ, ν (spect. norm reg.)	δ, ν
	$\delta = 0.81, \nu = 0.30$ (1.49)	$\delta = 0.96, \nu = 0.20$ (1.49)	$\delta = 0.96, \nu = 0.20$ (1.49)	$\delta = 0.70, \nu = 0.35$ (2.12)	$\delta = 0.80, \nu = 0.23$ (3.14)	$\delta = 0.79, \nu = 0.29$
	$\delta = 0.86, \nu = 0.29$ (1.59)	$\delta = 0.86, \nu = 0.29$ (1.89)	$\delta = 0.96, \nu = 0.26$ (1.62)	$\delta = 0.96, \nu = 0.26$ (1.62)	$\delta = 0.90, \nu = 0.27$ (2.81)	$\delta = 0.89, \nu = 0.28$
	$\delta = 0.88, \nu = 0.28$ (1.59)	$\delta = 0.89, \nu = 0.28$ (1.89)	$\delta = 0.76, \nu = 0.32$ (2.52)	$\delta = 0.96, \nu = 0.26$ (1.62)	$\delta = 0.91, \nu = 0.27$ (1.80)	$\delta = 0.90, \nu = 0.27$
	N/A, N/A (1.0)	N/A, N/A (1.0)	N/A, N/A (1.0)	N/A, N/A (1.0)	N/A, N/A (1.0)	None (Training with spectral reg. <1.0 across all layers)
	N/A, N/A (1.6)	N/A, N/A (1.6)	N/A, N/A (1.6)	N/A, N/A (1.6)	N/A, N/A (1.6)	None (Training with spectral reg. <1.6 across all layers)
	N/A, N/A (2.0)	N/A, N/A (2.0)	N/A, N/A (2.0)	N/A, N/A (2.0)	N/A, N/A (2.0)	None (Training with spectral reg. <2.0 across all layers)

Table 16: Experiment results for the FGM attack

Network Type	Type of Attack	Accuracy on the Adversarial Test Dataset (Attack Strength ϵ)						
		$\epsilon = 0.10$	$\epsilon = 0.15$	$\epsilon = 0.20$	$\epsilon = 0.25$	$\epsilon = 0.30$	$\epsilon = 0.35$	$\epsilon = 0.40$
$\delta = 0.79, \nu = 0.29$	FGM	0.720	**0.704**	0.678	0.646	**0.614**	0.579	0.547
$\delta = 0.89, \nu = 0.28$	FGM	**0.722**	0.701	0.676	**0.647**	0.613	**0.583**	**0.550**
$\delta = 0.90, \nu = 0.27$	FGM	0.721	0.703	**0.703**	0.643	0.609	0.578	0.548
Baseline Training with spec. reg. <1 across all layers	FGM	0.437	0.436	0.432	0.430	0.427	0.425	0.419
Baseline Training with spec. reg. <1.6 across all layers	FGM	0.665	0.656	0.635	0.614	0.589	0.562	0.538
Baseline Training with spec. reg. <2.0 across all layers	FGM	0.719	0.701	0.673	0.641	0.608	0.569	0.534

Table 17: Experiment results for the Iterative PGD attack (k=100)

Network Type	Type of Attack	Accuracy on the Adversarial Test Dataset (Attack Strength ϵ)						
		$\epsilon = 0.10$	$\epsilon = 0.15$	$\epsilon = 0.20$	$\epsilon = 0.25$	$\epsilon = 0.30$	$\epsilon = 0.35$	$\epsilon = 0.40$
$\delta = 0.79, \nu = 0.29$	PGD	0.720	**0.703**	**0.671**	0.635	**0.599**	0.556	0.520
$\delta = 0.89, \nu = 0.28$	PGD	**0.722**	0.698	0.670	**0.638**	0.598	**0.563**	**0.523**
$\delta = 0.90, \nu = 0.27$	PGD	0.721	0.701	0.670	0.633	0.593	0.557	0.522
Baseline Training with spec. reg. <1 across all layers	PGD	0.437	0.436	0.432	0.430	0.427	0.425	0.419
Baseline Training with spec. reg. <1.6 across all layers	PGD	0.665	0.654	0.633	0.611	0.580	0.554	0.522
Baseline Training with spec. reg. <2.0 across all layers	PGD	0.717	0.696	0.667	0.629	0.587	0.543	0.500

Table 18: The test accuracy of the DNNs after training

Network Type	Test Accuracy on the Clean Dataset	Network Type	Test Accuracy on the Clean Dataset
$\delta = 0.79, \nu = 0.29$	0.74	Baseline Training with spec. reg. <1 across all layers	0.44
$\delta = 0.89, \nu = 0.28$	0.74	Baseline Training with spec. reg. <1.6 across all layers	0.68
$\delta = 0.90, \nu = 0.27$	0.74	Baseline Training with spec. reg. <2.0 across all layers	0.74

L. Further details on the experiments

We validate our results by training several 3-layer fully connected forward-nets on the MNIST data set using the Adam optimizer with a learning rate of 0.001. Further, we train several AlexNet architectures on the CIFAR10 data set using Stochastic Gradient Descent optimization with a learning rate of 0.01 and momentum of 0.9. Additionally, we train several ResNet architectures on the SVHN data set and ResNet50 architectures on the Imagenet data set using Stochastic Gradient Descent optimization with a learning rate of 0.01 and momentum of 0.9. All networks are trained for 200 epochs. The pixel values for the input images are normalized to take values in $[-0.5, +0.5]$. When considering the baseline defense of weight decay (i.e., ℓ_2 regularized weights) networks, we performed cross-validation to select $\lambda \in [0.01, 0.05, 0.1]$. The parameters for our robust Lyapunov approach for each network architecture used in the experiments are specified in Appendix H. Baseline represents a DNN with no regularization enforced during training. Adversarial training for ℓ_2 FGM and ℓ_2 PGD attacks in our experiments follow the traditional definition of adversarial training, which indicate trainings on datasets of synthetically generated adversarial data points to enhance robustness.

All experiments were performed on a single Nvidia Tesla V100-SXM2-16GB GPU. The MNIST data set was downloaded from `http://yann.lecun.com/exdb/mnist/`. The data is randomly divided into the training, testing and validation data sets of size: 60000, 10000 and 5000 receptively. The CIFAR10 data set was downloaded from `https://www.cs.toronto.edu/~kriz/cifar.html`. The data is randomly divided into the training, testing and validation data sets of size: 45000, 10000 and 5000 receptively. The SVHN data set was downloaded from `http://ufldl.stanford.edu/housenumbers/`. The data is randomly divided into the training, testing and validation data sets of size: 73257, 26032 and 500 receptively. The ImageNet data set was downloaded from `http://image-net.org/download`. No sample was excluded.