LYAPUNOV EXPONENTS ON NEURAL NETWORKS AS A MEASURE FOR WEIGHT SENSITIVITY

Anonymous authors
Paper under double-blind review

ABSTRACT

Weight vulnerability is a major concern for safety and stability of modern machine learning structures such as Large Language Model (LLM) and Deep Neural Networks (DNN) . In this study, we explore weight specific weight sensitivity using Multilayer Perceptron (MLP) as a toy model. Despite various measures of weight sensitivity have been defined previously, most of the measures and analytical formula derived are not weight-specific. However, modern attacks such as Jailbreaking bit-flips or weight scaling on trained LLM or DNN are weight-specific. This study aims to define weight-specific weight sensitivity measures that quantify the sensitivity of neural network output to weight perturbations. This goal is achieved by treating MLP as a dynamical system and considering Lyapunov exponents, which is a stability measure of how fast a perturbation in the initial conditions of a dynamical system propagates through time. In this study, Lyapunov Exponent of a MLP is defined by treating MLP as a discrete dynamical system in two ways : (1) treating each layer of MLP as a vector on a trajectory at one time step; (2) treating the entire feedforward MLP as a discrete map.

1 INTRODUCTION

1.1 WEIGHT SENSITIVITY DEFINITIONS

Weight sensitivity of trained Multi-Layer Perceptron (MLP) have been an area of research since the 1990s. The early approaches focus on MLP with differentiable activation functions for small size perturbations. Various measures have been developed for weight sensitivity for specified input or for general input. The perturbation applied is usually to all weights, and usually there are some specified statistical distribution such as Gaussian i.i.d drawn noise with zero mean and specified variance. One example of definition of weight sensitivity is:

$$S^p(W^*) := \lim_{\sigma \to 0} \frac{\sqrt{\mathrm{var}[\Delta X_p(L)]}}{\sigma},$$

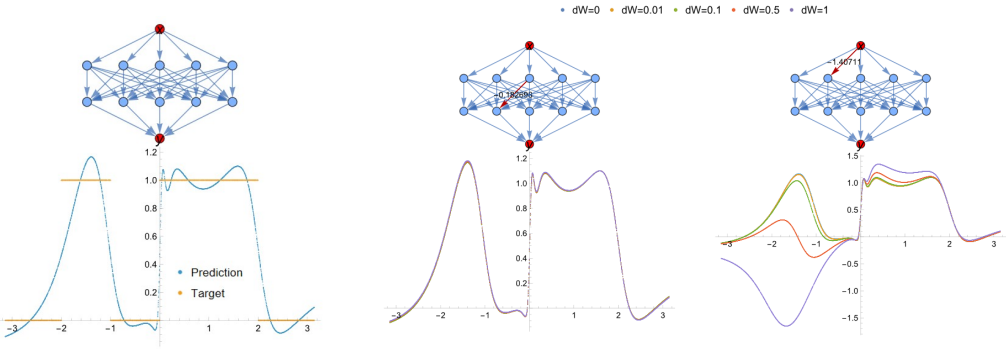
where $\Delta X_p(L)$ is the output error, σ is the standard deviation of each component of ΔW, the additive perturbation applied on all weights, with components in $\Delta W \sim \mathscr{N}(0, \sigma)$ and $\mathrm{var}[\cdot]$ is the variance of $[\cdot]$ Choi & Choi (1992). With such definitions, analytical formulas for weight sensitivity are derived. More recently, weight sensitivity has been studied for modern architecture of neural networks, such as the convolutional neural nets. Various definitions of weight sensitivity is introduced, such as neuron sensitivity and layer sensitivity Xiang et al. (2019), However, the weight sensitivity has rarely been defined in a weight-specific, nor perturbation-specific manner.

1.2 WHY DO WE CARE ABOUT WEIGHT-SPECIFIC WEIGHT SENSITIVITY?

Despite weight-sensitivity are rarely studied in a weight-specific nor perturbation specific manner, attacks of modern neural network structures ranging from Deep neural net (DNN) Hailesellasie et al. (2019) to Large Language Model (LLM) Coalson et al. (2025) have been designed in a weight-specific manner. Hailesellasie et al. (2019) developed a type of attack where the attacker only has access to weight memory, and it systematically perturb one weight at a time by scaling to find the sets of weights most influential to accuracy of the output. Coalson et al. (2025) has designed a bit-flip attack for which 5 to 25 bit-flips can break the refusal of billion parameter LLM models to generate harmful

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

response. The weight-specificity and the input-specificity of output response to weight perturbation can be seen in simple Multilayer Perceptron (MLP) models. Figure 1 shows that even with a simple task of training a double hidden-layer, 5-node MLP, with $tanh(\cdot)$ as activation function to fit a square-wave function, the effect of weight-specific sensitivity can be demonstrated. Perturbing the weight highlighted in left part of Fig. 1 (b) has seemingly no effect on performance, while perturbing the weight highlighted in right part of (b) has more complex effects on performance. When input is negative and $dW = 0.01, 0.1, 0.5$, the prediction in right part of (b) is reducing but still positive, which is in the range of the target function, and when $dW = 1$, the prediction dropped outside of the range of the target function. Perturbing the same weight while input is positive, the change of performance is less pronounced. This input-dependent and perturbation-size dependent response is similar to the "trigger word"-dependent weight-pollution attacks of more modern ML structures Li et al. (2024).



(a) MLP setup and comparison of Prediction and Target

(b) Effect of perturbing the weight highlighted in red.

Figure 1: Effects of additive perturbation of individual trained weights on Prediction Performance.

1.3 Lyapunov Exponents

Lyapunov exponent is a measure developed to quantify the sensitivity of a dynamical system to perturbations in initial conditions. It quantifies how a chaotic attractor stretches and contracts. For a discrete map $\boldsymbol{M}$, it is defined by Ott (2002) in the following way:

Definition 1.1 (Lyapunov Exponent)**.** Let $\vec{x}_0$ be a initial condition and $\boldsymbol{x_n}$ where $n \in \mathbb{N}$ be the corresponding orbit, and $\vec{v}_0$ be the infinitesimal perturbation from $\boldsymbol{x_0}$, then

$$\vec{v}_{n+1} = \boldsymbol{DM}(\vec{x}_n) \cdot \vec{v}_n$$

where $\boldsymbol{DM}$ denotes Jacobian matrix of partial derivatives of $\boldsymbol{M}$. For perturbation in direction $\vec{u}_0 = \frac{\vec{v}_0}{|\vec{v}_0|}$, with $|\cdot|$ denoting vector norm, the Lyapunov exponent (LE) λ is

$$\begin{aligned}\lambda(\vec{x}_0, \vec{u}_0) &= \lim_{n\to\infty} \frac{1}{n} \ln(|\vec{v}_n|/|\vec{v}_0|) \\ &= \lim_{n\to\infty} \frac{1}{n} \ln |\boldsymbol{DM}^n(\vec{x}_0) \cdot \vec{u}_0|,\end{aligned}$$

and the Finite-Time Lyapunov Exponent (FTLE) λ_{FT} is

$$\begin{aligned}\lambda_{FT}(\vec{x}_0, \vec{u}_0) &= \frac{1}{n} \ln(|\vec{v}_n|/|\vec{v}_0|) \\ &= \frac{1}{n} \ln |\boldsymbol{DM}^n(\vec{x}_0) \cdot \vec{u}_0|.\end{aligned}$$

1.4 Related Works

Neural networks can be seen as dynamical systems in the three ways according to Chemnitz et al. (2026): (1) treating each hidden layer as a time-slice and increasing depth as a progression in time; (2) let the neural network internal states evolves recurrently with input like in the case of Neural ODE; (3) consider the training dynamics of the stochastic gradient descent (SGD). In all three cases, Lyapunov exponents can be defined. The finite time maximal Lyapunov exponents (FTMLE) of a MLP has been utilized to explore the input sensitivity. Kondo et al. (2021) has found out that FTMLE is higher for inputs close to the decision boundaries of classification tasks than those that are further away. Thus, the FTMLE can be used to determine the difficulty of classification for a specific input. Storm et al. (2024) also investigated a similar definition of FTMLE for Deep neural network input sensitivity. Qiao et al. (2023) used MLP to predict Henon map and Lorenz-63 attractors and estimated the maximal Lyapunov exponents of the predicted trajectories.

2 Theoretical Definitions

For this study, the notations are defined in the table below:

Table 1: Notations for MLP weights, nodes, and outputs

Notation	Definition
$W^{\beta,l}_{\alpha,j}$	Weight connecting node j in layer α to node l in layer β
$W^{\beta,l}_{0,\aleph}$	Weight connecting the $\aleph$th input to node l in layer β
$\Delta W^{\beta,l}_{\alpha,j}$	Perturbation added to the weight $W^{\beta,l}_{\alpha,j}$
$\Delta W^{\beta,l}_{0,\aleph}$	Perturbation added to the weight $W^{\beta,l}_{0,\aleph}$
$W^{L+1,\aleph}_{\alpha,m}$	Weight connecting node m in layer α to output $y_\aleph$
$\Delta W^{L+1,\aleph}_{\alpha,m}$	Perturbation added to the weight $W^{L+1,\aleph}_{\alpha,m}$
$b_{L+1,\aleph}$	Bias of output $y_\aleph$
$b_{\beta,l}$	Bias added to node l in layer β
$N_{\beta,l}$	Node l in layer β
$\Delta N_{\beta,l}$	Change in the value of node $N_{\beta,l}$ caused by the weight perturbation
α, β	Layer indexes ranging from $\{1, 2, \cdots, L\}$. L is the number of hidden layers.
l, j, m	Node indexes ranging from $\{1, 2, \cdots, P\}$
$\aleph$	input/output indexes ranging from $\{1, 2, \cdots, d\}$, where d is the dimension for input/output
$x_\aleph$	$\aleph$'s component of input of the MLP
$y_\aleph$	$\aleph$'s component of output of the MLP
$\Delta y_\aleph$	Change in the output $y_\aleph$ caused by weight perturbation
σ	Activation function that is applied element-wise

The internal structure of a MLP, where nodes in each adjacent layers are fully connected, is very similar to a discrete map system, if we treat the layer depth as time progression, the only difference is that the map will change from one time step to the next due to the weights are different at each layer or time step. Layers, input, output of the MLP studied are related by following equations:

$$N_{1,l} = \sigma\left(W^{1,l}_{0,\aleph} x_\aleph\right) + b_{1,l}, \tag{1}$$

$$N_{\alpha+1,l} = \sigma\left(W^{\alpha,m}_{\alpha+1,l} N_{\alpha,m}\right) + b_{\alpha+1,l}, \tag{2}$$

$$y_\aleph = \sigma\left(W^{L+1,\aleph}_{\alpha,m} N_{\alpha,m}\right) + b_{L+1,\aleph}. \tag{3}$$

Each layer of nodes $N_{\alpha,l}$ can be considered as a vector in the time slice. Then, when the weight $W^{\alpha,m}_{\alpha+1,l}$ is additively perturbed, a perturbation of node $N_{\alpha,m}$ is caused, and this perturbation propagates through the layers after that. Thus, a Finite-time Lyapunov Exponents can be defined in the following way:

Definition 2.1 (Finite-time Lyapunov Exponent treating depth as time progression)**.** When $W^{\alpha,l}_{\alpha+1,j}$ is perturbed by $\Delta W^{\alpha,l}_{\alpha+1,j}$, or $W^{1,l}_{0,\aleph}$ is perturbed by $\Delta W^{1,l}_{0,\aleph}$ the node after perturbation $\tilde{N}_{\alpha,l} = N_{\alpha,l} + \Delta N_{\alpha,l}$.

$$h_{FT}\left(\vec{N}_\alpha, \Delta\vec{N}_\alpha\right) = \frac{1}{(L-\alpha+1)} \ln\left(\left|\Delta\vec{N}_L\right| / \left|\Delta\vec{N}_\alpha\right|\right) \tag{4}$$

Where $\vec{N}_\alpha$ stores the values for all the nodes in layer α, $\Delta\vec{N}_\alpha$ stores all elements of $\Delta N_{\alpha,l}$. Notice that we are not addressing the weights connecting to output, because we are focusing on the weights internal to MLP.

The MLP can also be treated as a d dimensional map, since in this case it takes a d dimensional vector and outputs a d dimensional vector. With that, we can define the same Lyapunov Exponents and look at how changing each weight affects the λ in each case.

3 Experimental Methods and Results

3.1 Task 1: Predicting a Square wave using a L-hidden layer MLP

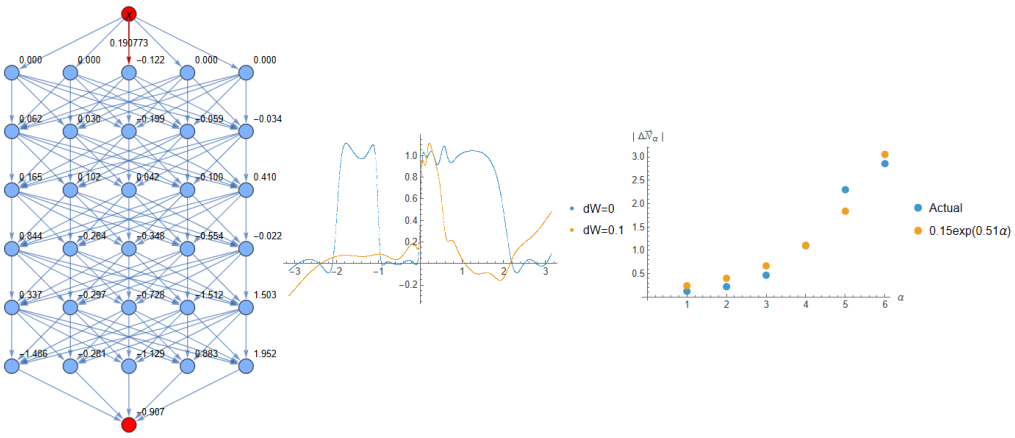


Figure 2: Demonstration of process of calculating the Finite time Lyapunov time using a 6-hiddenlayer MLP, after the highlighted weight is perturbed by adding 0.1, and the input $x = -1.46$

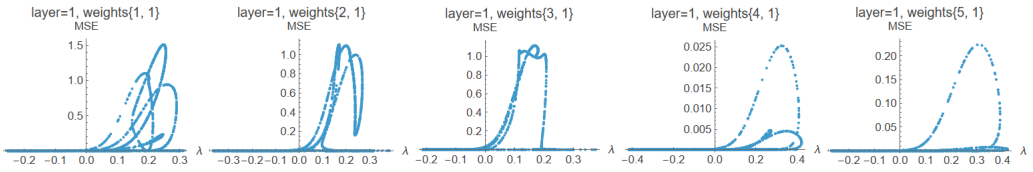


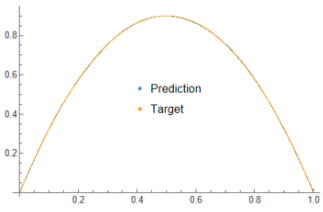
Figure 3: FTLEs vs MSE between output of trained 10-hidden layer MLP to the same trained MLP with all weights, but the weight specified to be perturbed by adding 0.1, held fixed. Since the input and output are one-dimensional, the title layer 1 weight $\{1, i\}$ means $W^{1,i}_{0,1}$ is perturbed

The task we used to calculate the finite time Lyapunov time as described in Definition 2.1 is prediction of square wave in Fig. 1 . The The MLP is first trained by the function NetTrain in Mathematica using the $Adam$ method. Then, one weight is perturbed by adding dW to simulate weight-specific bitflip attacks. Then for all x in the testing data set, the $\Delta\vec{N}_\alpha$ is calculated at each layers including and after the node directly connected to the perturbed weight. The FTLE is obtained by fitting $z = a\exp(bu)$, with $u = \alpha$, and $z = \Delta\vec{N}_\alpha$. This process is demonstrated in Fig. 2. Figure 3 shows that the maximum MSE comparing the output of the perturbed neural net and the unperturbed trained is larger when the FTLE is larger. However, the relation is not purely proportional. In addition

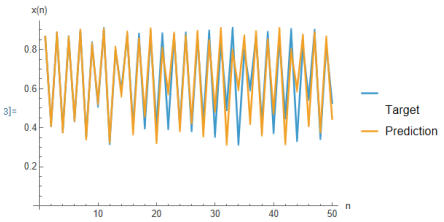
to that, there are cases where calculated FTLE is positive, but the MSE is minimal. Also, some of the fit for FTLE did not converge to desired precision under limited computing time.

3.2 Task 2: Training to predict Logistic map at $r = 3.6$

The task for investigation of effect of additive weight perturbation on Lyapunov exponent as specified in 1.1 is prediction of one-dimensional Logistic map. In training MLP feed forward to learn Logistic map and to predict the chaotic trajectories is a toy model task to simulate effect of weight perturbation on other recurrent structures such as reservoir computer or Long Short-Term Memory (LSTM) network Gers et al. (2001) for prediction of complex time-series.



(a) Logistic Map $M(x)$ vs x prediction from a 2-hiddenlayer MLP and target trajectory



(b) Logistic Map serial prediction from a 2-hiddenlayer MLP and target trajectory

Figure 4: Effects of additive perturbation of individual trained weights on Prediction Performance.

The Logistic map is specified by

$$x_{n+1} = r x_n (1 - x_n), \tag{5}$$

as shown in Fig. 4 is chaotic with a Lyapunov Exponent of around 0.183 at $r = 3.6$. The MLP learned the relation shows in Fig. 4 a), and then the prediction is done by serially predicting x_{n+1} from x_n starting from x_0. The x_0 is obtained by the output of first running the $r = 3.6$ Logistic Map by 1000 steps to avoid transient effects. The same is done for Fig. 5 for the trained MLP with specified weight perturbation. In Fig. 6, the Lyapunov exponents as specified in Definition 1.1 is calculated for a range of dW.

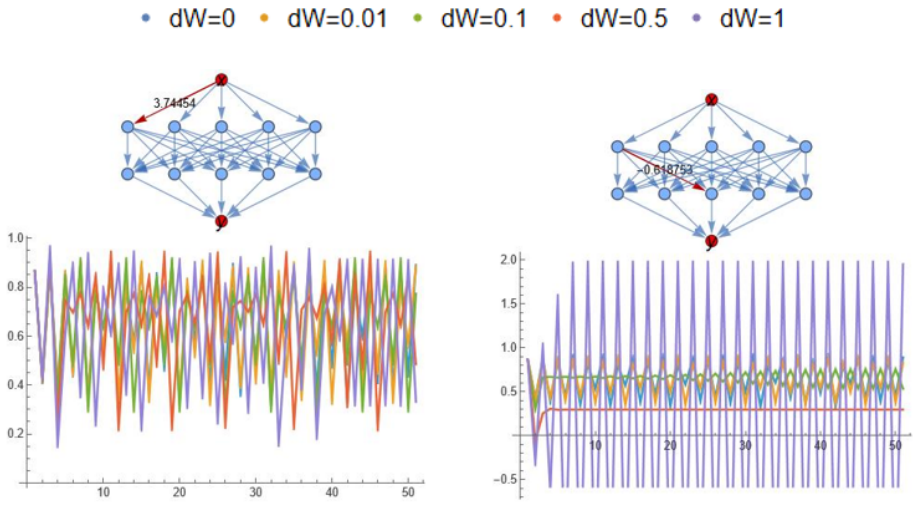


Figure 5: Trajectory produced for trained 2-hiddenlayer MLP with the weight highlighted in red and magnitude labeled perturbed by dW.

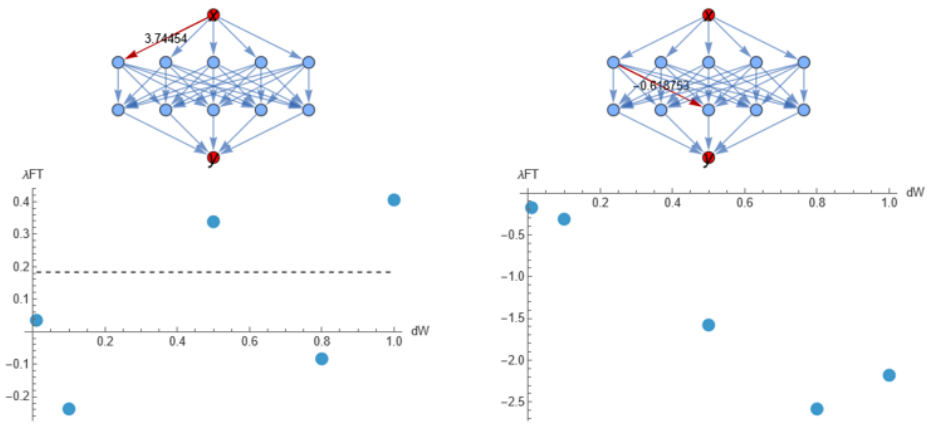
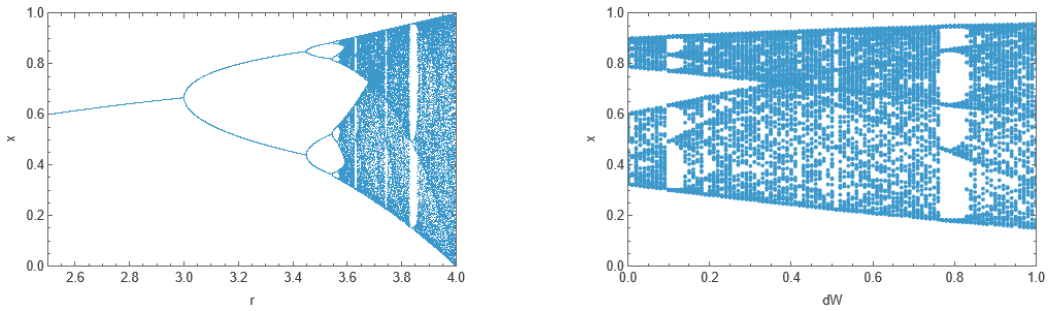


Figure 6: Finite Time Lyapunov time as a function dW, when the weight highlighted in red and magnitude labeled is perturbed by adding dW.

Fig. 5 and Fig 6 demonstrated that by additive perturbation, the trajectory of the predicted Logistic Map show behavior similar to bifurcation, in which the dynamics of the map changes from being chaotic to being periodic and to chaotic by varying parameter of weight highlighted. There are also large sensitivity to weight perturbation for LE. The LE predicted by trained MLP drops from 0.1972, which is close to the real Logistic map of 0.183 to a value close to zero, when only adding 0.01 to a weight of size around 3.74. This bifurcation-like behavior is further shown by the bifurcation diagram obtained by methods specified in Ott (2002).



(a) Logistic Map orbits as a function of r obtained using a step size of 0.001 for r in the range of $[2.5, 4]$.

(b) Logistic Map serial prediction orbits from a 2-hiddenlayer MLP as a function of dW obtained using a step size of 0.01 for dW in the range [0,1].The single weight perturbed is $W_{0,1}^{1,1}$.

Figure 7: Effects of additive perturbation of individual trained weights on Prediction Performance.

Figure 7 (b) shows that when the weight $W_{0,1}^{1,1}$ is perturbed, a behavior similar to period-doubling bifurcation shown in Fig. 7 (a) for $r > 3$ is appearing. This means that ergodic property of the predicted trajectory can change by flipping one bit on a single weight.

4 Discussion and Future Works

In this work, the effect of additive weight perturbation or bit flip on MLP Lyapunov Exponents and finite-time Lyapunov Exponent are investigated. The internal FTLE tends to be positive, when the deviation from the trained output is large. The MLP seems to have period-doubling bifurcation

as we vary its weight dW. Despite when internal MLP weights are perturbed, there seems to be changes in both the internal FTLE and the external LE, a mechanism for the change is not understood. Derivation of an analytical theory for such LE and FTLE for specified trained weights is needed for future studies. Furthermore, this study only considered MLP in which each hiddenlayer has the same number of nodes. The effect of network structures of different geometries needs to be studied.

ACKNOWLEDGMENTS

Thanks for John Doe's inspiration of defining a Lyapunov exponent to investigate weight sensitivity, and thanks John Doe for code help.

REPRODUCIBILITY STATEMENT

To ensure reproducibility, codes that is needed to reproduce all the figures included in the pdf is included in the zipped folder in supplementary material.

AI DISCLOSURE

GPT-5.6 Luna and Gemini are utilized for literature search and Latex formatting, as well as writing the function MLPGraph for visualizing MLP structure, and other syntax-related debug problems. All codes are checked and understood by the author.

REFERENCES

Dennis Chemnitz, Maximilian Engel, Christian Kuehn, and Sara-Viola Kuntz. A dynamical systems perspective on the analysis of neural networks, 2026. URL https://arxiv.org/abs/2507.05164.

Jin Young Choi and Chong-Ho Choi. Sensitivity analysis of multilayer perceptron with differentiable activation functions. *IEEE Transactions on Neural Networks*, 3(1):101–107, 1992. doi: 10.1109/72.105422.

Zachary Coalson, Jeonghyun Woo, Chris S. Lin, Joyce Qu, Yu Sun, Shiyang Chen, Lishan Yang, Gururaj Saileshwar, Prashant Nair, Bo Fang, and Sanghyun Hong. Prisonbreak: Jailbreaking large language models with at most twenty-five targeted bit-flips, 2025. URL https://arxiv.org/abs/2412.07192.

Felix A. Gers, Douglas Eck, and Jürgen Schmidhuber. Applying lstm to time series predictable through time-window approaches. In Georg Dorffner, Horst Bischof, and Kurt Hornik (eds.), *Artificial Neural Networks — ICANN 2001*, pp. 669–676, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg. ISBN 978-3-540-44668-2.

Muluken Hailesellasie, Jacob Nelson, Faiq Khalid, and Syed Rafay Hasan. Vaws: Vulnerability analysis of neural networks using weight sensitivity. In *2019 IEEE 62nd International Midwest Symposium on Circuits and Systems (MWSCAS)*, pp. 650–653, 2019. doi: 10.1109/MWSCAS.2019.8884977.

Misaki Kondo, Satoshi Sunada, and Tomoaki Niiyama. Lyapunov exponent analysis for multilayer neural networks. *Nonlinear Theory and Its Applications, IEICE*, 12(4):674–684, 2021. doi: 10.1587/nolta.12.674.

Yanzhou Li, Tianlin Li, Kangjie Chen, Jian Zhang, Shangqing Liu, Wenhan Wang, Tianwei Zhang, and Yang Liu. Badedit: Backdooring large language models by model editing. In *International Conference on Learning Representations*, volume 2024, pp. 26117–26134, 2024.

Edward Ott. *Chaos in Dynamical Systems*. Cambridge University Press, 2nd edition, 2002.

M. Qiao, Y. Liang, A. Tavares, and X. Shi. Multilayer perceptron network optimization for chaotic time series modeling. *Entropy*, 25(7):973, June 2023. doi: 10.3390/e25070973.

L. Storm, H. Linander, J. Bec, K. Gustavsson, and B. Mehlig. Finite-time lyapunov exponents of deep neural networks. *Phys. Rev. Lett.*, 132:057301, Feb 2024. doi: 10.1103/PhysRevLett.132.057301. URL https://link.aps.org/doi/10.1103/PhysRevLett.132.057301.

Lin Xiang, Xiaoqin Zeng, Yuhu Niu, and Yanjun Liu. Study of sensitivity to weight perturbation for convolution neural network. *IEEE Access*, 7:93898–93908, 2019. doi: 10.1109/ACCESS.2019.2926768.