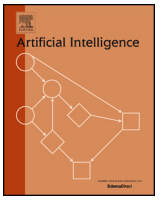
Artificial Intelligence 327 (2024) 104060

Contents lists available at ScienceDirect

Artificial Intelligence

journal homepage: www.elsevier.com/locate/artint



Fragility, robustness and antifragility in deep learning

Chandresh Pravin [a], Ivan Martino [b], Giuseppe Nicosia [c,d], Varun Ojha [e,*]

[a] University of Reading, United Kingdom
[b] KTH Royal Institute of Technology, Sweden
[c] University of Catania, Italy
[d] University of Cambridge, United Kingdom
[e] Newcastle University, United Kingdom

ARTICLE INFO

Keywords:
Deep neural networks
Robustness analysis
Adversarial attacks
Adversarial robustness
Adversarial training
Network sparsity

ABSTRACT

We propose a systematic analysis of deep neural networks (DNNs) based on a signal processing technique for network parameter removal, in the form of *synaptic filters* that identifies the *fragility*, *robustness* and *antifragility* characteristics of DNN parameters. Our proposed analysis investigates if the DNN performance is impacted negatively, invariantly, or positively on both clean and adversarially perturbed test datasets when the DNN undergoes synaptic filtering. We define three *filtering scores* for quantifying the fragility, robustness and antifragility characteristics of DNN parameters based on the performances for (i) clean dataset, (ii) adversarial dataset, and (iii) the difference in performances of clean and adversarial datasets. We validate the proposed systematic analysis on ResNet-18, ResNet-50, SqueezeNet-v1.1 and ShuffleNet V2 x1.0 network architectures for MNIST, CIFAR10 and Tiny ImageNet datasets. The filtering scores, for a given network architecture, identify network parameters that are *invariant in characteristics* across different datasets over learning epochs. Vice-versa, for a given dataset, the filtering scores identify the parameters that are invariant in characteristics across different network architectures. We show that our synaptic filtering method improves the test accuracy of ResNet and ShuffleNet models on adversarial dataset when only the robust and antifragile parameters are selectively retrained at any given epoch, thus demonstrating applications of the proposed strategy in improving model robustness.

1. Introduction

Deep neural networks (DNNs) are extensively used in various tasks and domains, achieving noteworthy performances in both research and real-world applications [1,2]. It is the critical weaknesses of DNNs, however, that warrant investigation if we are to better understand how they learn abstract relationships between inputs and outputs [3,4]. We propose to investigate the effects of a *systematic analysis* on DNNs by using a signal processing technique for network parameter filtering (the terms DNN and network are used interchangeably), in contrast to random filtering [5–7] methods.

Our work analyzes the performance of a DNN under (a) *internal stress* (i.e., the synaptic filtering of DNN parameters) and (b) *external stress* (i.e., perturbations of inputs to the DNN). We define internal and external stress within the context of DNNs as a novel concept taking inspiration from the applications of stress on biological systems [8]. Through analyzing the performance of

* Corresponding author.
E-mail address: varun.ojha@newcastle.ac.uk (V. Ojha).

https://doi.org/10.1016/j.artint.2023.104060
Received 16 September 2022; Received in revised form 29 March 2023; Accepted 15 December 2023

Available online 19 December 2023
0004-3702/© 2023 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

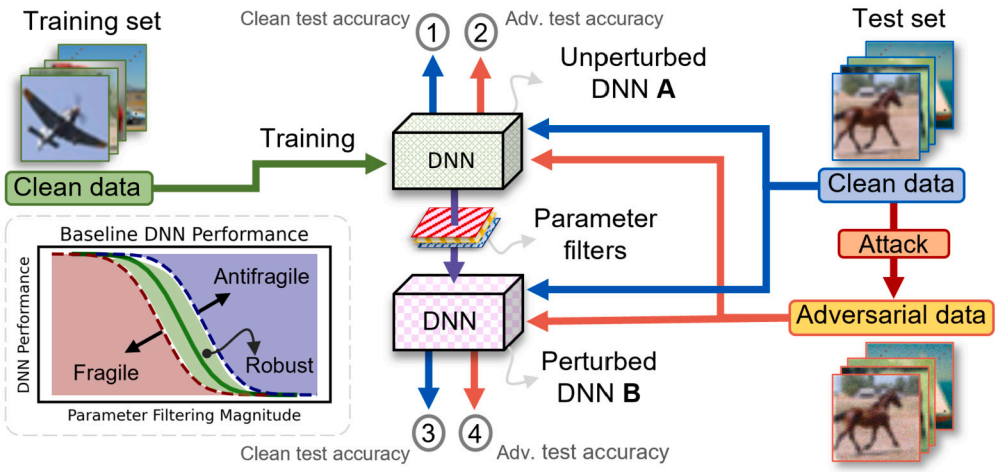


Fig. 1. Our methodology of parameter filtering and evaluating DNN performances on clean and adversarial datasets. Passing a DNN through parameter filters is equivalent to internal stress and applying an adversarial attack with various magnitudes on clean data is equivalent to external stress on a DNN. In this methodology, the DNN performances (labelled 1, 2, 3, and 4) are individually compared against a defined baseline DNN performance (solid green curve in the illustration shown on the lower left) in order to characterize DNN parameters as fragile (red shaded area), robust (green shaded area), or antifragile (blue shaded area). (For interpretation of the colours in the figure(s), the reader is referred to the web version of this article.)

a network to input perturbations (external stress) formed using an adversarial attack [9,10], we bring the weakness of the DNN to the foreground. We simultaneously apply synaptic filtering (internal stress) to the network parameters in order to identify the specific parameters most susceptible to the input perturbations, thus characterizing them as *fragile*. Similarly, we identify parameters of the DNN that are *invariant* to both internal and external stress when considering the network performance, thus characterizing them as *robust* to the applied stress. Following this reasoning, we introduce a novel notion of *antifragility* [11] in deep learning as the circumstance in which any applied perturbations (internal and external) on a network result in an improvement of the network performance.

When considering external stress, such as variations to the network input, we focus our analysis specifically on varying magnitudes of adversarial attack perturbations [9,10] due to their ability to exploit the learned representations of a network to decrease network performance [12]. In our study, we focus on the fast gradient sign method (FGSM) attack for its equal single-step perturbation calculation for increasing network loss [13]. Our synaptic filtering methodology (see Fig. 1) offers a comparative study of state-of-the-art DNNs using clean and adversarially perturbed datasets, and therefore, the study is relevant for any variation of perturbation introduced to the input space. We apply our methodology to expose the fragility, robustness and antifragility of network parameters over network learning epochs, which subsequently enables us to examine the *learning landscape* (performance variations over epochs) of the network learning process.

In order to better understand how an adversarial attack is effective in bringing a network to failure [14], we take a novel methodology that considers network susceptibility to adversarial perturbations in conjunction with network architecture and the learning processes (see Fig. 1). The proposed synaptic filters are considered to be the *lenses* under which we can *characterize parameters* of network architecture. Introducing an adversarial attack to the methodology in Fig. 1 offers a unique insight into how the characterization of network parameters varies between clean and adversarial inputs. We validated the analysis on the ResNet-18, ResNet-50 [15], SqueezeNet-v1.1 [16], and ShuffleNet V2 x1.0 [17] networks for the MNIST [18], CIFAR10 [19] and the Tiny ImageNet datasets [20].

Our main contributions of this work, therefore, are as follows:

- We offer a novel methodology based on signal processing techniques that apply internal stress (parameter removal) and external stress (adversarial attack) on DNNs to characterize the network parameters as either fragile, robust, or antifragile.
- We offer parametric filtering scores that use a defined *baseline network performance* to quantify the influence of specific parameters on the network performance.
- We apply internal stress on networks in the form of synaptic filters and use the filtered network performances to show that networks trained on different datasets contain parameter characterizations that are *invariant* to different datasets throughout the network training process.
- We apply external stress to networks, in the form of an adversarial attack, to identify the *specific parameters* targeted by the adversary through a comparison of the synaptic filtering performances of the clean and adversarial test datasets.
- We show that our synaptic filtering method boosts the test accuracy of ResNet and ShuffleNet models on adversarial dataset when only the robust and antifragile parameters are retrained at any given epoch, thus proving a useful strategy for improving network robustness.

The following Sec. 2 gives insights into the background and related works. Section 3 offers definitions of the terms and concepts introduced in the proposed methodology. Section 4 reports the proposed methodologies. Section 5 shows the experimental results acquired using the proposed methodologies, and Sec. 6 concludes the work.

2. Background and related work

We propose evaluating the resilience of DNNs using a physiologically inspired approach concerning the resilience of humans to stress on their physiology [8,21]. Therefore, we analyze the performance of DNNs to internal and external stress. Within the context of deep learning, we consider internal stress to be the perturbations to the network parameters (i.e., synaptic filtering) [22,23] and we take external stress to be variations to the learning environment of the network (i.e., input perturbations) [24–26].

There exist various avenues of research when considering an analysis of DNNs to input perturbations [13,27] and synaptic filtering [23,6]. The works of Szegedy et al. [9] and Goodfellow et al. [13] invited attention to investigate the vulnerability of DNNs to a particular method of crafting input perturbations in the form of adversarial attacks. The rapid development of new adversarial attacks [28] and equally abundant adversarial defence techniques [29,30], call for methods of analyzing the resilience of DNNs to carefully crafted input perturbations, designed to bring networks to failure.

The scrutiny of DNN resilience to these perturbations can be expanded to incorporate perturbations into network architectures. The study proposed by Han et al. [31] details how network parameters can be filtered out to reduce network size, without significantly affecting network performance. However, there may be conditions when filtering parameters may lead to improvements in the network performance. Therefore, we use a notion of antifragility to describe an increase in network performance whilst being subjected to internal and/or external stress, in the form of synaptic filters [7,23,6] and adversarial attacks [14,28,30]. Our notion of antifragility in DNN is in line with the antifragility notion described by Taleb and Douady [11] to refer to a phenomenon whereby a system subjected to stress shows to improve in performance. We describe the related works on internal and external stress as follows:

Internal stress (parameter filtering) Network architecture affects how and what DNNs learn [32–35]. Therefore, the works of Ilyas et al. [36] highlight the presence of robust and non-robust features within networks. In a similar context, we highlight the presence of fragile, robust and antifragile parameters of different network architectures on both clean and adversarial test datasets. For the characterization of the network parameters, we propose a synaptic filtering methodology (see Fig. 1).

Identifying fragile, robust and antifragile parameters informs us about the *compressibility* (network sparsity) of a network based on the variation and degradation in the network performance [37]. A central principle of network compression techniques is to reduce network size whilst retaining network performance [31]. A method of achieving network compression is through using network pruning techniques [38,23,6]. Our work of parameter filtering differs from the objective of pruning techniques that aim to reduce DNN size, whereas we aim to analyze the characteristics of DNN parameters by systematically filtering them. As well as our works differ from those systematic tuning of DNN hyperparameters such as the number of layers and number of neurons in a layer to analyze the DNN performance [39], i.e., we systematically internal architecture of the DNN.

Siraj et al. [40] proposed a robust sparse regularisation method for network compactness while simultaneously optimizing network robustness to adversarial attacks. Similarly, we use our synaptic filtering methodology (a network parameter removal technique) to study the performances of a DNN on clean and adversarial datasets, which enable us to identify parameters that cause a decrease in network performance on the adversarial dataset [41] compared to the clean dataset, thus characterized as fragile in our work. We characterize parameters that are invariant to synaptic filtering on both clean and adversarial datasets as robust. Whereas the parameters that, when filtered, show to increase the network performance on the adversarial dataset compared to the clean dataset are characterized as antifragile.

External stress (adversarial attacks) There are numerous methods for computing adversarial attacks on DNNs in the literature [25,26]. The primary objective of adversarial attacks is to deceive a network into misclassifying otherwise correctly classified inputs [9,13]. The importance of the analysis of adversarial attacks on DNNs is significant due to the existence of adversarial examples in real world applications [42,43]. Similarly, in our work, we analyze the adversarial attack in order to characterize network parameters into the parameters that affect network performance negatively (fragile), invariantly (robust), and positively (antifragile). Adversarial examples are by design created to decrease network performance; however, when simultaneously carrying out synaptic filtering methods [41] it is possible to observe an increase in network performance, even under an adversarial attack, thus requiring the notion of antifragility.

3. Definitions

In this Section, we define *fragility*, *robustness*, and *antifragility* within the scope of DNNs. For defining fragility, robustness, and antifragility, we also need to define the *internal stress*, *external stress* and *baseline network performance* of DNNs. Here the stress on a DNN is a systematic perturbation, either internal (synaptic filtering) or external (adversarial attack). The purpose of applying the stress on DNN is to test the operating conditions of the DNN for both learned and optimized states, when evaluated on unseen datasets. The concepts of network fragility, robustness, antifragility and stress are shown in Fig. 2, where Fig. 2a shows the application of stress on a DNN and Fig. 2b shows the interpretation of DNN performance for parameter characterization. For detailed definitions of the above-mentioned concepts, we consider the following notations.

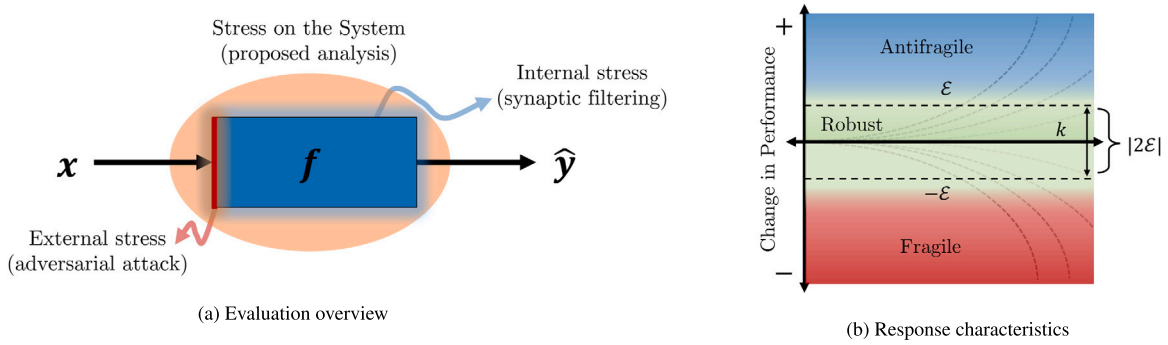


Fig. 2. (a) Overview of the proposed system evaluation method. (b) Characteristics of fragility, robustness and antifragility through analysing the performance of a system $\mathcal{F}$ whilst under stress.

Consider a neural network *architecture* as a set of functions $f(x,\cdot)$ that consists of a configuration of parameters, such as convolutions, batch normalization, pooling layers, activation functions, etc. [38], we define a *parameterized* neural network as $f(x,\mathbf{W})$, for specific parameters $\mathbf{W}$ and input x. For an l-layer network with a d dimensional input $x \in \mathbb{R}^d$; the K-class classification function is thus $f : \mathbb{R}^d \to \mathbb{R}^K$. The prediction of $f(x,\mathbf{W})$ is given by $\hat{y} = \arg\max_{1\le k\le K} f_k(x,\mathbf{W})$. The network parameters $\mathbf{W}$ are assumed to be optimized, either partially or fully, using back-propagation and a loss function $\mathcal{L} : \mathbb{R}\times\mathbb{R} \to \mathbb{R}$ given by $\mathcal{L}(\hat{y}, y)$ to calculate the network error.

3.1. Stress on DNNs

To formulate internal stress on the network, we consider two filtering domains: *local* (the parameters of any specific layer) and *global* (the parameters of the whole network). We apply synaptic filtering to filter the parameters of *trainable convolutional layers* and *fully connected layers* of the network, the non-trainable parameters, however, remain unaffected by the synaptic filtering procedure. The l-th layer network parameters (local parameters) are given as $\mathbf{W}^{(l)}$, while the global network parameters are $\mathbf{W}$. For convenience, we denote the network parameters to be evaluated by the synaptic filtering methods as θ, where $\theta = \mathbf{W}^{(l)}$ is the local parameter analysis [31] and $\theta = \mathbf{W}$ is the global parameter analysis, as mentioned in [44].

Definition 1 *(Synaptic filtering).* The synaptic filtering involves taking a network $f(x,\theta)$ with parameter θ as an input and producing a filtered network $f(x,\tilde{\theta}_\alpha)$ with filtered parameter $\tilde{\theta}_\alpha$ as:

$$\tilde{\theta}_\alpha = B_\alpha \odot \theta_\alpha, \quad B_\alpha \in \{0,1\}^{|\theta_\alpha|}, \tag{1}$$

where $\alpha = \{\alpha_0, \alpha_1, \ldots, \alpha_A\}$ is the normalised synaptic filtering thresholds across the complete parameter range of the evaluated network/layer with a lower bound $\alpha_0 = 0$, upper bound $\alpha_A = 1$ and step size Δ_α given by $\alpha_1 = \alpha_0 + \Delta_\alpha$. For synaptic filtering of a network, we have $\hat{y} = f(x,\theta)$ as the network predictions for the unperturbed network and $\hat{y}_\alpha = f(x,\tilde{\theta}_\alpha)$ as the network predictions for the perturbed network. In Eq. (1), B_α is a binary mask for a threshold α that filters parameters, θ_α are the set of parameters to be filtered that may be different to θ, and $\odot$ is the element-wise product operator

To further constrain the internal stress analysis, we define that the network parameters to be filtered θ is not a zero vector prior to the synaptic filtering, i.e., θ must be a trained network: $\theta \neq \mathbf{0}$. If this constraint is not met, the prediction of the network $f(x,\theta)$ will result in output values of zero for all inputs.

Definition 2 *(Internal stress - synaptic filtering of the DNN parameters).* The internal stress on a DNN is the application of the synaptic filtering method with various magnitudes of α ranging from a minimum filtering threshold α_0 to the maximum filtering threshold α_A in order to obtain a set of $|\alpha|$ filtered networks S_α:

$$S_\alpha = \{f(x,\tilde{\theta}_{\alpha_0}), f(x,\tilde{\theta}_{\alpha_1}), \ldots, f(x,\tilde{\theta}_{\alpha_A})\}. \tag{2}$$

With evaluating a network to internal stress, we examine how the filtering of learned parameters of a network, influences the network performance, thus identifying the specific *filtering thresholds* required to bring the network to failure.

Considering external stress as variations to the input x, we introduce $x_\epsilon = x + \delta_\epsilon$ as the *perturbed example* of x with an adversarial perturbation $\delta_\epsilon \in \mathbb{R}^d$, where $\epsilon = \{\epsilon_0, \epsilon_1, \ldots, \epsilon_E\}$ is the perturbation magnitude with minimum perturbation magnitude ϵ_0 and maximum perturbation magnitude ϵ_E with step size $\epsilon_1 = \epsilon_0 + \Delta_\epsilon$. Using a single adversarial attack formulation method δ we define $\hat{y} = f(x,\theta)$ as the network predictions on clean dataset and $\hat{y}_\epsilon = f(x_\epsilon,\theta)$ as the network predictions on the adversarial dataset [Fig. 1(Left)]. When dealing with external stress only, θ is taken as the complete set of network parameters $\mathbf{W}$.

The performance of $f(x_\epsilon,\theta)$ can inform us of the ability of the network to remain stable to external stress (input perturbations) applied to the network. This is achieved through a comparison of the network performance on a clean dataset and an adversarially

perturbed dataset. There are numerous variations of δ that can be used to form external stress to the network, from targeting specific features of x to drawing distortions from a different distribution [25,26]. However, in this work, we only focus on one perturbation method δ, i.e., FGSM attack, as our objective is to only compare DNN performance on clean and perturbed inputs (Fig. 1).

When applying external stress with various magnitude ϵ, we get a set of perturbed inputs for the network S_ϵ:

Definition 3 *(External stress - adversarial attack on DNN).* The external stress on a DNN is the application of an adversarial attack with various perturbation magnitudes ϵ ranging from a minimum perturbation magnitude ϵ_0 to the maximum perturbation magnitude ϵ_E in order to obtain a set of $|\epsilon|$ inputs to the network S_ϵ:

$$S_\epsilon = [f(x_{\epsilon_0}, \theta), \ldots, f(x_{\epsilon_E}, \theta)]. \tag{3}$$

With external stress on a network we examine how the variations in the input environment influence the network performance, thus identifying the specific magnitudes of the attack required to bring the network to failure.

An important consideration to make when analyzing networks using internal and external stress in Definitions 2 and 3, is that a resultant perturbed network (S_α and S_ϵ) may offer equal performance to the unperturbed network, i.e., for all inputs x in test set, we observe:

$$p(\hat{y}_\alpha = y | f(x, \tilde{\theta}_\alpha)) \approx p(\hat{y}_\alpha = y | f(x, \theta)) \text{ for internal stress threshold } \alpha, \text{and}$$

$$p(\hat{y}_\epsilon = y | f(x_\epsilon, \theta)) \approx p(\hat{y}_\epsilon = y | f(x, \theta)) \text{ for external stress magnitude } \epsilon,$$

where $p(\cdot)$ is a function that measures the network accuracy over all inputs x. This indicates that even under stress, a DNN may perform equivalently to an unperturbed network. Therefore, in order to evaluate the performance of a network to stress, we must define a baseline network performance against which we can measure the performance of perturbed and unperturbed networks.

A baseline network performance can vary for different types of stress (internal or external), as there may arise instances where the response of the baseline network performance, defined as $\hat{f}(x, \theta_\alpha)$ for internal stress and $\hat{f}(x_\epsilon, \theta)$ for external stress, is not necessarily the same as the performance of the initially trained network (unperturbed network) $f(x, \theta)$. The baseline network performance for a combination of internal and external stress is defined as $\hat{f}(x_\epsilon, \theta_\alpha)$, where the baseline network is a function of ϵ and α.

To give context on why baseline network performance may not necessarily be the same as the performance of an unperturbed network, take the example when we apply internal stress to a DNN, the result is a set of filtered networks S_α. If we define the upper bound of the stress magnitude equal to the total number of network parameters, i.e., $\alpha_A = |\theta|$, then we obtain a network with parameter value zero $\tilde{\theta}_{\alpha_A} = \mathbf{0}$. Noticeably, the performance of a maximally perturbed network $f(x, \tilde{\theta}_{\alpha_A})$ cannot equal to the performance of unperturbed network $f(x, \theta)$, i.e., $f(x, \tilde{\theta}_{\alpha_A}) \neq f(x, \theta)$. Thus we require the baseline network performance to be a function of the magnitude of stress applied on the DNN. A detailed description of baseline network performance is given later in Sec. 4.1.2.

3.2. Fragility, robustness and antifragility

Here we define the three characterizations of network parameters: fragility, robustness and antifragility. In order to define the different characterizations of network parameters, we must establish the stress to which we can evaluate network parameter fragility, robustness and antifragility. The stress in question may be internal (S_α) or external (S_ϵ), or a combination of the two.

For simplicity, we consider only internal network stress for the definitions provided below. However, the change of variables from S_α to S_ϵ, from $\hat{f}(x, \tilde{\theta}_\alpha)$ to $\hat{f}(x_\epsilon, \theta)$, and from Δ_α to Δ_ϵ will give the definition of fragility, robustness and antifragility for external stress.

Definition 4 *(Fragility).* The parameters of a network are fragile if the performance of the networks *decreases* below a threshold $-\varepsilon$, compared to the baseline network performance for all magnitudes of the applied stress. Formally, the fragility to internal stress can be defined as:

$$\sum_{i=0}^{A} [S_{\alpha_i} - \hat{f}(x, \tilde{\theta}_{\alpha_i})] \Delta_\alpha < -\varepsilon, \tag{4}$$

where Δ_α is the change in synaptic filtering threshold α, A is equal to $|\alpha|$, $\varepsilon \geq 0$ and asserts a variable fragility measure, as shown in Fig. 2b (red shaded region). When the threshold $\varepsilon = 0$, we have a strict fragility condition. Equation (4) computes the discrete area difference between the stressed network performance and the baseline network performance for all stress magnitudes of α.

Definition 5 *(Robustness).* The parameters of a network are robust if the performance of the networks is *invariant* to a threshold $\pm\varepsilon$, compared to the baseline network performance for all magnitudes of the applied stress. Formally, the robustness to internal stress can be defined as:

$$-\varepsilon \leq \sum_{i=0}^{A} [S_{\alpha_i} - \hat{f}(x, \tilde{\theta}_{\alpha_i})] \Delta_\alpha \leq \varepsilon, \tag{5}$$

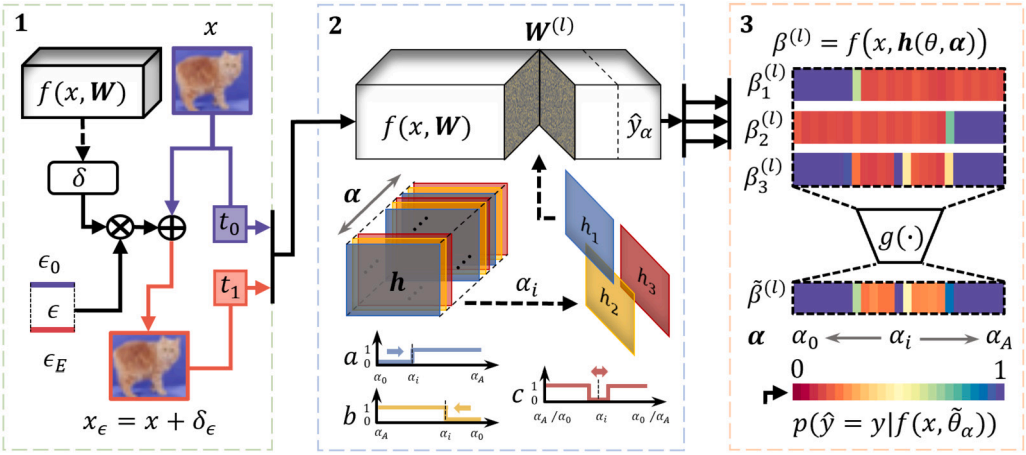


Fig. 3. Synaptic filtering framework. *Left* block (**1**) shows the input x at time t_0; network $f(x, \boldsymbol{W})$ with parameters $\boldsymbol{W}$; the adversarial attack δ [this study computes δ using $f(x, \boldsymbol{W})$]; the perturbation magnitude ϵ and the resultant adversarial example x_ϵ at time t_1. The perturbation magnitude ϵ is bounded by $(\hat{y}_\epsilon \approx \hat{y})$ and $(\hat{y}_\epsilon K > 1)$ for K classes; $\hat{y}$ and $\hat{y}_\epsilon$ are clean and adversarial accuracies. *Middle* block (**2**) outlines the set of synaptic filters $\boldsymbol{h}$, containing h_1, h_2 and h_3 filters at each point α_i applied to layer $\boldsymbol{W}^{(l)}$, resulting in the network performance to the filters. There are A sets of $\boldsymbol{h}$ for each $\alpha_i \in [\alpha_0, \alpha_A]$. *Right* block (**3**) shows $\beta^{(l)} = f(x, \boldsymbol{h}(\theta, \alpha))$ as the system performances for all values of α, where θ is $\boldsymbol{W}^{(l)}$ for a local analysis at layer l. The function $g(\cdot)$ combines $\beta_1^{(l)}, \beta_2^{(l)}$ and $\beta_3^{(l)}$ into a combined system performance $\tilde{\beta}^{(l)}$.

where Δ_α is the change in synaptic filtering threshold α, $\varepsilon \geq 0$ and asserts a variable robustness measure, as shown in Fig. 2b (green shaded region). When the threshold $\varepsilon = 0$, we have a strict robustness condition. Equation (4) computes the discrete area difference between the stressed network performance and the baseline network performance for all stress magnitudes of α.

Definition 6 *(Antifragility).* The parameters of a network are antifragile if the performance of the networks *increases* to a threshold ε, compared to the baseline network performance for all magnitudes of the applied stress. Formally, the antifragility to internal stress can be defined as:

$$\varepsilon < \sum_{i=0}^{A} [S_{\alpha_i} - \hat{f}(x, \tilde{\theta}_{\alpha_i})] \Delta_\alpha, \tag{6}$$

where Δ_α is the change in synaptic filtering threshold α, $\varepsilon \geq 0$ and asserts a variable robustness measure, as shown in Fig. 2b (blue shaded region). When the threshold $\varepsilon = 0$, we have a strict antifragility condition. Equation (4) computes the discrete area difference between the stressed network performance and the baseline network performance for all stress magnitudes of α.

4. DNN parameters characterization methodology

In this Section, we present the methodology of DNN parameter characterization that is shown in Fig. 1. Concisely, Fig. 1 shows that this methodology has two major aspects a) the application of internal and external stress on DNN in terms of synaptic filtering and adversarial attack, and b) the need of a process to characterize parameters into fragile, robust and antifragile. This section first explains how we apply internal and external stress on DNNs in Sec. 4.1 and then introduces parameter scores that characterize the parameters in Sec. 4.2. Finally, we discuss the experiment setting in Sec. 4.3.

4.1. Framework of internal and external stress on DNNs

We systematically apply internal and external stress on DNNs. The process of internal and external stress on DNNs is shown in Fig. 3, which is a three-step framework (adversarial attack on DNNs, synaptic filtering of DNNs, combined network performance) that leads to parameter score calculation for the DNN parameter characterization.

4.1.1. Attack on DNNs

In evaluating networks to internal stress, we compare the network performances to the synaptic filtering procedure for clean and adversarial (external stress) datasets (discussed in the following Sec. 4.1.2). In this study, we work primarily with the FGSM attack [13] for the adversarial perturbation formulation; other attack formulation methods would not affect the synaptic filtering described in this section. The synaptic filtering technique is designed to be applied to a network with any variation on the inputs, therefore, the nature of the attack formulation method can be changed without affecting the synaptic filtering technique.

In order to experiment with an adversarial dataset, we must define some constraints of the attack [Fig. 3(Left block)], such that the synaptic filtering responses are comparable between different network architectures and datasets. The constraints imposed upon the adversarial attack magnitude ϵ are, as follows:

Definition 7 *(minimum attack bound ϵ_0 – constraint 1.).* We limit the adversarial attack to follow $p(\hat{y}_\epsilon = y|x + \delta_{\epsilon_0}) < p(\hat{y} = y|x)$, for all inputs x in the test dataset. This constraint allows us to select a suitable minimum attack magnitude ϵ_0, such that otherwise correctly classified inputs are misclassified, due to the adversarial attack.

Definition 8 *(maximum attack bound ϵ_E – constraint 2.).* We limit the adversarial attack to a suitable maximum attack magnitude ϵ_E, such that the network test accuracy is above a random guess ($\hat{y}_{\epsilon_E} K > 1$), i.e., we have the constraint: $p(\hat{y}_{\epsilon_E} = y|x + \delta_{\epsilon_E})K > 1$, for all inputs x in the test dataset.

Definition 9 *(relative attack ϵ – constraint 3.).* To compare the performance of different network architectures and datasets to the synaptic filtering procedure, we must consider values of ϵ for different networks/datasets that reduce the network performance equally. Considering two different networks f_1 and f_2, we use a single attack δ, for which ϵ_1 and ϵ_2 are the *relative attack* magnitudes for f_1 and f_2. Suitable values of ϵ_1 and ϵ_2 should be chosen, such that $f_1(x,\theta) - f_1(x + \delta_{\epsilon_1}, \theta) \approx f_2(x,\theta) - f_2(x + \delta_{\epsilon_2}, \theta)$ thus ensuring that the adversarial perturbations affect the network performances equally.

4.1.2. Synaptic filtering of DNNs

We investigate a set of synaptic filters $h = \{h_1, h_2, h_3\}$ containing three different synaptic filters [Fig. 3(Middle block)]: h_1, the ideal *high-pass* filter; h_2, the ideal *low-pass* filter and h_3 the *pulse wave* filter. The operation of filtering for the three different filters is detailed in Eq. (1).

We apply filter h_1 to the learned (unperturbed) network parameters θ, resulting in perturbed network parameters $\tilde{\theta}_{1,\alpha_i}$ for a given threshold α_i, as per:

$$\tilde{\theta}_{1,\alpha_i} = h_1(\theta, \alpha_i) = \begin{cases} 0 & \text{if } \theta \leq \alpha_i, \\ 1 & \text{otherwise} \end{cases}, \tag{7}$$

where $\alpha_i \in \alpha$, and $\alpha = \{\alpha_0, \alpha_1, \ldots, \alpha_A\}$. We create $|\alpha|$ thresholds between the lower and upper bounds $\alpha_0 = \min(\theta)$ and $\alpha_A = \max(\theta)$. This results in a set of filtered networks S_α with each threshold defined as $\alpha_i = \alpha_{i-1} + \Delta_\alpha$ for a step length $\Delta_\alpha = [\max(\theta) - \min(\theta)]/A$ (i.e. $\Delta_\alpha = 1/A$ when α is normalised between 0 and 1) and $i = \{i \in \mathbb{N} : 0 \leq i \leq A\}$.

Similar to filter h_1, we apply the filter h_2 to the learned (unperturbed) network parameters θ from the opposite direction, resulting in perturbed network parameters $\tilde{\theta}_{2,\alpha_i}$ for an $\alpha_i \in \alpha$:

$$\tilde{\theta}_{2,\alpha_i} = h_2(\theta, \alpha_i) = \begin{cases} 0 & \text{if } \theta \geq \alpha_i, \\ 1 & \text{otherwise} \end{cases}, \tag{8}$$

where $\alpha_i = \alpha_{i-1} - \Delta_\alpha$, $\alpha_0 = \min(-\theta)$, and $\alpha_A = \max(-\theta)$.

The pulse wave filter h_3, applied to θ results in equal filtered parameters θ_{3,α_i} for values of α_i increased from $\min(\theta)$ to $\max(\theta)$ or decreased from $\max(\theta)$ to $\min(\theta)$. The results of filter h_3 are given by:

$$\tilde{\theta}_{3,\alpha_i} = h_3(\theta, \alpha_i) = \begin{cases} 0 & \text{if } \alpha_i - \frac{\Delta_\alpha}{2} < \theta \leq \alpha_i + \frac{\Delta_\alpha}{2}, \\ 1 & \text{otherwise} \end{cases}, \tag{9}$$

where $\alpha_i = \alpha_{i-1} \pm \Delta_\alpha$, $\alpha_0 = \min(\pm\theta)$, and $\alpha_A = \max(\pm\theta)$. In Eq. (9), the threshold window shifts by Δ_α centred at threshold α_i with either side having a length $\frac{\Delta_\alpha}{2}$.

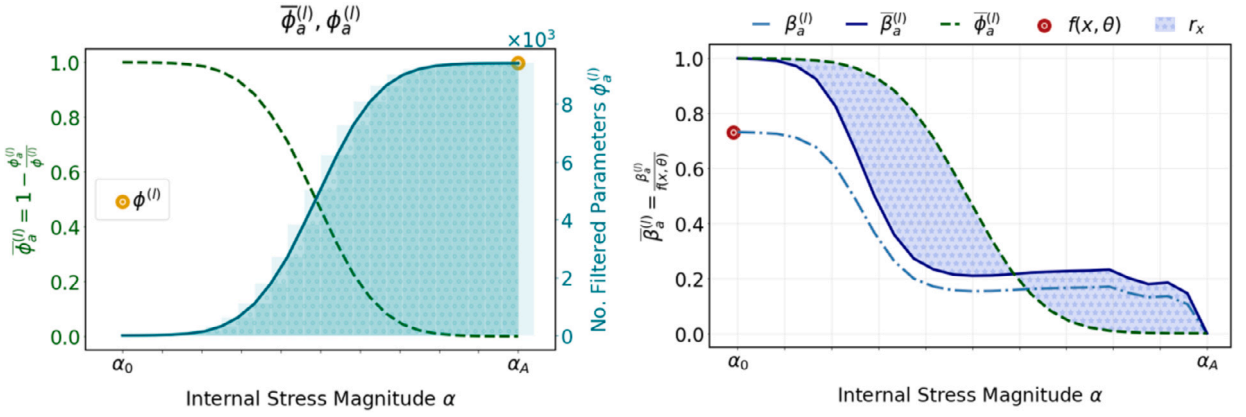
These three filters h_1, h_2, and h_3 with distinct properties when applied to a DNNs with threshold α_i offers three sets of distinct perturbed networks $f(\tilde{\theta}_{1,\alpha_i}, x)$, $f(\tilde{\theta}_{2,\alpha_i}, x)$, and $f(\tilde{\theta}_{3,\alpha_i}, x)$. Therefore, we require three baseline network performances corresponding to the properties of the respective synaptic filters against which the three sets of perturbed networks are compared.

Baseline network performances We denote ϕ_1, ϕ_2 and ϕ_3 to be the number of parameters filtered out by the synaptic filters h_1, h_2 and h_3 corresponding to filtering threshold α_i. If the synaptic filtering procedure is only applied to a local layer l (e.g., only on a convolutional layer or a linear layer) then $\phi^{(l)}$ is the maximum number of parameters in local layer l. For the whole network ϕ denote the maximum number of parameters in the network. Let us consider $\phi_1^{(l)}$ to the number parameters filtered out by the filter h_1 for the layer l at threshold α_i, then the base network performance $\overline{\phi}_{1,\alpha_i}^{(l)}$ at threshold α_i is given as:

$$\overline{\phi}_{1,\alpha_i}^{(l)} = 1 - \frac{\phi_{1,\alpha_i}^{(l)}}{\phi^{(l)}}. \tag{10}$$

Similarly, the baseline network performances for filters h_2 and h_3 are $\overline{\phi}_{2,\alpha_i}^{(l)}$ and $\overline{\phi}_{3,\alpha_i}^{(l)}$. In Eq. (10), the fraction $\frac{\phi_1^{(l)}}{\phi^{(l)}}$ is a ratio between the number of parameters removed to the total number of parameters in the layer, defining the *compactness* of the filtered layer.

We consider the baseline network performance for all values of α, which we use to determine the parameter characteristics to synaptic filtering, as a function that reduces the network performance proportionally to the internal stress (synaptic filtering) applied on the network (see Fig. 4a). Using this definition of the baseline network performance, we expect the network performance to decrease proportionally to the number of parameters filtered by the synaptic filtering procedure (see Fig. 4b). The underlying



(a) Baseline Network Performance.

(b) Baseline Network Performance and Scaled Synaptic Filtering Performance.

Fig. 4. (a) Baseline network performance (green dotted line) $\bar{\phi}_1^{(l)}$ [Eq. (10)] for ResNet-18 trained for 100 epochs on CIFAR10. $\phi_1^{(l)}$ is the function that contains the number of parameters filtered (teal solid line) for filtering thresholds in α for filter h_1 on layer l. The maximum number of parameters in layer l is denoted by $\phi^{(l)}$ (yellow dot). (b) Comparison of the scaled of synaptic filtering performance with baseline network performance, and synaptic robustness computation. The network performance to the synaptic filter is $\beta_1^{(l)}$ (blue dotted line), which is scaled w.r.t. the unperturbed network accuracy $f(x,\theta)$ (red dot), resulting in $\bar{\beta}_1^{(l)}$ (blue solid line). The blue shaded region r_x, enclosed by base system response $\bar{\phi}_1^{(l)}$ (green dotted line) is the area [Eq. (14) and Eq. (15)] of synaptic robustness.

assumption of the baseline network performance is that the parameters being filtered out have an overall influence on the network performance. Hence, the baseline network performance represents the expected behaviour of the network, given as the classification accuracy on the test set, whilst the network is subjected the synaptic filtering procedure for all synaptic filtering threshold values α.

Network compactness Our synaptic filtering method is a systematic ablation of DNN parameters to analyze variations in network performance caused by parameter filtering. We show that a proportion of the network parameters can be filtered out from a DNN, whilst retaining (and occasionally improving) the network performance on both clean and adversarially perturbed test sets [40]. The characteristics of the baseline network performance, describe a network with parameters that, when filtered, proportionally influence the network performance. From Eq. (10), the proposed baseline network performance is linked to the compactness of the network/layer; the characteristics of the baseline network performance are inversely proportional to the compactness ratio of the network/layer weights. For a specific non-random synaptic filtering method, the compactness characteristics of a network are constant for different variations to the input (e.g. adversarial attacks). Thus, we can compare the scaled network performances of a network to both clean and adversarial datasets, against the baseline network performance.

Network vs. adversary For a network, we define the network performances for all synaptic filtering thresholds α to be an $|\alpha|$-length vector of the network prediction accuracies $p(\hat{y}_\alpha = y | f(x, \tilde{\theta}_\alpha))$ on the test set x. The network performance to the synaptic filtering h_1 [Eq. (7)], h_2 [Eq. (8)] and h_3 [Eq. (9)] are given as β_1, β_2 and β_3 respectively. We construct a clean network performance matrix β on inputs x by combining β_1, β_2 and β_3 as:

$$\beta = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} = \begin{bmatrix} p(\hat{y}_\alpha = y | f(\tilde{\theta}_1, x)) \\ p(\hat{y}_\alpha = y | f(\tilde{\theta}_2, x)) \\ p(\hat{y}_\alpha = y | f(\tilde{\theta}_3, x)) \end{bmatrix}. \tag{11}$$

We further apply the synaptic filtering to the network under an adversarial attack δ with perturbation magnitudes ϵ, resulting in adversarial network performance matrix β_ϵ.

$$\beta_\epsilon = \begin{bmatrix} \beta_{1,\epsilon} \\ \beta_{2,\epsilon} \\ \beta_{3,\epsilon} \end{bmatrix} = \begin{bmatrix} p(\hat{y}_\alpha = y | f(\tilde{\theta}_1, x_{\delta_\epsilon})) \\ p(\hat{y}_\alpha = y | f(\tilde{\theta}_2, x_{\delta_\epsilon})) \\ p(\hat{y}_\alpha = y | f(\tilde{\theta}_3, x_{\delta_\epsilon})) \end{bmatrix} \tag{12}$$

Targeted parameters The matrices β and β_ϵ are the network performances on clean and adversarial datasets to the synaptic filtering method that are the two different DNN states to compare. Thus, through a comparison of β and β_ϵ (see Fig. 5), we expose the specific parameters (targeted parameters) that are either negatively, invariantly or positively affecting the synaptic filtering performances for the adversarial dataset, compared to the clean dataset.

4.1.3. Combined network performance of synaptic filters

Different synaptic filtering methods expose different characterisations of parameters of the network. Thus we combine the network performances of different synaptic filters using a function $g(\cdot)$ to form a combined network performance $\tilde{\beta}$, as shown in the synaptic filtering framework in Fig. 3(right). In order to combine the performances, let us consider β as the network performance to be combined; the procedure is the same for the adversarial network performances to the synaptic filters β_ϵ. We take $\tilde{\beta}$ as the performance

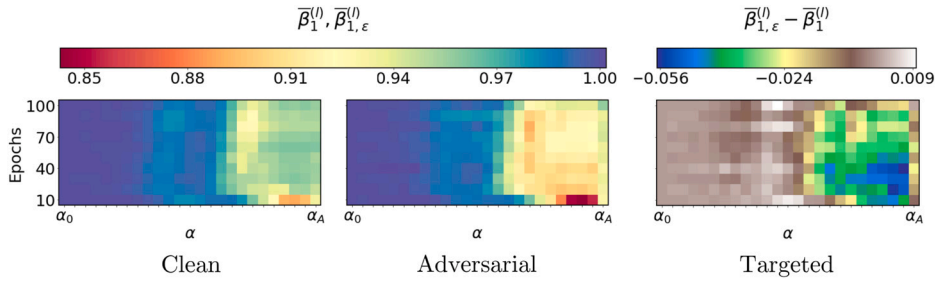


Fig. 5. Learning *landscape* of layers and the *regime change* of test accuracy. Targeted parameters of ResNet-18 trained on MNIST using filter h_1. Showing the combined responses for layer 'layer3.0.conv2', measured every 10 epoch up to 100 epochs. The difference between clean (left) and adversarial (middle) responses results in the targeted parameters (right). Every pixel on the clean and adversarial images represent the network test accuracy and for targeted image it is the difference between former two over all evaluated epochs and α.

of the perturbed network (synaptic filtering performance) relative to the unperturbed network performance $f(x,\theta)$. Subsequently, we take the mean of the performances of the network of all three different filters, as such:

$$\tilde{\beta}_i = g(\overline{\beta}_i) = \frac{1}{|h|}\sum_{j\in h}\overline{\beta}_{j,i} \quad \text{for } i = 1,\ldots,|\alpha|. \tag{13}$$

Fig. 6 shows an example of network accuracy results of the synaptic filtering procedure applied to layer 'conv1' of ResNet-18 trained on CIFAR10. The top row shows the effects of three different filters on network accuracy; the middle row shows each filter's epoch-wise effect on network accuracy. The third row is the effect of the combined response [as per Eq. (13)] of the filters.

Similarly, the combined adversarial network performance $\tilde{\beta}_\epsilon$ is computed by replacing $\bar{\beta}$ with $\bar{\beta}_\epsilon$ in Eq. (13), where $\bar{\beta}_\epsilon$ is the performance of the perturbed network (synaptic filtering performance) relative to the unperturbed network performance $f(x_\epsilon,\theta)$ on adversarial perturbed datasets x_ϵ. Although the combined network performances $\tilde{\beta}$ and $\tilde{\beta}_\epsilon$ offer more descriptive information to examine the network parameters, a single synaptic filter is also able to expose the targeted network parameters. As calculating $\tilde{\beta}$ and $\tilde{\beta}_\epsilon$ is computationally expensive for local analysis (as this increases exponentially to the number of local layers in a DNN), we suggest computing $\tilde{\beta}$ and $\tilde{\beta}_\epsilon$ for all network parameters (i.e., global analysis). Fig. 7 shows the combined synaptic filtering responses (local response is in the left three columns and the global response is on the rightmost column in Fig. 7) for ResNet-18 trained on MNIST, CIFAR10, and Tiny ImageNet datasets (see each row in Fig. 7 for respective dataset) for every 10 epochs up to 100 epochs).

4.2. Parameter scoring for DNN parameter characterization

To expose the network parameters targeted by the adversary, let us consider the network performance $\beta_1^{(l)}$ for synaptic filter h_1 on layer l. We scale $\beta_1^{(l)}$ relative to $f(x,\theta)$ resulting in $\overline{\beta}_1^{(l)}$; the baseline network performance is $\overline{\phi}_1^{(l)}$ [Eq. (10)] and the procedure is captured in Fig. 4. Similarly, we compute $\overline{\beta}_2^{(l)}$ and $\overline{\beta}_3^{(l)}$ for synaptic filters h_2 and h_3. The combined performance of the three different synaptic filters is $\hat{\tilde{\beta}}^{(l)}$ [Eq. (13)].

4.2.1. Parameter score for clean data

We take the baseline network performance $\tilde{\phi}_1^{(l)}$ [Eq. (10)] for synaptic filter h_1 as the point to which we evaluate the filtered network responses to. We take $\bar{\phi}_1^{(l)}$ to describe a network/layer that contains neither an excess nor a deficiency of parameters that influence the network performance (i.e. the removal of any parameter affects the network performance). The network performance, on average will react *inversely proportional* to ablation of network parameters to synaptic filtering. The parameter score to synaptic filtering for a network using a clean dataset is r_x is shown in Fig. 8a(top)] and given as:

$$r_x = \sum_{i=0}^{A}(\bar{\beta}_1^{(l)} - \bar{\phi}_1^{(l)})\Delta_\alpha, \tag{14}$$

Where Δ_α is the change in the α threshold window. A parameter score *equal to* 0 signifies that the network/layer responds, on average, *proportionally* to synaptic filtering, i.e., proportional to variations in architecture and thus is considered *robust*. Where the score r_x is *less than* 0, this indicates that the network/layer contains *fragile* parameters to the network performance. Conversely, where the value of r_x is *greater than* 0, the parameter score indicates that the network/layer contains *antifragile* parameters, where the removal of parameters from the network/layer results in a network performance that is better than the baseline network performance.

4.2.2. Parameter score for adversarial data

The parameter score to synaptic filtering for a network using an adversarial dataset is r_{x_ϵ}, and is calculated using the baseline network performance $\bar{\phi}_1^{(l)}$. The baseline network performance is compared with the adversarial dataset performance $\bar{\beta}_{1,\epsilon}^{(l)}$ to give the parameter characterization score r_{x_ϵ}, as per:

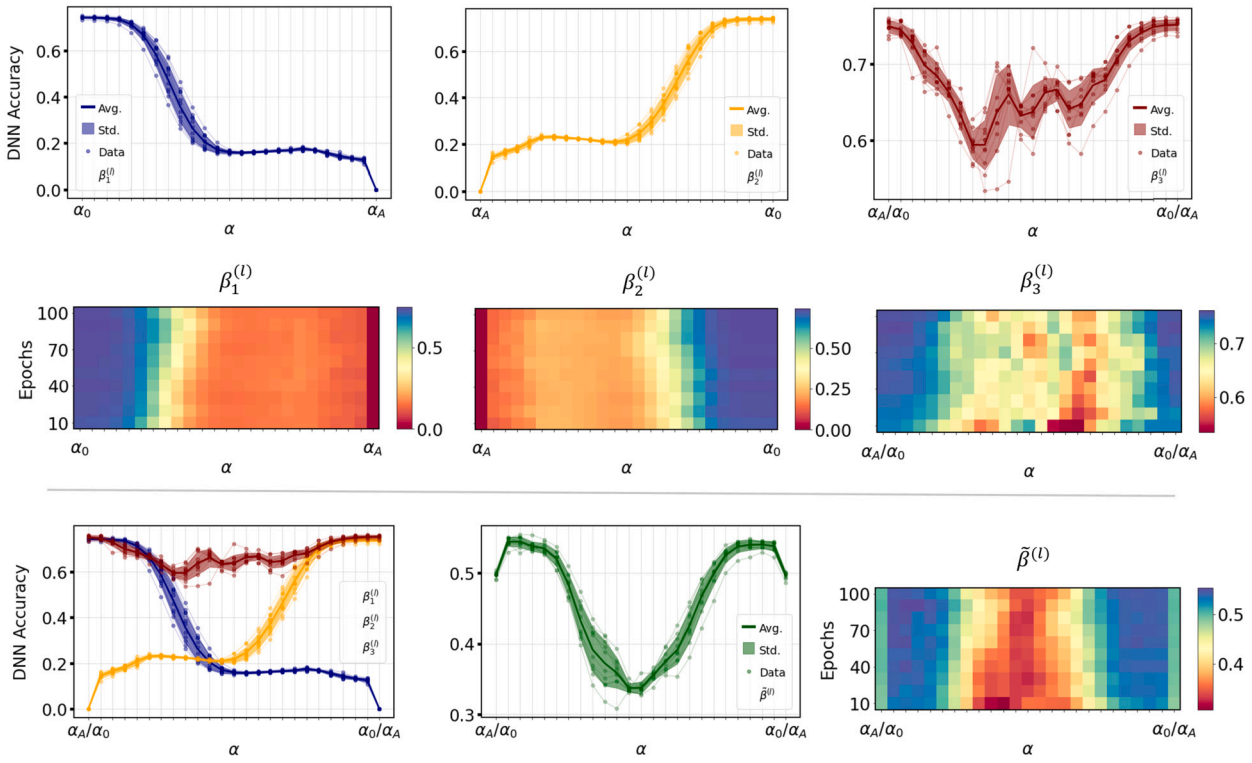


Fig. 6. Example of network accuracy results of the synaptic filtering procedure applied to layer 'conv1' of ResNet-18 trained on CIFAR10, shown to illustrate the combined system response. The bottom-left plot is a combination of three top-row plots.

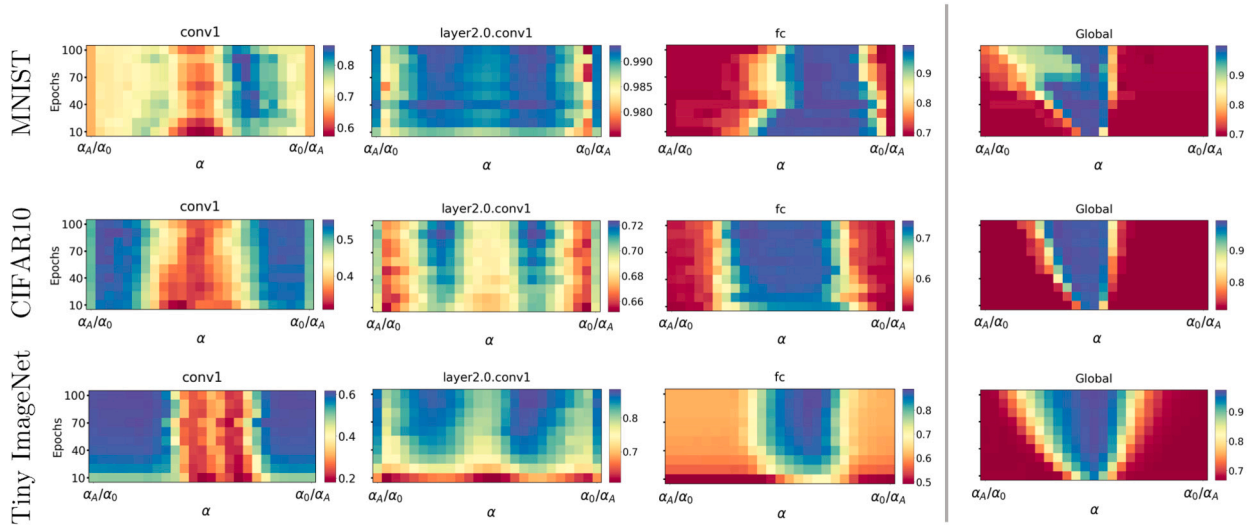
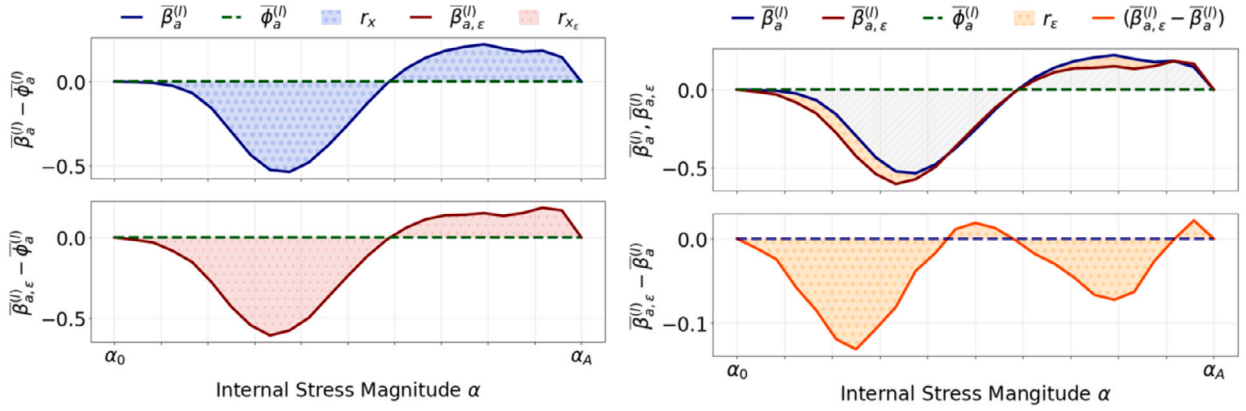


Fig. 7. Combined synaptic filtering responses for ResNet-18 trained on CIFAR10, MNIST and Tiny ImageNet datasets for every 10 epochs up to 100 epochs. (**1**) Local layer-wise system response to the filtering methods for all α values. (**2**) Global network responses using the full network for all α values. Pixel intensities on the shown images represent the average network accuracy using the different synaptic filters on the clean test dataset, for each α_i in α.

$$r_{x_\epsilon} = \sum_{i=0}^{A} (\tilde{\beta}_{1,\epsilon}^{(l)} - \tilde{\phi}_1^{(l)}) \Delta_\alpha, \tag{15}$$

Where Δ_α is the change in the α threshold window. A parameter score *equal to* 0 signifies that the network/layer responds, over all magnitudes of internal stress, *proportionally* to synaptic filtering, i.e., proportional to variations in architecture and thus is considered *robust*. Where scores r_x and r_{x_ϵ} are *less than* 0, this indicates that the network/layer contains *fragile* parameters w.r.t. the



(a) Clean and Adversarial Performances

(b) Parameter Score Computation

Fig. 8. (a) Synaptic parameter score computation for clean and adversarial inputs (ResNet-18, CIFAR10, 100 epochs, layer 'conv1'). The scaled network performance to the clean $\bar{\beta}_1^{(l)}$ (top) and adversarial $\bar{\beta}_{1,\epsilon}^{(l)}$ (bottom) datasets. The clean and adversarial parameter scores are r_x and r_{x_ϵ}. (b) The behaviour of the network responses (ResNet-18, CIFAR10, 100 epochs, layer 'conv1') using synaptic filtering on the CIFAR10 clean $\bar{\beta}_1^{(l)}$ and adversarial $\bar{\beta}_{1,\epsilon}^{(l)}$ datasets over normalised α (top). The area r_ϵ [Eq. (16)] is the adversarial parameter score (bottom).

network performance. Conversely, where r_x and r_{x_ϵ} are *greater than* 0, the scores indicate that the network/layer contains *antifragile* parameters.

4.2.3. Difference of parameter scores

To compute the effects of the adversarial attack on the parameter characterisation, using our proposed synaptic filtering method, we take the baseline network performance to be the synaptic filtering performance on the clean dataset ($\bar{\phi}_1^{(l)} = \bar{\beta}_1^{(l)}$). The difference in the adversarial dataset performance $\overline{\beta}_{1,\epsilon}^{(l)}$ and clean dataset performance $\overline{\beta}_1^{(l)}$ (baseline network performance), results in the effects of the adversary on the synaptic filtering performance of the network. We take the *area of the residual* as the effects of the adversary on the network. The value of r_ϵ is computed by taking the discrete area difference, as shown in Fig. 8b (bottom) and expressed as:

$$r_\epsilon = \sum_{i=0}^{A} (\overline{\beta}_{1,\epsilon}^{(l)} - \overline{\beta}_1^{(l)}) \Delta_\alpha. \tag{16}$$

If the network performs equally to clean and adversarial datasets for all filtering thresholds α, the value of $r_\epsilon = 0$. Where $r_\epsilon < 0$, the network performance on the adversarial dataset is greater than the network performance on the clean dataset. This signifies that the evaluated network/layer contains parameters that increase the network performance on the adversarial dataset compared to the clean dataset. Conversely, $r_\epsilon > 0$ signifies that the network performance on the clean dataset is greater than the network performance on the adversarial dataset. This signifies that the evaluated network/layer contains parameters that decrease the network performance on the adversarial dataset compared to the clean dataset. Hence, the magnitude of r_ϵ gives us a scalar value of the difference in clean and adversarial responses to the filtering.

4.3. Experimental set-up

Our experiment setting includes standard training of state-of-the-art DNNs on popular benchmark datasets.

State-of-the-art datasets used All experiments[1] in this study are performed on the MNIST [18], CIFAR10 [19] and Tiny ImageNet [20] datasets. The MNIST and CIFAR10 datasets both respectively contain 80,000 examples in the training set and 10,000 examples in the test set. The Tiny ImageNet dataset contains 80,000 training and 20,000 test examples from the original training set [20].

State-of-the-art DNNs studies On the benchmark datasets, we train ResNet-18, ResNet-50 [15], SqueezeNet v1.1 [16] and ShuffleNet V2 x1.0 [17]. Each network was trained for 100 epochs, and the model of every 10 epochs was stored for analysis of our methodology. We investigated all convolutional and fully connected layers of ResNet-18, ResNet-50, SqueezeNet v1.1 and ShuffleNet V2 x1.0 only, any intermediary functions, such as the batch normalization layers, activation functions and pooling layers remain unaltered.

Training of DNNs on clean datasets For the training, the parameters of each DNN were initialised using the Kamming Uniform [45] method. We use a cross-entropy loss function and the Adam optimizer [46] configured with $\gamma = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\lambda = 0$ and

[1] Source code: https://github.com/SynapFilter/InferLink.

$\epsilon = 1 \times 10^{-08}$ for training networks. We train the networks using clean datasets only and apply the adversarial attack only to the test datasets for analysis using the synaptic filtering methodology. Networks are saved at every 10 epochs during network training to a maximum of 100 epochs. Saved networks are subsequently evaluated using the proposed synaptic filtering methodology, the results presented in Sec. 5 are shown for the saved networks.

Adversarial attack on datasets For the adversarial attack, we use the single-step FGSM attack [13] and analyze the difference in network performance on the test set to the proposed synaptic filtering methods (Sec. 4). The effectiveness of an adversarial attack on a given dataset can be attributed to the complexity of the datasets the attack has been applied to.

Collection of results We normalize the r_x and r_{x_ϵ} parameter score values from Sec. 4.2.1 to be between -0.5 (indicating fragility) and 0.5 (indicating antifragility) with the mid-point being 0 (indicating robustness). We carry out the same normalization procedure independently for all r_ϵ values from Sec. 4.2.3 to be between -0.5 and 0.5.

For each network and dataset, the synaptic filtering responses are averaged over three different randomly initialised (as per [47]) and trained networks. In order to satisfy constraint 3 from Sec. 4, we use a line search algorithm to find the optimal ϵ value for each model and dataset, that satisfies: $f(x + \delta_\epsilon, \mathbf{W}) \approx 0.5 \cdot f(x, \mathbf{W})$. When carrying out the synaptic filtering procedure, we select $A = 25$ for all experiments carried out. Therefore, the filtering step size $\Delta_\alpha = 0.04$ over the normalised range of parameters in the evaluated network/layer. Where computational resources permit, we recommend using larger values of A for experimentation in order to more accurately compute the parameter scores.

5. Results and analysis

The results of global (full network parameters) and local (network layer parameters) analysis shown in Figs. 9 and 10 describe the fragility, robustness, and antifragility characteristics of parameters (cf. Sec. 4.2.1 and Figs. 9 and 10). Furthermore, the results show the adversarially targeted (r_ϵ) parameters (cf. Sec. 4.2.3 and Figs. 9 and 10). We identify parameter characteristics that are invariant for clean and adversarial datasets across different datasets and networks.

Fragility, robustness, and antifragility The global parameter scores for networks on different datasets are shown in Fig. 9. We find that ResNet18 and ResNet50 networks exhibit invariant parameter characteristics to different datasets: particularly for r_x and r_{x_ϵ} values related to the CIFAR10 and ImageNet Tiny performances, over 100 epochs. The adversarial targeting results (r_ϵ values) are comparable for the CIFAR10 and ImageNet Tiny responses, with r_ϵ values for MNIST, suggesting that the clean dataset response is consistently greater than the adversarial dataset performance. From the ShuffleNet V2x1.0 parameter scores, we find distinctions in r_x and r_{x_ϵ}, for the MNSIT dataset, over 100 epochs. We see the ShuffleNet V2x1.0 parameters as transitioning from fragile to antifragile for the ImageNet Tiny dataset. From the SqueezeNet-v1.1 results for the ImageNet Tiny dataset, we observe a convergence of r_x and r_{x_ϵ} to 0 over 100 epochs, indicating the network performance as robust, for both clean and adversarial datasets.

The local parameter scores provide a *learning landscape* to examine individual network parameters (Fig. 10). All of the evaluated network and dataset parameter scores exhibit *invariant* fragility characteristics (marked as 'Fr') at the 1-st convolutional layer and the l-th linear layer, for both clean (r_x) and adversarial (r_{x_ϵ}) parameter scores. This is further shown in Fig. 10a ImageNet Tiny; Fig. 10c MNIST, CIFAR10 and ImageNet Tiny, and Fig. 10d ImageNet Tiny. We see the presence of robust parameters (marked as 'Ro') in Fig. 10a CIFAR10 and ImageNet Tiny; Fig. 10b ImageNet Tiny; Fig. 10c MNIST and CIFAR10, and Fig. 10d CIFAR10 ImageNet Tiny. Antifragile parameters (marked as 'Af') are distinctly visible in Fig. 10a MNIST and CIFAR10; Fig. 10b MNIST, CIFAR10 and ImageNet Tiny; Fig. 10d MNIST and CIFAR10. Furthermore, periodic robustness characteristics are shown in Fig. 10a ImageNet Tiny; Fig. 10b MNIST, CIFAR10 and ImageNet Tiny; Fig. 10c MNIST, CIFAR10, ImageNet Tiny, and Fig. 10d CIFAR10 and ImageNet Tiny.

Adversarially targeted parameters In Fig. 5, we present targeted parameters to an adversarial attack using the combined network response for ResNet-18 trained on MNIST. We further see targeted parameters using the parameter scores r_ϵ (Sec. 4.2.3) from Fig. 9 and Fig. 10. In Fig. 10, we show that the network response is greater for the adversarial dataset than the clean dataset (marked by 'r_{x_ϵ}'), as shown in layers of Fig. 10c CIFAR10 and ImageNet Tiny. We find instances where both the adversarial performance and clean performance are equal, indicating that the layer response is robust (marked by 'Ro') and shown in Fig. 10a MNIST, CIFAR10 and ImageNet Tiny; Fig. 10b MNIST, CIFAR10 and ImageNet Tiny; Fig. 10c MNIST, and Fig. 10d CIFAR10 and ImageNet Tiny. Furthermore, we see instances of the network performances for the clean dataset being greater than that of the adversarial dataset (marked by 'r_x'), shown in Fig. 10a MNIST and CIFAR10; Fig. 10b MNIST, CIFAR10 and ImageNet Tiny, and Fig. 10d MNIST and CIFAR10.

Effects of batch normalization We investigated the phenomenon of the network retaining classification performance despite all features at layer l removed (see column α_A in Fig. 5). When we investigate the output of layers deeper than l, we discover that residual features continue to propagate through the network despite the filtering out of network weights at the l-th layer. This is attributed to the Batch normalization (BN) layers that follow convolutional layers and are tasked with minimising covariance shift in the network [48]. When implementing a network architecture, we utilise the standard models in accordance with literature; the functionality of batch normalization layers is also predefined and remains unaltered in our analysis. Consider the condition where a convolutional layer l has been filtered maximally using a synaptic filter, the subsequent batch normalization computation is given as:

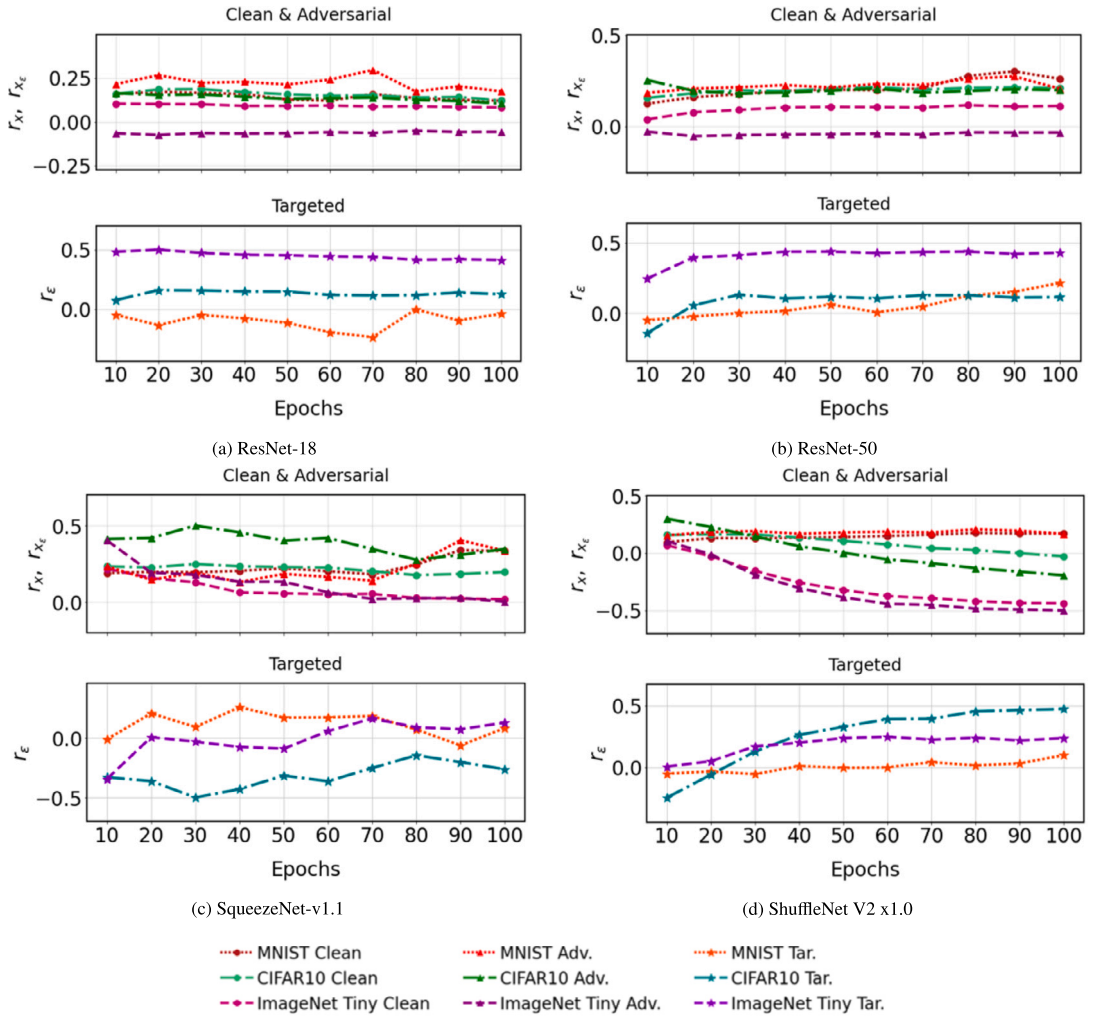


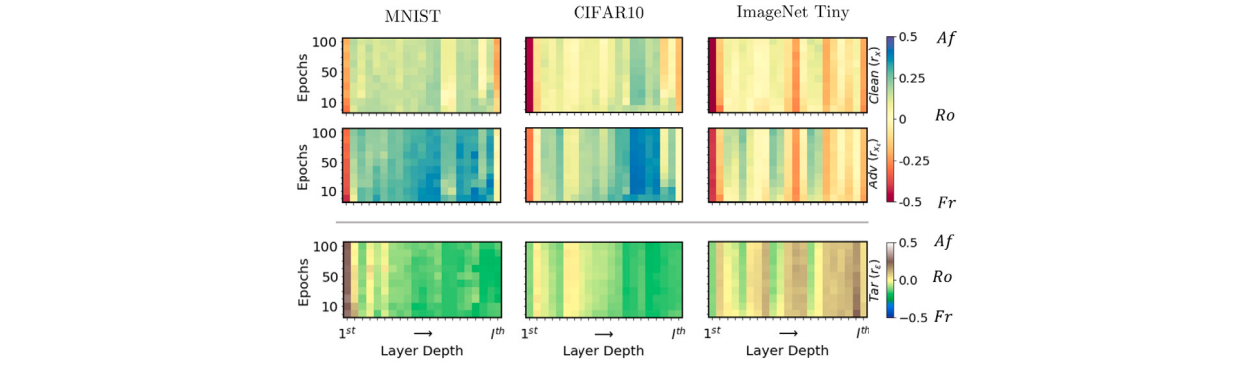
Fig. 9. Global parameter scores of (a) ResNet-18, (b) ResNet-50, (c) SqueezeNet-v1.1 and (d) ShuffleNet V2x1.0 over all datasets are r_x, r_{x_ϵ} and r_ϵ, measured every 10 epochs up to 100 epochs for the whole network parameters using synaptic filter h_1. The parameter score interpretation is given in Sec. 4.2.1 and Sec. 4.2.3.

$$\hat{y}^{(l)} = \frac{x^{(l)} - \mathbb{E}[x]}{\sqrt{\mathrm{Var}[x] + \epsilon}} * \gamma^{(l)} + \beta^{(l)} \tag{17}$$

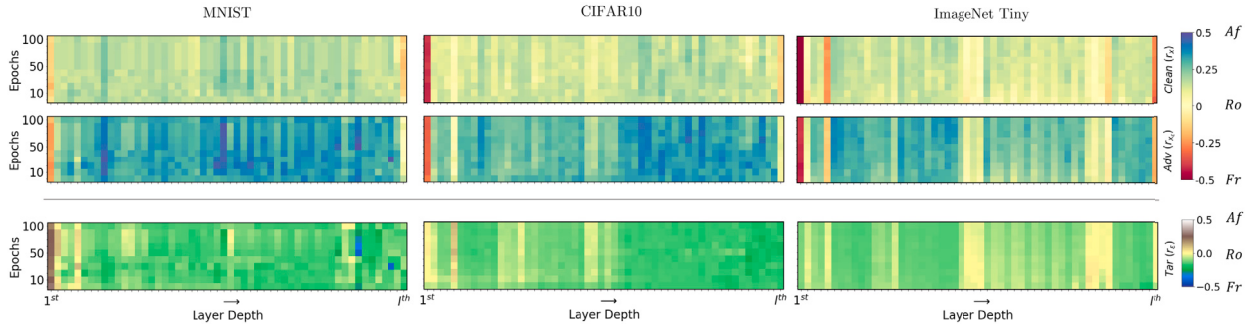
Where $\hat{y}^{(l)}$ is the output of the batch normalization process at the output of convolutional layer l; $y^{(l-1)}$ is the output of the previous convolutional layer $l-1$ given by $f(\tilde{\theta}^{(l)}_{\{1,2,3\}}, \hat{y}^{(l-1)})$. The variables $\gamma^{(l)}$ and $\beta^{(l)}$ are learnable parameter vectors and ϵ is a value added to the denominator for numerical stability (set to 1×10^{-5}). Implementations of networks compute the expectation and variance from Eq. (17) as running statistics during network training; the statistics calculated during training are used during network inference. In consequence, when the input to the BN layer following convolutional layer l is a 0 vector, the case where layer l has been filtered maximally through synaptic filtering, the BN layer retains features of the training batches, even when evaluating test sets. This is shown from the results in Fig. 11, where the filtering of parameters from certain layers results in only a slight decrease of network performance. The ability of the network to retain sufficient performance, despite the filtering out of certain layer parameters, is due to the features propagated during the forward pass by the batch normalization layer following the filtered layer.

Selective backpropagation on robust and antifragile parameters Upon identifying robust, fragile and antifragile parameters using the difference in parameter scores (see Sec. 4.2.3) we consider fragile parameters to be parameters that, when perturbed, result in greater degradation of synaptic filtering performance on the clean dataset compared to the adversarial dataset. Robust parameters show to be invariant to both clean and adversarial datasets, and antifragile parameters show to have an increased network performance on the clean dataset compared to the adversarial dataset.

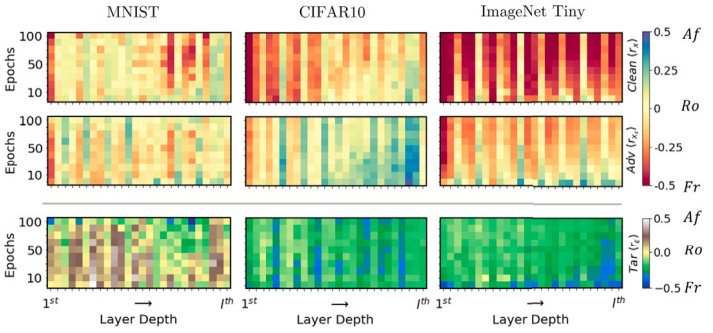
Thus, we consider fragile parameters to be parameters that are important to the network performance on the adversarial dataset. We propose selectively retraining only the robust and antifragile parameters using backpropagation. In order to carry out this operation during network training, we take a layer-wise approach that considers the parameter characterization scores of individual



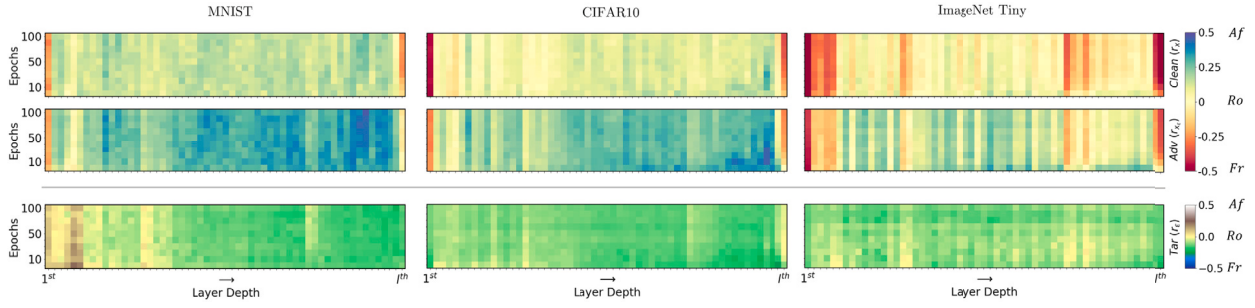
(a) ResNet-18



(b) ResNet-50



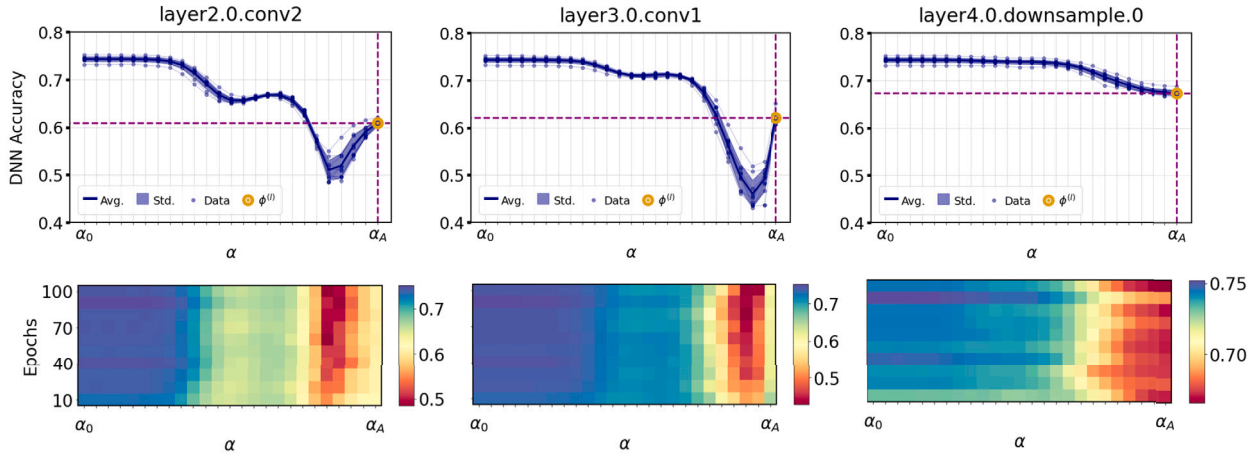
(c) SqueezeNet-v1.1



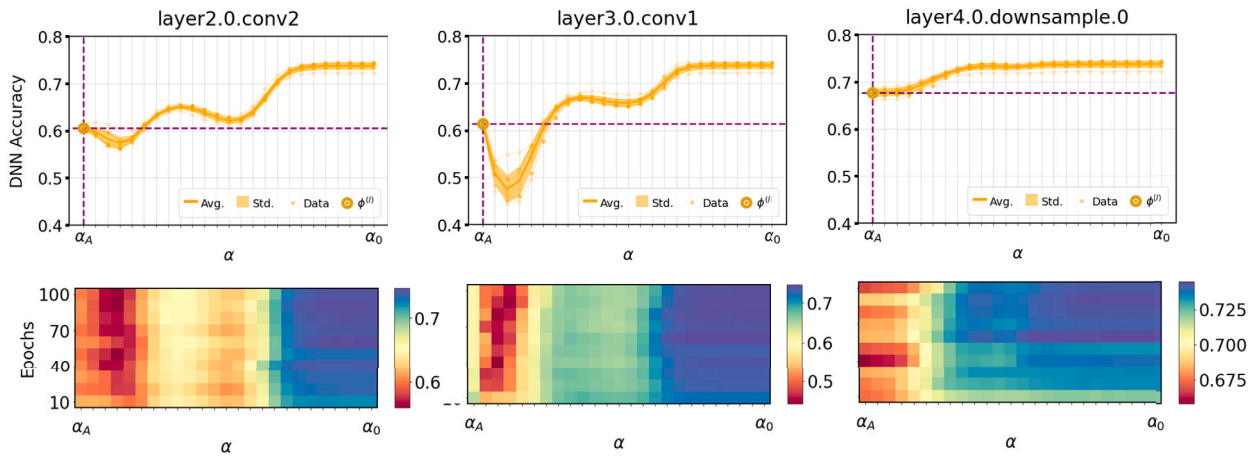
(d) ShuffleNet V2 x1.0

Fig. 10. Local parameter scores of (a) ResNet-18, (b) ResNet-50, (c) SqueezeNet-v1.1 and (d) ShuffleNet V2 x1.0 over all datasets. The parameter scores r_x, r_{x_ϵ} and r_ϵ are measured every 10 epochs up to 100 epochs and for all layers in the network for filter h_a. The parameter score interpretation is given in Sec. 4.2.1 and Sec. 4.2.3. The fragile, robust, and antifragile parameters of the network are respectively represented by values 'Fr,' 'Ro,' and 'Af' (see rightmost colour bar).

network layers and we subsequently omit the characterized fragile layers corresponding to negative parameter characterizations scores from network training by zeroing out the update gradients during training.



(a) Filter h_1



(b) Filter h_2

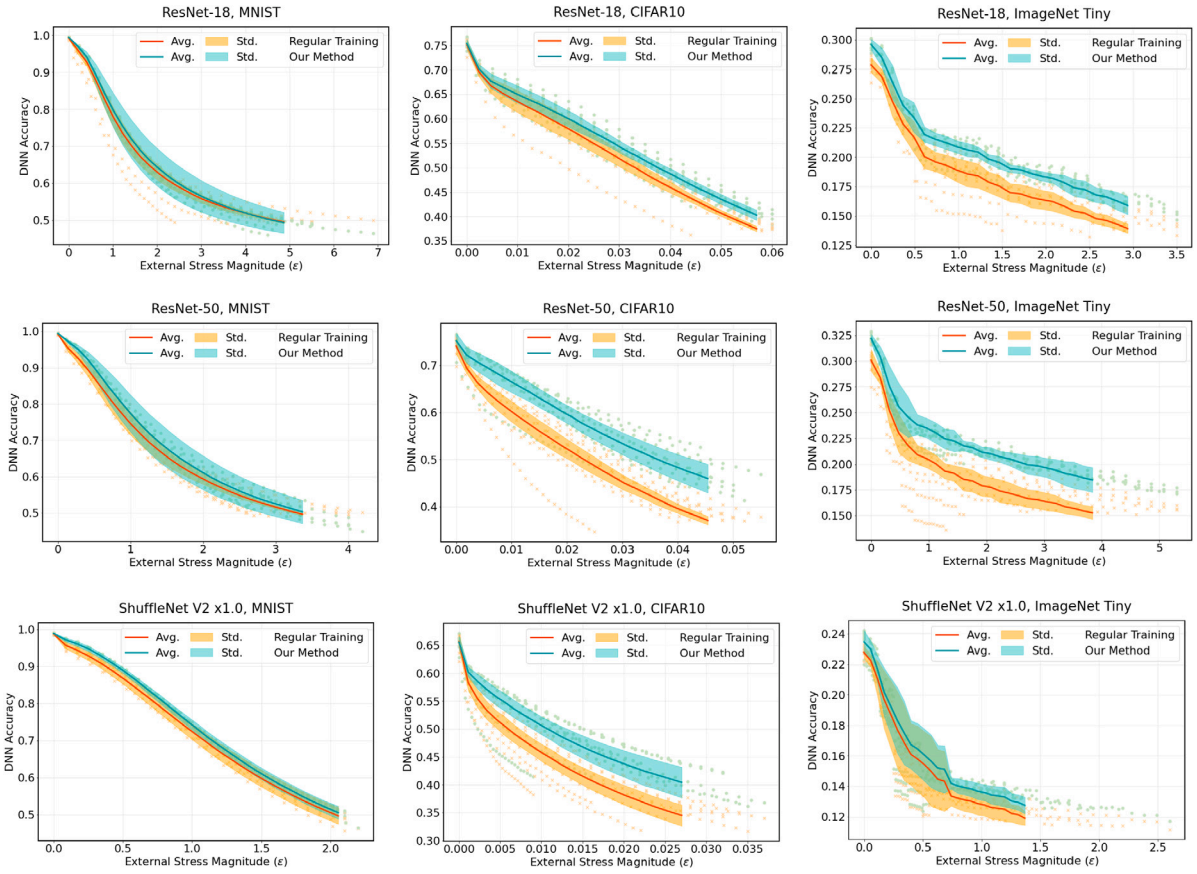
Fig. 11. Synaptic filtering network performances of ResNet-18 trained on CIFAR10 for layers 'layer2.0.conv1', 'layer3.0.conv1' and 'layer4.0.downsample.0'. The results show that, even after filtering all parameters in a layer, the network performs relatively well (shown by the purple dotted lines at $\Phi^{(l)}$, the maximum number of parameters filtered). This is due to the following batch normalization layer features propagating through the network during the forward pass, thus highlighting the effects of batch normalisation layers on network performance.

The results from our selective backpropagation method are shown in Fig. 12 where the mean (solid lines) and standard deviation (coloured shaded regions) of network performances are shown for networks tested at epoch 10 to epoch 100 measured every 10 epochs. We test each network to a maximum perturbation magnitude (external stress magnitude) of ϵ_E, which is selected using Definitions 7, 8 and 9. As can be seen from the results, our proposed method, shown in teal, outperforms the networks trained using regular backpropagation training, shown in orange, when considering robustness to adversarial attacks. Our proposed method shows to improve network robustness better on the CIFAR10 (Fig. 12b) and ImageNet Tiny (Fig. 12c) dataset compared to the MNIST (Fig. 12a) dataset. The effectiveness of the selective backpropagation method on CIFAR10 and ImageNet Tiny compared to MNIST can be attributed to the complexity of the datasets [49], where MNIST can be considered to have a lower complexity relative to CIFAR10 and ImageNet Tiny.

6. Conclusions

We can examine deep neural networks using our proposed synaptic filtering technique to characterize parameters of the network as fragile, robust and antifragile on both clean and adversarial inputs as a test bed. When subjected to synaptic filtering and an adversarial attack the fragile parameters are the parameters that cause a decrease in DNN performance. Whilst parameters characterized as robust cause the DNN performance to remain within a defined tolerance threshold (e.g. ±2% change in DNN performance). Parameters characterized as antifragile cause an increase in DNN performance.

Such an identification method can be applied to distill a trained network in order to make it usable in several resource-constrained applications, such as wearable devices. We offer parameter scores to evaluate the affects of specific parameters on the network performance and expose parameters targeted by an adversary. We find that there are global and local filtering responses that have



(a) MNIST (b) CIFAR10 (c) ImageNet Tiny

Fig. 12. Selective backpropagation re-training of the robust and antifrigle parameters. Network accuracy of ResNet-18 on (a) MNIST, (b) CIFAR10, (c) ImageNet Tiny datasets to external stress (adversarial attack) with magnitude ϵ.

invariant features to different datasets over the learning process of a network. For a given dataset, the filtering scores identify the parameters that are invariant in characteristics across different network architectures. We analyze the performance of DNN architectures through a selective backpropagation technique where we only retrain robust and antifragile parameters at given epoch. We compare the selective backpropagation technique with regular training to show that retraining only robust and antifragile parameters improves DNN robustness to adversarial attacks on all evaluated datasets and network architectures.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

We use open-source data

Appendix A. Supplementary material

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.artint.2023.104060.

References

[1] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.
[2] W. Samek, G. Montavon, et al., Explaining deep neural networks and beyond: a review of methods and applications, Proc IEEE 109 (3) (2021) 247–278.
[3] N. Papernot, P. McDaniel, et al., The limitations of deep learning in adversarial settings, in: EuroS&P, 2016, pp. 372–387.
[4] N. Carlini, D. Wagner, Towards evaluating the robustness of neural networks, in: Proc IEEE Symp Secur Priv (SP), 2017.
[5] N. Srivastava, G. Hinton, et al., Dropout: a simple way to prevent neural networks from overfitting, JMLR 15 (1) (2014) 1929–1958.

[6] R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V.I. Morariu, X. Han, M. Gao, C.-Y. Lin, L.S. Davis, NISP: pruning networks using neuron importance score propagation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.
[7] Z. Mariet, S. Sra, Diversity networks: neural network compression using determinantal point processes, arXiv preprint, arXiv:1511.05077.
[8] B.S. Oken, I. Chamine, W. Wakeland, A systems approach to stress, stressors and resilience in humans, Behavioural brain research 282 (2015) 144–154.
[9] C. Szegedy, W. Zaremba, et al., Intriguing properties of neural networks, in: ICLR, 2014.
[10] B. Biggio, I. Corona, et al., Evasion attacks against machine learning at test time, in: ECML PKDD, 2013.
[11] N.N. Taleb, R. Douady, Mathematical definition, mapping, and detection of (anti) fragility, Quantitative Finance 13 (11) (2013) 1677–1689.
[12] T. Freiesleben, The intriguing relation between counterfactual explanations and adversarial examples, Minds and Machines 32 (1) (2022) 77–109.
[13] I.J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, in: ICLR, 2015.
[14] S. Huang, N. Papernot, et al., Adversarial attacks on neural network policies, in: ICLR, 2017.
[15] K. He, X. Zhang, et al., Deep residual learning for image recognition, in: CVPR, 2016.
[16] F.N. Iandola, S. Han, et al., SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and < 0.5MB model size, arXiv:1602.07360v4.
[17] N. Ma, X. Zhang, et al., ShuffleNet V2: practical guidelines for efficient CNN architecture design, in: ECCV, 2018.
[18] Y. LeCun, C. Cortes, MNIST handwritten digit database, http://yann.lecun.com/exdb/mnist/, 2010.
[19] A. Krizhevsky, G. Hinton, et al., Learning multiple layers of features from tiny images, https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf, 2009.
[20] Y. Le, X. Yang, Tiny ImageNet visual recognition challenge, 2015, Stanford CS 231N.
[21] I.N. Karatsoreos, B.S. McEwen, Psychobiological allostasis: resistance, resilience and vulnerability, Trends in cognitive sciences 15 (12) (2011) 576–584.
[22] V. Ramanujan, M. Wortsman, A. Kembhavi, A. Farhadi, M. Rastegari, What's hidden in a randomly weighted neural network?, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 11893–11902.
[23] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, J. Kautz, Importance estimation for neural network pruning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019.
[24] X. Gao, R.K. Saha, M.R. Prasad, A. Roychoudhury, Fuzz testing based data augmentation to improve robustness of deep neural networks, in: 2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE), IEEE, 2020, pp. 1147–1158.
[25] Y. Wang, S. Wu, et al., Demiguise attack: crafting invisible semantic adversarial perturbations with perceptual similarity, in: IJCAI, 2021.
[26] H. Xu, Y. Ma, et al., Adversarial attacks and defenses in images, graphs and text: a review, Int. J. of Autom. and Comput. 17 (2) (2020) 151–178.
[27] D. Tsipras, S. Santurkar, et al., Robustness may be at odds with accuracy, in: ICLR, 2019.
[28] N. Akhtar, A. Mian, Threat of adversarial attacks on deep learning in computer vision: a survey, IEEE Access 6 (2018) 14410–14430.
[29] P. Samangouei, M. Kabkab, R. Chellappa, Defense-GAN: protecting classifiers against adversarial attacks using generative models, in: ICLR, 2018.
[30] X. Yuan, P. He, et al., Adversarial examples: attacks and defenses for deep learning, IEEE Trans Neural Netw Learn Syst 30 (9) (2019) 2805–2824.
[31] S. Han, J. Pool, et al., Learning both weights and connections for efficient neural networks, in: NIPS, 2015.
[32] K.A. Sankararaman, S. De, et al., The impact of neural network overparameterization on gradient confusion and stochastic gradient descent, in: ICML, 2020.
[33] S. Kornblith, M. Norouzi, et al., Similarity of neural network representations revisited, in: ICML, 2019.
[34] P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak, I. Sutskever, Deep double descent: where bigger models and more data hurt, Journal of Statistical Mechanics: Theory and Experiment 2021 (12) (2021) 124003.
[35] V. Ojha, G. Nicosia, Backpropagation neural tree, Neural Networks 149 (2022) 66–83.
[36] A. Ilyas, S. Santurkar, et al., Adversarial examples are not bugs, they are features, in: NIPS, 2019.
[37] C. Pravin, I. Martino, G. Nicosia, V. Ojha, Adversarial robustness in deep learning: attacks on fragile neurons, in: International Conference on Artificial Neural Networks, Springer, 2021, pp. 16–28.
[38] D. Blalock, J.J. Gonzalez Ortiz, J. Frankle, J. Guttag, What is the state of neural network pruning?, in: I. Dhillon, D. Papailiopoulos, V. Sze (Eds.), Proceedings of Machine Learning and Systems, vol. 2, 2020, pp. 129–146.
[39] R. Taylor, V. Ojha, I. Martino, G. Nicosia, Sensitivity analysis for deep learning: ranking hyper-parameter influence, in: 2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI), IEEE, 2021, pp. 512–516.
[40] A.S. Rakin, Z. He, L. Yang, Y. Wang, L. Wang, D. Fan, Robust sparse regularization: simultaneously optimizing neural network robustness and compactness, 2019.
[41] S. Ye, K. Xu, S. Liu, H. Cheng, J.-H. Lambrechts, H. Zhang, A. Zhou, K. Ma, Y. Wang, X. Lin, Adversarial robustness vs. model compression, or both?, in: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2019.
[42] A. Kurakin, I. Goodfellow, S. Bengio, Adversarial machine learning at scale, in: ICLR, 2017.
[43] J. Wang, Adversarial examples in physical world, in: IJCAI, 2021.
[44] J. Frankle, M. Carbin, The lottery ticket hypothesis: finding sparse, trainable neural networks.
[45] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: surpassing human-level performance on imagenet classification, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2015.
[46] D.P. Kingma, J. Ba Adam, A method for stochastic optimization, in: ICLR, 2015.
[47] K. He, X. Zhang, et al., Delving deep into rectifiers: surpassing human-levelperformance on ImageNet classification, in: ICCV, 2015.
[48] S. Ioffe, C. Szegedy, Batch normalization: accelerating deep network training by reducing internal covariate shift, in: F. Bach, D. Blei (Eds.), Proceedings of the 32nd International Conference on Machine Learning, in: Proceedings of Machine Learning Research, vol. 37, PMLR, 2015, pp. 448–456.
[49] F. Branchaud-Charron, A. Achkar, P.-M. Jodoin, Spectral metric for dataset complexity assessment, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019.