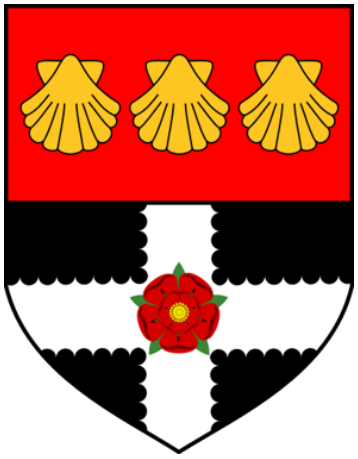
Fragility, Robustness, and Antifragility in Deep Neural Networks



Chandresh Pravin

Supervisor: Dr Varun K Ojha

Department of Computer Science
University of Reading

This dissertation is submitted for the degree of
Doctor of Philosophy

July 2023

I dedicate this PhD thesis to my family and friends, who have championed me throughout this journey. To my loving parents, Pravin and Bhumika, I express my deepest gratitude. Your never-ending encouragement has been my motivation for completing this work. I am so grateful for your love and guidance. To my friends, I thank you for your support in so many ways. You have listened to me when I needed to talk, you have celebrated my successes, and you have been there for me through challenging times. I am so grateful for your friendship. I could not have completed this PhD without the support of my family and friends. Thank you for everything.

Declaration

I declare that the thesis has been composed by myself and that the work has not be submitted for any other professional qualification or degree. I confirm that the work submitted is my own, except where work which has formed part of jointly-authored publications has been included. My contribution and those of the other authors to this work have been explicitly indicated below. I confirm that appropriate credit has been given within this thesis where reference has been made to the work of others.

The work presented in Chapter 3 was previously published in the International Conference on Artificial Neural Networks (ICANN 2021) as *Adversarial robustness in deep learning: attacks on fragile neurons* by myself, as the first author, along with Dr Ivan Martino, Dr Giuseppe Nicosia, and Dr Varun Ojha (PhD Supervisor). This study was conceived by all of the authors where, I carried out the necessary research experimentations and conducted the writing of the literature.

The work presented in Chapter 5 was previously published in the International Conference on Machine Learning Applications (ICMLA 2020) as *A novel ECG signal denoising filter selection algorithm based on convolutional neural networks* by myself, as the first author, along Dr Varun Ojha (PhD Supervisor). This study was conceived by all of the authors, where I carried out the necessary research experimentations and completed the writing of literature. Furthermore, the work presented in Chapter 5 was carried out in collaboration with a research project within the department of computer science at the University of Reading.

Chandresh Pravin
July 2023

Acknowledgements

First and foremost, for his endless guidance and support, both academically and personally, I would like to offer an immense thank you to my primary PhD supervisor, Varun Ojha, without whom this thesis would not have been made possible. Your direction and counselling during my PhD studies has taught me, amongst many other things, the research approaches required to complete this work. I would also like to thank Giuseppe Nicosia and Ivan Martino for sharing their knowledge and advice on embarking into the world of academia, I have learnt a great deal from their discussions on research topics.

I would like to show my particular appreciation to Anna Kristensen, for her untiring belief in me and for her availability to decipher my ideas. I would also like to thank Olivia Bashiri, for helping to read the thesis from a perspective that was not my own. I would like to thank my fellow PhD colleague Mohamed Ihmeida, for sharing the research journey with me. Our almost constant daily discussions on research and academia were especially comforting for me. I thank you all endlessly.

I would like to end by thanking the EPSRC and joint industrial partners, Kosnic ltd., for their financial support throughout this PhD. It has been a privilege to work in association with the research council, who have provided both the financial freedom to explore academic topics, and opportunities for development. Working in collaboration with Kosnic ltd. has also led to my development both academically and professionally. I am indebted to the donors for providing me this opportunity.

I, finally, would like to show my appreciation to all my friends, colleagues, and particularly my family. Without your support I would not be who I am today, and this PhD journey would not have been possible for me to reach - Thank you all.

Abstract

This PhD thesis investigates the relationship between network architectures and the robustness against adversarial attacks using a novel methodology that considers both aspects as part of the robustness analysis. Through an investigation on the adversarial targeting of neurons, specifically in the first convolutional layer of a deep neural network (DNN), we observe a relationship between neurons that affect the test accuracy of the DNN, when inferring on a clean, unperturbed dataset, subsequently characterising them as fragile, and those neurons targeted by a potential adversarial attack. We show how the fragile neurons of a DNN convolutional layer evolve over the network training procedure and propose an algorithm to show the targeting of fragile neurons by adversarial attacks. Using the developed adversarial targeting algorithm we show that adversarial attacks focus on specific components of the convolutional layer, framing the adversarial perturbations as attacks on fragile neurons. The task of analysing the robustness of DNNs, thus, leads us to the identification of fragile and non-fragile network parameters, where non-fragile refers to any parameters that do not degrade the performance when subjected to perturbations, as opposed to fragile parameters that do degrade network performance. When discussing perturbations, we consider both variations to the network parameters and the input dataset, in the form of adversarial attacks.

We further extend the analysis to characterise the parameters of deep neural networks as either *fragile*, *robust*, or *antifragile*, and show that network accuracy is impacted negatively, invariantly, or positively w.r.t. defined global and local robustness scores that are computed using a baseline network performance. We design a signal processing technique in the form of *synaptic filters* that identify the fragility, robustness and antifragility characteristics of deep neural network parameters. We subject a network to synaptic filters and compare the network responses for both clean and adversarial datasets, subsequently exposing parameters targeted by the adversary. Our results identify the *structural fragility* of network architectures and show how they evolve over the training process, thus informing us on the *learning landscapes* of DNNs. We find that, for a given network architecture, global and local filtering responses have *invariant features* to different datasets over the learning landscape. Vice-versa, for a given dataset we identify invariant features across different network architectures. Our

proposed analysis of fragility, robustness and antifragility of deep neural networks is useful for designing compact networks by removing particularly the antifragile parameters. We improve the adversarial robustness of networks using a selective backpropagation method that, upon identification of parameter characterisations, retrains only the robust and antifrgaile parameter updates, whilst omitting the fragile parameter updates during the training procedure.

Following this, we develop DNNs for two novel, real-world applications; a DNN designed to identify the the optimum denoising filter for noisy ECG waveforms, and DNNs designed to classify human activities and motion intensities from signals measured using an ultra wide-band radar system. We use original datasets for both tasks and develop novel DNN architectures for the classification tasks. Subsequently, we apply the developed selective backpropagation method to train the custom-designed DNNs and observed an increase in adversarial robustness for the DNNs evaluated. Furthermore, for both the signal denoising filter selection and activity classification tasks, we discern an improvement in the test accuracy when applied to the clean, unperturbed dataset. We successfully show that the proposed selective backpropagation method is capable of improving the adversarial robustness of networks, and in certain instances, also the regular test accuracy. Supporting results for these findings are presented across the chapters of this thesis.

Table of contents

List of figures

List of tables

Nomenclature

Acronyms and Abbreviations

ANN	Artificial neural networks
CNN	Convolutional neural network
CSG	Cumulative Spectral Gradient
dB	Decibels
DNN	Deep neural networks
DWT	Discrete wavelet transform
ECG	Electrocardiogram
FGSM	Fast gradient sign method
FIR	Finite impulse response
ICA	Independent component analysis
ML	Machine learning
PCA	Principle component analysis
RMSE	Root-mean-square error
SNR	Signal-to-noise-ratio
UWB	Ultra-wideband

Variables

α	Synaptic filtering parameter

α_0	Minimum synaptic filtering threshold value
α_A	Maximum synaptic filtering threshold value
$\bar{S}$	Non-fragile neurons
β	Network response to be combined
Δ	Step size
δ	Adversarial attack
Δ_α	Synaptic filtering threshold step size
Δ_ε	Adversarial perturbation step size
ε	Adversarial attack magnitude
ε_0	Minimum perturbation magnitude
ε_E	Maximum perturbation magnitude
η	Scaling factor
γ	Radius of adversarial robustness neighbourhood
$\hat{y}_\varepsilon$	Output for a perturbed input
$\hat{y}_x$	Output for an unperturbed input
$\mathbb{R}$	Set of real numbers
$\mathbf{W}$	Network parameters (pre-filtering)
$\mathscr{L}$	Loss function
$\mathscr{S}$	Set of filtered networks
μ	Mean
ν_s	Sampling frequency
$\odot$	Element-wise product operator
ω_c	Filtering hyperparameters

$\overline{\phi}$	Baseline network response
ϕ	Number of parameters filter by synaptic filtering
π	mathematical constant
Σ	$n \times n$ non-negative diagonal matrix
σ	Standard deviation
τ	Filtering threshold
$\mathbf{T}_f^{-1}$	General inverse transform operator
θ	Network parameters
$\theta^{(l)}$	l-th layer network parameters
θ_α	Network parameters filtered to a threshold
$\tilde{\theta}$	Modified network parameters
ε	Robustness threshold
$\widetilde{\sigma}$	Singular value threshold row
$\widetilde{K}$	Reconstructed weight matrix
B	Adversarial robustness neighbourhood
c_1	Constant 1
c_2	Constant 2
f	Deep neural network function
$g(\cdot)$	Network response combining function
$g(k)$	1D Gaussian noise
h	Set of filters
h_1	Ideal low-pass filter
h_2	Ideal high-pass filter

h_3	Bi-directional pulse-wave filter
H_f	Convolution filter function
K	Number of classes
$K_{\bar{S}}$	Matrix of non-fragile neurons
K_S	Matrix of fragile neurons
l	Network layers
N	Signal Length
r_0	Signal filtered using an elliptical filter
r_1	Signal filtered using a wavelet filter
r_ε	Difference of parameter score between clean and adversarial datasets
r_{x_ε}	Parameter score of adversarial dataset
r_x	Parameter score of clean dataset
U	$m \times m$ unitary matrix
u	Wavelet coefficients
V^T	$n \times n$ transposed unitary matrix
x	Network input
$x(k)$	Discrete signal
x_ε	Adversarial example
y	True class label
$z(k)$	Noisy signal
S	Fragile neurons

Chapter 1

Introduction

Machine learning (ML) algorithms have been widely adapted to various tasks and domains, achieving significant performances in both real-world applications and in numerous research environments [1–3]. Tasks, that decades ago were thought to be exclusively solvable by humans alone, are now considered standard practice for learning systems [4]. In particular, supervised deep learning, a subclass of machine learning algorithms making use of deep neural networks (DNNs) [5, 6], have attained near human-level performance on applications ranging from healthcare [7], speech recognition [8], computer vision [9], physics modelling [10], and many others [11–14]. Moreover, we are increasingly seeing the emergence of DNNs in areas such as text-to-image generation, which only a few years ago, were considered not possible by artificial intelligent agents [15]. Increasing advances in the field have justifiably been a cause of optimism for the emergence of artificial general intelligence: the ability for an intelligent agent to learn any intellectual task that a human can [16].

It is, however, important to establish the current distinctions between artificial intelligence and human intelligence if effective progress in the field is to be made. The notion that an artificial neural network (ANN), a general structure of DNNs without the condition of deep layers, is related to a biological neural network is widely invalidated for numerous reasons [17]. This distinction between artificial and biological neural networks permits us to investigate the functionality of ANNs, and by extension DNNs, using novel methodologies to better understand the strengths and natural pitfalls in artificial learning agents. To develop better, more powerful classes of DNNs, we must first understand the critical weaknesses of current network architectures and address them comprehensively. A natural entry point into evaluating the performance of DNNs is using semantically equivalent data that differ from the dataset learned by the network, sometimes described as counterfactual examples [18]. In evaluating networks to variations of inputs with similar or equivalent meaning, we are,

in essence, assessing the ability of networks to retain the significant information of the dataset, during the learning process. This introductory chapter will outline the study by first contextualising the field and background of the thesis, followed by stating the research problem, the aims, the objectives, the questions, the significance of the study, and finally, the limitations.

A specific subset of counterfactual examples, with the conditions of imperceptibility, are *adversarial examples*, which have been thoroughly investigated in literature to evaluate network robustness [19, 20, 18, 21]. To contextualise adversarial examples, they are the product of an adversarial attack applied on a DNN by an antagonistic adversary and is a particular realisation of noise applied to an input example, subsequently termed adversarial example. Such attacks on a network are often imperceptible and can be described as malicious distortions applied to the input of DNNs, intended to cause the network into missclassifying or incorrectly predicting otherwise correctly classified or predicted inputs. Adversarial attacks on DNNs were proposed to exploit network weaknesses and are used to evaluate the performance of DNNs to critical instances of noise.

In employing adversaries to evaluate networks, we may leverage the property of adversarial attacks to identify input features that are most susceptible to variations, with respect to the performance of the network on a specific task. Such attacks offer an insight into the vulnerability of networks and are not strictly a theoretical phenomenon; various instances of adversarial examples have also shown to appear in real-world settings [22, 23]. Furthermore, adversarial examples present a possible security threat for practical deep learning applications, such as facial recognition, autonomous driving systems [24], or automated medical imaging. The study of adversarial robustness detailed in this thesis is directed at an analysis of DNNs under adversarial settings and parameter filtering, such that we can identify the critical weaknesses of network architectures using adversarial attacks. In doing so, the aim is to highlight significant features of DNN inputs, as well as network parameters, to ultimately develop more robust network architectures.

Frequently we find that DNNs deployed in real-world environments, as opposed to research environments where variables are often tightly controlled, exhibit inherent randomness that is difficult to predict and embed within networks. Adversarial examples intrinsically represent the critical cases for DNNs, and an analysis of networks under such conditions allows us to estimate the worst-case robustness of networks in real-world settings. The critical analysis of networks thus offers a unique viewpoint from which to evaluate networks compared to other methods of analysis, such as random testing, where it can be difficult to distinguish a network that fails one time in a million trials from a network that never fails.

Adversarial attacks, however, permits us to distinguish the difference between networks that seldom fail and networks that never fail. Given that a sufficiently powerful adversary is incapable of bringing a network to failure, we may assume that the network will not be brought to failure by unforeseen randomness experienced in real-world applications [25]. To follow this reasoning, we propose enhancing the robustness of DNNs to the randomness exhibited in real-world applications through identifying and subjugating the critical weaknesses of DNNs to adversarial attacks.

There have been numerous methods proposed to design adversarial attacks [26]. With every iteration of a new adversarial attack put forward, a corresponding adversarial defense method has been suggested to circumvent the attack [27]. Methods designed to form adversarial attacks can vary depending on input constraints and information on the target network. We may broadly categorise adversarial attacks into two groups: white-box attacks, where the adversary has complete knowledge on the target network, and black-box attacks, where the adversary has no knowledge on the target network. Adversarial defenses, much like adversarial attacks, can be divided into different categories: (i) defenses focusing on gradient masking/obfuscation, whereby gradients of the network weight used by adversaries to form attacks are disguised, (ii) robust optimization [28], where the network parameters are altered to increase adversarial robustness, and (iii) adversarial example detection, where the goal is to detect an adversarial input and process this entity differently to ordinary inputs [29]. The task to build a defense model able to remain unbeaten by an ever-growing selection of adversarial attacks has proven to be a difficult task [30, 31].

When the goal is to make DNNs more robust to adversarial attacks, indeed adversarial defense methods have proven to be an effective tool in various instances [32], none more so it should be noted, than adversarial training [33, 34], whereby the critical weaknesses of DNNs are embedded within the network. When the goal is to understand how to devise robust DNNs, however, one should be cautious to employ defense methods without identifying and understanding the particular robust reinforcements, brought about by various defense methods. It may be the case that a defense strategy shows to improve network robustness without fully addressing the cause of the critical weakness that it has defended against. To use a biologically inspired analogy, in such circumstances, an effective defense method may be acting as a plaster repressing a wound. Even in assuming that we have effective defense methods, a procedure is required to evaluate the effects of the defense on the DNN and to highlight the critical weakness of the network, if we are to develop better, more robust networks.

Evaluating a network under an adversarial attack is one possible course of action in measuring the variability of network performance to sub-optimal conditions. Adversarial attacks, however, are also a limiting factor of network analysis as we are restricted to changes explicitly to inputs of the network. To holistically analyse a system, DNN and input included, we must also investigate the variability of network performance to changes in the network architecture itself. A DNN that has learned dataset noise, or acquired weak features of the data, is scarcely considered to be fully optimised, and is often referred to as network overfitting [35, 36]. Identifying the influence of learned representations on network performance also exposes vulnerabilities of the network. Indeed, it has been shown that the invariability of DNN performance to adversarial attacks and network overfitting are intertwined [37].

The analysis of DNNs in this thesis is carried out using a signal processing methodology that evaluates DNNs using various forms of stress applied to the system; broadly categorising the stress as internal and external to the network. Where internal stress relates to variations/perturbations of the network parameters that govern the functionality of DNNs on a specific task, and where external stress is concerned primarily with variations/perturbations to the environment (input noise) within which the DNN is applied. We further specify the external stress on DNNs to be of an adversarial nature, such that the focus is put on highlighting the critical weakness of DNNs. In analysing DNNs, we use the terms *fragility*, *robustness* and *antifragility* as characterisations of the composite components that form DNN systems with respect to DNN performance.

When referring to fragility, we broadly define a system that exhibits a *decrease* in performance for a given task, when subjected to stress, be it internal and/or external. When discussing robustness on the other hand, we define a system that, despite being subjected to internal and/or external stress, shows an invariance in performance for the specified task. Following this line of reasoning, we define antifragility to a system that, when subjected to internal and/or external stress, exhibits an *increase* in performance for the task [38]. In the following thesis we offer context to the aforementioned themes of internal stress, external stress, network performance, fragility, robustness and antifragility.

Applications of the proposed robustness analysis of DNNs are directed towards networks developed for real world tasks. In this thesis, we consider specifically the task of human activity recognition using ultra-wide (UWB) radar systems [39, 40] and associated preprocessing of inherently noisy radar signals [41, 42]. DNNs designed for applications using UWB radar systems are required to be invariant to the various forms of noise commonly found in radar acquired data [43]. We work to analyse the adversarial robustness of novel DNN

architectures designed with newly collected datasets, in order to improve the performance of the DNN models. Bridging the gap between DNNs designed under controlled, research environments and those built for real-world datasets is one of the objectives of the work developed in this thesis. Through applying the proposed analysis on custom-designed DNN architectures on original datasets, we are able to evaluate the effectiveness of the analysis at developing stronger networks that are able to perform, even under sub-optimal conditions.

Aim of Thesis Adversarial attacks and defense mechanisms have been at odds in literature since the inception of robustness analysis of DNNs under adversarial settings. Assuming that we even have a defense mechanism that is effective against a large subset of all possible adversaries, little evidence of such methods show to exist in literature, however; the need for analytical methods to identify the specific reinforcements generated by the defense, is still required. In this thesis, we aim at developing an experimentally derived analytical method for DNNs that is capable of identifying the strengths and weaknesses of different network architectures, to various methods of internal and external stress. The proposed analysis of DNNs is intended to aid in the development of more robust DNN architectures, through the identification and post-processing of specific, analytically informed, and isolated network components. To efficiently characterise the strengths and weaknesses of different networks, we utilise the notions of fragility, robustness and antifragility [38] to describe DNNs that, when subjected to stress, respectively either decrease, remain invariant or increase in performance for a specified task. We develop the work in this thesis to investigate the connections between DNN architectures and adversarial robustness through an evaluation networks that considers both external and internal stressors, in adversarial attacks and network parameter perturbations.

The resulting methodologies will be used on DNNs designed for real-world applications, such that networks trained on limited and noisy datasets with limited resources are still able to perform sufficiently. This is particularly useful for applications where data preprocessing may be resource expensive and the useful features of the datasets are subdued by randomness from the environment. Developing DNN models capable of performing sufficiently within such applications and environments will lead to the advancement of deep learning beyond the realms of research laboratories. Applicable areas where such robust DNNs would be used include, but are not limited to, DNNs for security [44], monitoring applications using radar systems [45], speech recognition [46, 8], and computer vision [9]. We aim to apply the developed analysis to identify and improve weaknesses in DNN architectures, such that we develop stronger networks able to withstand greater variations to the operating conditions.

1.1 Research Problems and Contributions

Identifying and addressing the critical weaknesses of DNNs is the focus of the following thesis. We employ adversarial attacks to expose the fragility, robustness and antifragililty of DNNs to external stress, representing a change to the environment within which the DNN operates. Equally, we investigate the effects of internal stress on network parameter, which is when the internal components of networks are altered in order to measure the affects of the internal perturbations on the network performance. To better understand the effects of the stress analysis, we use the characterisations of fragility, robustness and antifragility to describe how DNN performances vary as a consequence of the applied stress. Using the different characterisations of DNN performances, we identify and consolidate the constitute fragile, robust and antifragile components. In carrying out such an analysis, we aim to establish relationships between adversarial robustness and network architectures through an experimental analysis of DNNs under adversarial settings. Specifically, the significant problems addressed, and contributions made, in this thesis are summarised below.

- We formalise, and discuss the rationale for, carrying out an analysis of DNNs through a dichotomy of internal and external stress to identify the weaknesses of different network architectures for various datasets. Establishing an experimental procedure that considers both network input and architecture is essential in comprehensively analysing DNNs, as to evaluate the learning system from opposing, yet associated, directions. A central basis of carrying out a stress analysis of DNNs is to subject networks to sub-optimal conditions and subsequently measure the variations in network performances for different datasets. Internal and external stress represent sub-optimal conditions for networks; in this thesis, we formalise these concepts within the context of DNNs and show how they can be realised for different networks and datasets. We take internal stress to be perturbations of the network architecture and we develop a parameter filtering methodology that systematically removes network parameters. The proposed parameter filtering is agnostic to variations of network architectures, thus, can be used on any type of DNN using weight parameters. We take external stress to be perturbations to the network input, and specifically, perturbations of an adversarial nature. In establishing a standardised analysis method for DNNs, we are able to compare the DNN responses of different network architectures and datasets, subsequently highlighting the properties and characteristics of networks that are more robust compared to others.

- Thereafter, we construct a methodology for characterising the performances of DNNs, to internal and external stress, using the notions of fragility, robustness and antifragility periodically through the network training process. We formulate a scoring method to encapsulate the effects of the stress analysis on DNNs and detail the conditions used to characterise the scores as fragile, robust or antifragile. We define fragility to be the condition where a network, when subjected to internal and/or external stress, shows a decrease in performance for a given task. Robustness is defined as the condition where, when a network is subjected to internal and/or external stress, the performance of the network for a specified task remains invariant, within a bounded threshold. We introduce the notion of antifragility, within the context of deep learning, to be the condition where, when a network is subjected to internal and/or external stress, we observe an increase in performance for a given task. The conditions of fragility, robustness and antifragility are used to categorise network architectures, using the developed notions of internal and external stress.

- In the final analysis, we employ the proposed fragility, robustness and antifragility network characterisations to develop more robust networks through an informed network training procedure. Upon characterising the network components, we selectively use backpropagation on only the robust and antifragile network parameters. Network parameters characterised as fragile are omitted from the further re-training after characterisation. We provide a reasoning behind the *selective backpropagation* procedure and present results associated with the fragile, robust and antifragile characterisations. We empirically observe that periodic characterisation of network parameters, using the proposed mythologies, allow us to apply an informed fine-tuning of specific network parameters, which in turn, results in networks that are more robust to adversarial attacks. We demonstrate the effectiveness of the proposed methodology on a real-world application: the classification of human activities and intensities using 24 GHz UWB radar systems and deep learning. We work with a novel radar signal dataset that is recorded for a real-world application, and as such, is limited in nature and contains various irregularities. The proposed methodology shows to improve the performance a deep learning system on a real world application. This leads us to argue that inherent weaknesses of DNNs, and the stronger counterparts, inform us on the functionality of networks.

1.2 Thesis Outline

This thesis is organised as follows. In Chapter 2, we provide a review of the related works in literature and offer a technical background on the topics of DNN architectures, adversarial robustness, network compression methods, and the relationships between the aforementioned subjects, which are required for the ensuing works in Chapter 3. In addition, for Chapters 3, Chapter 4, and Chapter 5 we provide related works specific to the contents within each chapter. Furthermore, in Chapter 2 we provide the related works on the applications of robustness analysis that are investigated in Chapter 5. Following this, in Chapter 3 we delve into the novel aspects of robustness analysis where we focus on a sub-section of a network; the first convolutional layer, to identify fragile and non-fragile neurons (filter kernels). The work presented in Chapter 3 details the formulations of the adversarial attack employed and the developed parameter filtering procedure used to identify fragile and non-fragile parameters.

The work from subsequently Chapter 3 extends onto an evaluation of the complete network in Chapter 4, where we also expand on the definitions of fragile and non-fragile parameters to adopt fragile, robust, and antifragile network parameter characterisations. The expansion on the definitions of fragile and non-fragile, as discussed in Chapter 3, to the characterisations of fragile, robust, and antifragile in Chapter 4 permits us identify network parameters that also increase in performance when subjected to adversarial attacks and parameter filtering, as opposed to the performance only decreasing (fragile), or remaining invariant (non-fragile). The implications of this is also discussed further in Chapter 4 where we outline a training regime that only trains specific parameters of the network, termed selective backpropagation. In addition, we provide the formal definitions of network performance, internal stress and external stress with respect to adversarial attacks and parameter filtering, fragility, robustness, and antifragility within the context of DNNs.

In Chapter 5 we outline a novel signal denoising filter selection algorithm based on a new DNN architecture that is capable of predicting the optimal signal denoising filter, given a noisy unprocessed waveform for applications where focused data preprocessing is unavailable. Further to this, we outline a second real-world task comprising of an original radar signal dataset, and a purpose-built DNN architecture for human activity and activity intensity classification. The contents of Chapter 5 lead us to employ the selective backpropagation methodology developed in Chapter 3 and Chapter 4, such that we improve the performance of the DNN applications presented in Chapter 5, for both the adversarial dataset and the regular

dataset. We conclude the thesis in Chapter 6 by summarising our work and highlighting the future directions of work instigated by that presented in this thesis.

1.3 Publications

Upon expanding on the findings from Reference [47] in Chapter 4, we aim at applying the analysis on DNNs developed for real-world applications. This thesis addresses two such DNN applications using novel datasets and networks, the first of which is from Reference [42], and details the study of classifying optimal filtering methods for noisy radar signal denoising. In the work we train a 1-dimensional convolutional neural network (CNN) to predict the optimal denoising filter, from an elliptical filter and a wavelet filter, given a noisy ECG signal that emulates signals acquired using a radar system. The second application of the analysis method developed in this thesis, is to classify human activities using UWB radar system and a custom-designed DNN. The denoising filter selection algorithm developed in Reference [42] is also used as the preprocessing step in the activity classification model. The development of the novel DNN and accompanying custom dataset for human activity classification has been come to fruition as the result of a collaboration with an industrial partner and has been the subject of product patenting by the industrial partners. Furthermore, much of the work presented in Chapter 4 explores and extends on the findings developed in Reference [47], has been submitted to the Journal of Artificial Intelligence, pending review.

Source Code We provide access to all code and instructions required to replicate the methods outlined in this thesis. For training the DNN models, we reply primarily on the PyTorch 1.8.2 [48] and Torchvision 0.11.0 [49] libraries running on the Python 3.7 language. The adversarial attacks used on the implemented models were generated using auxiliary PyTorch functions. Techniques used for the computation of parameter scores, that are used to characterise network parameters, are implemented using the standard PyTorch and NumPy 1.23.3 [50] functions. The full implementable code for the proposed methodology can be found at https://github.com/SynapFilter/InferLink.

Chapter 2

Background

In this chapter, we provide a review of previous work related to the topics of focus in this thesis; adversarial attacks, robustness in deep learning and network ablation. The notion of robustness in deep neural networks, and adversarial robustness in particular, has been an active area of research within the field of deep learning since the works of Biggio et al. [20], Szegedy et al. [19] and Goodfellow et al. [51]. Many subsequent works have been spawned from the early explorations of the adversarial phenomenon on DNNs [52–54, 23]. There is vast literature available on the broader topics of adversarial attacks, adversarial defenses and DNN robustness. To avoid covering the full spectrum of literature in the field, we focus our attention on the seminal works relating to techniques of measuring the robustness of DNNs with a focus on adversarial attacks and network ablation. More in-depth reviews of adversarial attacks, defense methods, and network architectures are presented in [31, 26, 27, 55], whilst literature on robustness analysis and verification techniques can be found in [56, 57]. Further to detailing the seminal works relevant to this thesis, we conclude the chapter with a review of related applications of robustness analysis in making DNNs more robust, as these works constitute the benchmarks for our methods, discussed in Chapter 3 and Chapter 4.

It should be noted that this thesis focuses primarily on the robustness of DNNs to various methods of stress, both under adversarial settings and to network parameter filtering. The topic of robustness analysis in machine learning (ML) is an extensive topic, and there exist various techniques and methodologies on evaluating DNN robustness in literature. Further investigations into the robustness analysis of DNNs against labelling errors [58]; poisoning attacks [59]; principle component analysis (PCA) bounds [60]; distributional shifting adversaries [61]; and statistical robustness [62, 63] are available in literature.

This chapter is organised into Section 2.1, Section 2.2 and Section 2.3 that provide seminal works relating to the themes explored in this thesis. In Section 2.1 we discuss DNN architectures, and specifically, methods relating to achieving network compression, robustness and how compression and robustness interplay. In Section 2.2 we detail exclusively adversarial robustness of DNNs, with reference to commonly use defenses and analytical approaches to evaluating the robustness of networks. In Section 2.3, we explore the applications of robustness and lay the basis of how the proposed robustness analysis is intended to be applied on novel, real-world networks and datasets.

2.1 Architectures of Deep Neural Networks (DNNs)

Deep neural network architectures are an extension of the general neural network structure that consists of multiple (greater than three) layers between input and output. Shallower neural network architectures have faced various challenges in approximating more complex problems, such as tasks with sparse and local approximation solutions [64]. The increased number of layers in deep neural networks are effective in representing higher complexity functions [65], compared to shallower neural network architectures, however, the use of DNNs bring with it further constrains regarding data and model tuning [66]. DNNs have since been applied in numerous domains: speech recognition [67, 68], healthcare [69, 70, 42], computer vision [71], and human computer interaction applications [72, 73]. An overview of the challenges and directions of work regarding DNNs can be found in the following works [74, 7, 75]. The following work focuses on the analysis of DNNs, with the motivation of understanding the weaknesses and strengths of network architectures. The ultimate objective is to develop stronger networks capable of performing sufficiently, even under sub-optimal conditions.

Recent DNNs are comprised of a variety of different layers that can be broadly categorised into the following: input layers [76], convolutional and fully-connected layers [77], sequence layers [78], activation [79], normalisation [80], pooling [81], object detection [82] and output layers [83]. Whilst all of the different types of DNN layers address specific requirements of networks, we focus our study on specifically the convolutional layers and fully connected (linear) layers. The convolutional layers transform the input in order to extract high-level features for inference, whilst the linear layers are the classifiers that connect the features to outputs.

In this thesis, the internal analysis of DNN architectures is directed towards the convolutional layers of networks. Even considering only the convolutional layers of networks, the

works of Zhang et al. [84] show that not all layers are equal in importance to the network. Studies have previously investigated layer importance in DNNs, the works of [85] investigate layer-wise pruning based on explainability of networks. An application of the robustness analysis methods explored in this thesis is to identify the the layer-wise importance of network layers grounded on adversarial vulnerabilities of the layers.

This section is divided as per the following: in Section 2.1.1, we detail the adversarial robustness of DNN architectures, Section 2.1.2 presents how compressing network architectures affect the performance on specified tasks, and in Section 2.1.3 we relate network compression to the adversarial robustness of different networks.

2.1.1 Robustness of DNNs

Despite the wide use of DNNs and extensive applications in numerous fields, several foundational properties of DNNs are yet to be deciphered and have been the subject of analysis in literature for some time now. Particularly, the robustness of DNNs to different forms of perturbations has been the topic of increasing attention, due to the importance of DNN applications in safety critical systems, such as medical image classification and autonomous driving implementations [86]. The predominant definition of perturbations when analysing the robustness of DNNs has been directed to the network inputs [19, 51]. The vulnerabilities of networks to small input perturbations can be encapsulated in adversarial examples and attacks. A more detailed review of adversaries is presented in Section 2.2. To elaborate briefly however, adversarial robustness relates to small, carefully crafted perturbations applied to the input of networks that subsequently result in a large variation in the performance of different network architectures [87, 26]. Such perturbations provide novel insights into developing better networks that are more resistant to input variations. The ability, or inability in most cases, of a DNN to resist perturbations can be attributed to the structure of the network itself. Indeed, through an analysis of polynomial classifiers, the works of Fawzi et al. [88] suggest that higher-degree classifiers, relating to the complexity of the network architecture, tend to result in classifiers that are more robust to perturbations. The relationship between network complexity and robustness can be realised by the increased adversarial performance observed in DNNs with deeper layers [89]. The depth of networks have shown to improve both regular and adversarial performance on a variety of tasks [90, 91]. Currently, however, there exists no universally accepted network architecture that can be considered robust to all forms of adversaries that may be formed.

We consider the perturbations applied to network inputs as an external stressor on a network; the change brought about by the perturbation directly affects the environment exposed to the network, hence labeled as external. Following this reasoning, naturally we must also consider internal stress on DNNs, if we are to be comprehensive in the analysis [92]. We define internal stress to be systematic perturbations applied to the parameters/connections or nodes/filters of network architectures. The objective of perturbing network parameters, as opposed to the input of networks, is to evaluate the impact of the learned network representations on the task performance. The dichotomy of the internal-external stress on systems has previously been explored in the biological sciences [93, 94], and in this thesis, we take inspiration on the notions of internal and external stress developed for biological systems. In particular, the works of Oken et al. [95] describe stress on biological systems as the result of perturbations arising from the internal or external environment, henceforth referred to as stressors. The variation in network performances to adversarial perturbations on inputs (external stress) is used to evaluate the critical features of networks with respect to the input domain. In analysing the variation of network performances to parameter perturbations [96, 97], we are in essence evaluating the critical features of the network with respect to the network architecture. The works of Molchanov et al. [97] developed a method for estimating the impact of network parameters on network performance, and importantly, show that composite filters from various network architecture can be removed whilst retaining performance for a given task.

2.1.2 DNN Architecture Compression

The ability of deep neural networks to perform a specified task to near-human levels relies on various factors, an important one of which, is the depth of network layers used. The question as to whether DNNs need deeper layers in order to perform on tasks sufficiently, has been explored in various studies in the field [98]. The work of Urban et al. [89] suggest that DNNs are required to be both deep and convolutional if they are attain high levels of performance on more complex problems. Deeper network layers result in more resource hungry networks, which can be a limiting factor for many real-world applications. The works of Zhang et al. [84] show that not all layers, however, are critical to network performance, thus, leading us to the study of network compression [99]. Network compression deals with reducing network size without greatly affecting network performance. As deep learning researchers begin to push the limits of big compute [100], compression techniques have become a particularly important area of study in deep learning research for developing more efficient architectures.

To investigate methods of making networks more efficient, the topic of network compression naturally leads us to network pruning methods [101]. Network pruning tackles the task of over-parameterised DNNs through two primary directions; network parameter-based methods [102] that focus on the weights of DNNs, and unit-based methods [103, 104] that consider individual neurons for removal.

A different group of techniques for network compression are quantisation methods [105], where the goal is to reduce the number of bits representing network weights. This can be achieved through approaches such as binary characterisation of weights, low-precision representations, and parameter quantisation. Another notable way in which practitioners have achieved network compression is through low-rank factorisation methods [106], where matrix or tensor decomposition methods are employed to determine and retain only the most informative network parameters. The general objective of pruning methods is to achieve sufficient network compression through eliminating unimportant, redundant, or unnecessary network components, which can be either parameters or neurons. In turn, a network processed through a good compression method is expected to perform similarly to the original, pre-compressed network, with a smaller parameter load and decreased computational costs [107].

In the following thesis, we employ various methods of network compression, from unit-based pruning [104], as presented in Chapter 3, to applications of low-rank factorisation methods [106, 47], also presiding in Chapter 3, and network parameter filtering, as detailed in Chapter 4. Network compression has also been a topic of increased attention in research, as practitioners attempt to extract the far-reaching capabilities of DNNs without expending equally extensive resources. Applications of network compression is not only limited to research, in practice, real-world use cases of DNNs benefit greatly from compressed networks that consume less power, occupy less memory and require fewer resources to train [108]. In essence, network compression appears to be a natural progression of deep neural networks, both with regards to efficient resource management, and explainability of deep neural network decisions [85]. One may come across a resource constricted task, for which, a suitable yet over-parameterised DNN is readily available, and consider this an ideal circumstance to deploy any one of the numerous network compression techniques that are available in literature [99]. A side of caution should be taken when confronted with such a scenario, as there are various other challenges to consider when deciding which, or if, to use compression methods on a given network [107]. For instance, the complexity of the objective task may be safety critical, in which case, network compression may degrade the network performance on a particular type of detrimental of noise, such as naturally occurring

adversarial examples [109]. For such scenarios, we must consider network robustness in conjunction with network compression.

Although the purpose of this thesis is not primarily associated to network compression, the notion of internal stress developed throughout this work is strongly influenced by network compression techniques [106, 104]. We consider an internal stressor on DNNs to be any form of perturbation applied to the network connections/parameters or nodes/filters. This definition of internal stressors is closely linked to that of several network pruning methods [101], with a key distinction; we apply internal stress to DNNs in order to investigate the effects of external stress on specific network parameters. The motivations behind perturbing network components in this thesis arise from an analytical perspective, rather than one explicitly aiming to reduce network size. Thus, with the introduction of internal stress to the robustness analysis formulation, we are able to carry out an investigation on the impacts of external stressors, such as adversarial perturbations, on network performance, with an emphasis on the composite components of networks.

2.1.3 Robustness and Compression

Thus far we have discussed robustness of DNNs in Section 2.1.1 and network compression in Section 2.1.2. It is appropriate within the context of this thesis to also review works related to the adversarial robustness, with respect to network compression, as the two themes interplay consistently throughout this work. Network compression has, on numerous occasions, confirmed to be an effective method of network regularisation [110]; a countermeasure to deal with overly complex network architectures that overfit [35]. Ina similar manner, the works of Rice et al. [37] investigate the case of overfitting when networks are trained on adversarial data, such that the attacks are used as part of the training process, and suggest that network regularisation also benefits adversarial robustness under such circumstances. It should be noted however, that in their work, the authors ultimately conclude that early stopping [111], which is a different form of network regularisation whereby network training is concluded based on a validation loss criterion, achieves the most significant gains on adversarial robustness. Studies have previously tackled the question of input and network perturbations simultaneously to analyse robustness [112]. The works of Dhillon et al. [79] investigate a stochastic activation pruning method that is designed to make networks more robust to adversarial attacks. The works of Jordão and Hélio [113] explicitly investigate the effects of pruning on adversarial robustness, and in their work the author conclude that

network pruning not only address issues related to generalisation, but also increases network robustness to adversarial attacks.

In order to combine the topics adversarial robustness and network compression techniques cohesively, let us first consider how effective adversarial robustness can be achieved. One of the most effective methods of making networks more robust to adversarial attacks has been adversarial training [114, 34, 33]. Adversarial training is, within itself, an increasingly studied sub-field of adversarial robustness research [27, 115, 33]. The primary premise of the technique is to include adversarial examples within the network training data, such that the network learns the adversarial perturbations, that it is attempting to resist. Despite being one the most effective, consistent methods at defending networks to adversarial attacks, the performance of the adversarially trained networks on clean images is reduced if careful regularisation of the network is not implemented [116, 117, 24]. The works of Xie and Yuille [118] showed the effectiveness of adversarial training to improve DNN robustness require deeper networks, as the conventional "deep" neural networks that are currently used in practice are still too shallow for the task of adversarial learning. This necessity for deeper layers for improved performance on tasks is found for clean data, as well as adversarial data [89]. To provide a counter-argument, however, the works of Jordão and Hélio [113] show that shallower networks (where network layers have been post-processed) are still able to provide robustness gains.

It has been observed in studies of network compression techniques, that not all of the composite networks components are required to maintain performance on a task. Indeed, many redundancies and inefficiencies can be found, and subsequently removed from network architectures [99, 104], whilst retaining sufficient network performance [107]. The works of Ye et al. [112] extend on the proposition that network parameters can be removed without significant detriment to performance by applying a compression method under adversarial settings. The authors find that network compression can be applied whilst also retaining adversarial robustness. The works of Croce and Hein [119] also discuss the interplay between adversarial robustness and network architecture components. Principally, the authors suggest that some network architectures appear better suited for robustness and that careful modifications of non-robust architectures may significantly improve robustness and generalisation to unseen attacks. In this thesis, we utilise the findings of [112, 104, 85, 102] to form the basis of our study; we propose the use of network compression techniques, such as pruning methods and low-rank factorisation, to inform us on how to retain, and improve, adversarial robustness of networks.

2.2 Adversarial Robustness

Adversarial robustness relates to the ability of ML models to resist adversarial attacks, which is a broad category of methods developed to compute adversarial examples, using the inputs of ML models. Adversarial examples, in turn, are the result of an adversarial attack on an input, can be loosely defined as inputs corrupted with noise that have been intentionally crafted by an attacker, an adversary to the network. The motive of the attack is to deceive learning models into incorrectly predicting or missclassifying otherwise correctly predicted or classified inputs. In this thesis, we consider adversarial attacks as external stressor on a network, as the effects of the adversary, despite often being calculated using the network itself, is applied to the input of learning models. Whilst the origins of adversarial attacks and examples within the context of ML cannot be attributed to a single study or author, the findings by Szegedy et al. [19], who were initially examining image recognition networks to determine explainability of network decisions, prompted an accelerated interest on adversarial attacks within the field. Intriguingly, the authors observed that, guided by network gradients, performing minute and precise manipulations to correctly classified inputs subsequently resulted in adversarial examples that were able to fool the network into misclassifying the previously correctly classified input. Further to this, it was also found that the input manipulations could be focused to force the network into misclassifying to a specifically different class, also referred to as a targeted attack [120, 121].

Formally, for a given trained network f and a correctly classified input image x, it was possible to calculate a manipulation δ_ε with magnitude $\varepsilon \in \mathbb{R}$ that is a form of noise generated using gradient information of the network, and that results in an adversarial example x_ε, such that $f(x) \neq f(x+\delta_\varepsilon)$ and signifying a missclassification of the input. The magnitude of the noise ε in question need not be large, relative to the initial input x; to the human eye, differences between x and x_ε are often imperceptible.

Since the formal conception of adversarial attacks and examples in deep learning, various studies were carried out for different applications and learning models, including but not limited to: speech-to-text translation [121], radar image recognition [122], natural language processing [123] and healthcare [124]. Numerous approaches were, and continue to be, proposed to design adversarial attacks [30, 26, 53], form defense mechanisms against attacks [125, 27, 32], and attempt to explain the reasons for the effectiveness of adversarial attacks [18, 126, 51].

Perhaps the most interesting aspect on the effectiveness of adversarial attacks is the relative ease, from the perspective of the magnitude of input noise required, with which

even state-of-the-art DNNs can be broken down [51]. When we investigate the methods with which adversarial attacks are formed, primarily using a gradient-guided approach, it is apparent that the very way in which networks learn data distributions is also the facilitator of potential adversaries. Gradient-based learning methods, which are commonly used to train neural networks, are designed to optimise networks training data to generalise on un-seen data, using error metrics to quantify the accuracy of the predictions [127]. Such error metrics are based on statistical assumptions about the distribution of data used to train the network, and do not naturally consider adversarial examples, as is observed in [128]. The ability of a network to predict a dataset does not equate to the ability of the network to predict adversarial data. An independent method of evaluating networks to adversarial data is thus required.

Several methods exist that address the task of evaluating networks under adversarial settings. A commonly used method of evaluating DNNs is to compute adversarial attacks for all input instances in a test dataset and subsequently calculate the ratio between unsuccessful attacks and the total number of attacks applied [88, 129]. To provide provable guarantees in computing such robustness measures, the task can be formulated by establishing a bound on the norm of the adversarial perturbation magnitude ε, to which the network can be considered robust to. Formally, we take an input example x within an l_p-normed neighbourhood $B = \{x + \delta_\varepsilon | \|\varepsilon\|_p < \gamma\}$ around x and given the condition $\gamma > 0$ to verify the following:

$$f(x + \delta_\varepsilon) = f(x) \quad \forall x \in B.$$

Using the above verification [57], we can also compute the largest γ to which the verification is still satisfied. As evident by the variety of adversarial attacks proposed in literature [30, 26, 53, 121], the above verification can only ever provide an approximation on the robustness of neural networks. If an adversarial attack is successful, the above check will fail. If an adversarial attack in unsuccessful, the verification above remains and can either still be valid or not, as other attacks may prove otherwise.

Robustness verification offers a provable method by which neural networks can be evaluated under adversarial settings, and are one direction of analysing network stability to adversarial attacks. If the objective is shifted to not only analysing the robustness of neural networks, but to identifying the specific constituting components of neural networks that are most targeted by an adversary, we must equally adapt the approach with which we form the analysis [130]. In doing so, we extend the robustness analysis to evaluating networks, and also to explain how adversarial attacks work [131]. Adversarial attacks first came to attention through the works of Szegedy et al. [19] whilst the authors were studying network

decision boundaries, and interestingly, there are strong commonalities between adversarial attacks and the interpretability and explainability of machine learning models [132–134]. The interpretability and explainability of neural networks is of particular importance if adaptation of DNNs is to be expected in real-world applications, especially safety critical systems [86, 25]. Adversarial examples too have relevance to real-world applications; in the works of Zhao et al. [109] the authors propose a method to generate natural and legible adversarial examples from real-world applications. There are other works that explore the possibility of adversarial examples in the real world, without the need for an adversary or maliciously designed noise [22].

The focus of this thesis is to utilise the adversarial phenomenon in ML in order to bridge the gap between models developed in research and models designed for real-world applications. Particularly, in Chapter 3 we present a methodology to identify how an adversarial attack targets specific neurons in the first convolutional layer of networks. In Chapter 4, we extend the methodology presented in Chapter 3 to identify all network parameters targeted within various networks, and identify commonalities between different network architectures and datasets. In identifying the components of networks most susceptible to adversarial attacks, we may also fortify the networks through fine-tuning the highlighted elements; we explore improving network performances using the proposed methodologies in Chapter 5.

2.3 Applications of Robustness Analysis

Primary applications of robustness analysis are to evaluate networks to a particular class of noise; adversarial perturbations that often require deliberate, and sometimes resource expensive, computations to calculate [135]. Whilst computationally efficient attacks have previously been proposed in literature [136], the attack perturbations nevertheless represent an unlikely worst-case scenario for inputs, and thus, actively computing adversaries is still required in most cases. It should be noted that adversarial examples have been observed in the physical world [22]. Nevertheless, there remains a relative disconnect between adversarial robustness and the ability of networks to perform under other, more common types of perturbations [137], though several works have investigated this relationship [76, 138, 139]. Notably, Tsipras et al. have shown that robustness achieved through adversarial training comes with a trade-off of standard accuracy. The cause of this inversely proportional relationship between robustness and standard accuracy inherently stems from the difference of features learned by regular networks and more robust networks, trained using adversarial

data. The works of Li et al. [140] draw a connection between adversarial robustness and additive random noise through a certifiable analysis method.

Much of the research carried out on robustness analysis within deep learning research has been based primarily within the domains of image classification, natural language processes and speech recognition [29]. In this thesis, we explore DNN applications on radar systems and signals [141, 72, 142]. Conventional ML approaches to radar signal classification have been widely researched in literature [143]. Although widely used for radar-based classification systems, conventional ML approaches exhibit several drawbacks that hinder improvement in performance, specifically related to random noise robustness and generalisation [72, 144]. We look to evaluate the robustness of novel DNN architectures designed for noisy radar signal classification. In Chapter 5, we consider two tasks related to radar signal processing and classification, which is the focus of the robustness analysis methods developed in Chapter 3 and Chapter 4. The task of robustness analysis, particularly on the adversarial robustness of DNNs developed for radar systems, has yet to explored in depth. The question as to how adversarial attacks can aid in making radar-based classification networks more robust, even to random perturbations observed in practice, is one of the addressed themes in this thesis. This task is particularly interesting due to the sensitivity of radar systems to subtle movements from the target objective, as well as various forms of environmental noise which are both unpredictable and potentially damaging to the performance of the network [43].

Deep Learning for Radar Systems There are applications proposed that incorporate deep learning models with radar systems, including but not limited to: fall detection [145], automated target recognition [146], environmental monitoring [147], vital signs monitoring [148, 40], line-of-sight estimation [45], and human activity recognition [142, 72]. In the works of [142] the authors develop a system for recognising daily life activities such as sitting, standing, picking up objects, drinking water and fall events. The authors detail that optimal results were achieved by using data pre-processing techniques such as PCA and data augmentation prior to application of the classification network. Efficient data preprocessing has been highlighted as a recommended precursor to effective classification. Indeed the task of radar signal preprocessing for use with learning models has been explored in the following works [149, 150, 41]. In this thesis, we focus on the task of human activity recognition using UWB radar and deep learning, as detailed in Chapter 5, relates to the classification of human activities using 24 GHz UWB radar system and DNNs. Furthermore, based on the works of Pravin et al. [42], we apply an automated DNN-based radar signal denoising algorithm to preprocess the developed activity classification network. Both the activity classification

network and radar signal denoising network will be the subject of the robustness analysis method developed in this thesis. We use the proposed robustness analysis at evaluating and improving the adversarial robustness of both networks. We subsequently asses the reinforced networks on the ability to withstand random perturbations, as well as adversarial ones.

Chapter 3

Adversarial Robustness for DNNs: Attacks on Fragile Neurons

In this chapter we identify fragile and non-fragile neurons of deep learning architectures using nodal dropouts of the first convolutional layer. Using an adversarial targeting algorithm, we correlate the identified fragile and non-fragile neurons with the distribution of adversarial attack targeting on the convolutional layer. We define fragile neurons to be those that, when perturbed, reduce the network performance for a given task. Accordingly, we define non-fragile neurons to be those that, when perturbed, do not reduce the network performance on a specific task. It should be noted that in this chapter non-fragile refers to any neurons that do not reduce network performance, which may entail a relative invariance or increase of performance. Further discussions on this is presented in the following Chapter 4.

Adversarial robustness of neural networks has gained significant attention in recent times and highlights the intrinsic weaknesses of deep learning architectures against carefully constructed distortions applied to the input. In this section, we evaluate the robustness of state-of-the-art image classification models trained on the MNIST [151] and CIFAR10 [5] datasets against the fast gradient sign method attack, a simple yet effective method of deceiving neural networks.

Our method identifies the specific neurons belonging to the first convolutional layer of DNNs that are most affected by the applied adversarial attack. We, therefore, propose to make fragile neurons more robust against adversarial attacks by compressing features within non-fragile neurons and amplifying the fragile neurons proportionally. Furthermore, we also propose *back-propagation filters* that address fragile and non-fragile neurons seperately during the network training process.

3.1 Overview

Deep neural networks have been widely adapted to various tasks and domains, achieving significant performances in both the real world and in numerous research environments [1]. Previously considered state-of-the-art DNNs have been subjected to a plethora of tests and experiments in an attempt to better understand the underlying mechanics of how and what exactly these learning models actually learn [152]. In doing so, we now better recognise the strengths, and more importantly the weaknesses of DNNs; subsequently we have developed better networks by building on from previous architectures [29].

The contents of this chapter are organised as follows: in the following section we provide an overview and motivations for the study, in Section 3.2 we provide relevant literature on the topic, Section 3.3 offers a breakdown of adversarial attack and defense formulation methods, in Section 3.4 we propose the adversarial targeting algorithm, in Section 3.5 we put forward the results and discussions of the findings, and conclusively, in Section 3.6 we present a summary of the work.

3.1.1 Adversarial Vulnerabilities

A particular method of analysing the robustness of a DNN is through the lens of an adversarial attack. Such methods introduce a small, carefully crafted distortion to the network input in an attempt to deceive the network into misclassifying the input with a high level of confidence [51, 19]. The small distortions to the input, termed adversarial perturbations, are hardly perceptible to humans even when the perturbation is amplified by several orders of magnitude [19]. The ability of adversarial attacks to fool DNNs with hardly perceptible changes in the input highlights an intrinsic difference between artificial intelligence and true intelligence, as human perception cannot easily be fooled by small perturbation of increasing magnitudes. Thus, adversarial attacks are a logical entry point to evaluate DNNs and the vulnerabilities they may contain.

Adversarial Formulation

There are many ways in which an adversarial perturbation can be crafted, utilising various tools and assumptions on the target network and intended input. Existing adversarial attacks, and methods for designing such distortions, can be broadly categorised into white-box and black-box attacks. The distinction between the two different types of attacks being the information that the adversary, the attack formulation method, has on the network and its

parameters. With the white-box attacks, the adversary is assumed to have complete access to the target network in question, including network parameters and architecture [153]. Conversely, the black-box attack is a type of perturbation method used by an adversary with limited or no information on the interested network, including knowledge on parameters or architecture [30].

In this chapter, we focus our efforts at evaluating the robustness [154] of ResNet-18, ResNet-50, and ResNet-101 networks against a simple yet effective white-box adversarial attack, the fast gradient sign method (FGSM) attack [51]. We apply the FGSM perturbations on the MNIST and CIFAR10 datasets for the mentioned networks and present a correlative relationship between the distribution of neurons with high influence and targeting by the adversarial attack. Furthermore, we evaluate methods in minimising the effects of such distortions through dealing with neuron features differently dependent on the influence of the neuron on the network performance.

Adversarial Defenses

With the numerous adversarial attack methods proposed against DNNs, there have been as many defense techniques also proposed in literature [30]. As discussed in Chapter 2, defense models struggle to remain unbeaten by newer, stronger adversarial attacks in an ever-growing area of adversarial research [30, 31]. Adversarial defenses, much like adversarial attacks, can be divided into different sub-categories: (i) defenses focusing on gradient masking/obfuscation, whereby the network weight gradients used by adversaries to form attacks are disguised; (ii) robust optimisation [28], where the network structure/parameters are altered to increase adversarial defenses; and (iii) adversarial example detection, where the goal is to detect an adversarial input and process this entity differently to ordinary inputs [29].

3.1.2 Targeted DNN Neurons

The goal of all adversaries is to deceive the network into predicting, classifying, or recognising an input as a different class to its true self. When the adversary has knowledge of the information learned by the network, as is the case for white-box attacks, it utilises the information to craft a perturbation that exploits weaknesses within the learned network representations of the data [30]. In this chapter, we propose viewing an adversarial attack as an exploitative method that targets specific neurons within a given layer. We refer to targeted DNN neurons as neurons whose outputs are most affected by an adversarial attack. We also

draw a relationship between the adversaries' targeted neurons and neurons that show to have higher influence on the unperturbed network performance.

We assume that, for a given layer, information about the input learned by the layer through back propagation is distributed unevenly amongst individual neurons. We propose using *nodal dropout* to identify influential and uninfluential neurons within an evaluated layer of a network [155]. In doing so, we find *fragile neurons* that carry more information about the input [156] and consequently affect the network performance more than other neurons. Additionally, we identify *non-fragile neurons* that, once removed, do not significantly affect the overall network performance. We consider such neurons to carry less information about the dataset. We examine how the FGSM attack affects different DNN models (ResNet-18, ResNet-50, and ResNet-101) at different stages (epochs) of learning, whilst also comparing how deeper network architectures affect the effectiveness of the formed attack. Therefore, we propose to reinforce the features within the identified non-fragile neurons by amplifying the neuron weights and leaving fragile neurons unchanged.

The FGSM attack utilises learned representations of a network, in the form of the network layer weights, to calculate an effective adversarial example. We aim to identify the fragile and non-fragile neurons within the first convolutional of the network, and post-process them separately during network training to investigate how they affect the overall robustness of the network against an adversarial attack.

3.2 Related Works

Robustness analysis of DNN against adversarial attacks aims to evaluate the ability of networks to resist malicious distortion applied to network inputs [26, 51, 31]. There are different types of attacks available for a potential adversary, each with their own strengths and limitations. Szegdey et al. [19] initially proposed adversarial examples for DNNs using the *Limited-memory Broyden-Fletcher-Goldfarb-Shanno* (L-BFGS) algorithm, an expensive linear search method to form adversarial examples. Thereafter, the FGSM attack proposed by Goodfellow et al. [51] had become one of the early benchmarks for adversarial attacks due to its computational process being less resource intensive when compared to other attacks. The FGSM attack performs a pixel-wise one step update along the gradient sign direction of increasing loss, thus ensuring the perturbation is uniform with respect to increasing network loss. Moosavi-Dezfooli et al. [157] developed another popular attack formulation method, the *Deepfool* method, which performs an iterative attack using linear approximations to identify the minimum distance between an unperturbed and perturbed input that is required

for the model to misclassify. A targeted adversarial attack method was proposed by Carlini and Wagner [158], the *C&W* attack, was designed to make the model misclassify to a specific class, rather than the closest decision boundary class. There are several other attack methods available in the literature. In this study however, we focus specifically on the FGSM adversarial attack due to its one-step gradient calculation and effective performance against state-of-the-art DNN models. Furthermore, we propose a method of highlighting network weaknesses when subjected to malicious input distortions and hence, the formulation method of the distortion is not the focal point of the study.

3.2.1 Adversarial Defenses

In terms of defenses against adversarial attacks, there are an equal number of approaches proposed in literature [31]. One method of defending a DNN against adversarial attacks is through masking network parameters, therefore making it more difficult for an adversary to exploit the information learned by a network to generate adversarial examples. This method, however, has shown to be ineffective against many types of attacks and there exist techniques to circumvent such defensive measures [29]. The works of Grosse et al. [62] show that adversarial examples are drawn from a different distribution to the regular dataset, and thus, can be identified using particular techniques. Methods based upon utilising the difference in input distributions to defend against adversarial examples are concerned primarily with the identification of adversarial examples form the dataset, such that they can be processed separately to regular inputs [29]. These methods are also subject to exploitation by techniques that bypass the adversarial examples detection, making such defense methods weaker to certain types of attacks [21].

3.2.2 Adversarial Learning

One of the most effective methods of defending against adversarial attacks is through adversarial training. In adversarial training, the network is simply trained on as many different adversarial examples as possible in an attempt to learn the different types of perturbations an adversary can utilise. This method, although proven to be highly effective [34], is limited to the variation of adversarial examples the network is trained upon and newly developed adversaries may be still able to deceive the network [159]. Another promising area of study is certifiable robustness against adversarial attacks [26, 160, 31]. Certifiable robustness aims to impose theoretical upper and lower bounds for an evaluated DNN using techniques such as Lipschitz constant estimation, interval bound propagation, and convex adversarial

polytopes [161, 57]. There exist other methods to ensure that a network learns a dataset more efficiently, and in doing so, reduces the effects a potential adversarial attack has on a network [162]. These methods can be classed within the broad category of regularisation techniques that aim to make small modifications to the network learning algorithm, such that the network is able to generalise with greater efficiency, even under an adversarial attack.

3.2.3 Targeting and Influence

In this section we try to find a relationship between highly influential neurons, when considering network performance on a regular dataset, and the likelihood of the identified neurons being targeted by an adversary. Upon identification of the fragile neurons, we propose a method of regularising the specific neurons during a post-training process. As we, the observer, propagate through the network we notice that the deconstructed, abstract characteristics of the data begin to take a shape of salient features, which are then assigned semantic meaning in the form of target labels [163]. Literature on leveraging the information content of features learned by DNNs have been used for various applications. We direct the reader to the works of Golatkar et al. [164] who suggest a method of selective forgetting, in which the authors investigate the erasing of information about a particular subset of a dataset from the trained weights of a DNN. We take inspiration from this framework and put forward the following proposition; that adversarial robustness is hinged on the distribution of *influential* and *uninfluential* neurons, referred to as sets of fragile and non-fragile neurons respectively within the context of this study.

We are motivated by the works of Li and Chen [155] along with related literature in reducing network complexity by using techniques such as nodal pruning. We leverage the idea that there exist neurons within a network that can be classified as redundant, or uninfluential, to the overall network performance. Removing redundant neurons in some cases also shows to improve robustness against attacks [165]. Conversely, we also consider the works of [156] that prove the existence of multi-model neurons within networks; representations that hold a higher degree of influence to the network performance. We investigate the correlation between neurons that show to have a higher influence towards the network performance and the average concentration of an adversarial targeting on influential neurons. In consequence, we draw attention to the nature of adversarial attacks and how such perturbations target the learned representations of a network that are most important to performance on the clean dataset.

3.3 Adversarial Attack and Defense Formulations

We consider an image classifier f with l layers, and trainable parameters θ that accepts an input image x and its associated true class label y. The network returns $\hat{y}$ as its prediction for input x. The goal of the model is to reduce loss function $\mathscr{L}(\hat{y}, y)$. The image $x_\varepsilon = x + \delta_\varepsilon$, is an adversarial example produced by an adversarial attack δ_ε applied to image x, where ε is the magnitude of the perturbation. The network prediction for an adversarial example x_ε is defined as $\hat{y}_\varepsilon$.

Our objective is to minimise the difference in network predictions $\hat{y}$ obtained for unperturbed input x, and adversarial prediction $\hat{y}_\varepsilon$ for perturbed input x_ε. We examine the trainable parameters $\theta^{(l)} \in \theta$ of a network for layer l at various stages of the training process. It should be noted that while assessing the significance of the neurons, we remove one-neuron at a time from $\theta^{(l)}$. We, therefore, identify two sets of neurons indices, S and $\bar{S}$ respectively representing (i) neuron indices within the layer L showing a higher influence on the overall model performance, and (ii) neurons indices with lower overall influence on model performance. In our work, we are concerned with removing one-neuron at a time; removing multiple neurons from the model $f(x, \theta)$ would warrant an alternative method that considers neuron sampling methods. We focus on removing single neurons to evaluate network performance, and specifically, assessed neurons of the first layer of networks. We investigate the first layer only, due to the high importance and influence of features learned by the first layer, with respect to the whole network [165].

3.3.1 Attack Formulation

The adversarial attack used in this work if formulated using the FGSM attack. This method leverages a learned representations of a network, in the form of layer weights $\theta^{(l)}$, to construct an efficient and effective adversarial perturbation x_ε. The FGSM attack is a perturbation for an input x computed as:

$$\delta_\varepsilon = \varepsilon \; sign(\nabla_x \mathscr{L}(x, y, \theta)), \tag{3.1}$$

where ∇_x are the gradients calculated using backpropagation. The adversarial example therefore is $x_\varepsilon = x + \delta_\varepsilon$, as detailed in [26, 51].

We find that for a 100 epoch pre-trained ResNet-50 model on the CIFAR10 dataset, a baseline model accuracy of 75.87% on the unperturbed inputs x is observed. The same model applied to the CIFAR10 dataset with an FGSM attack, using a perturbation magnitude of $\varepsilon = 0.01$, results in an accuracy of 58.88%. If we consider the same ResNet-50 architecture

trained equally for 100 epochs, with the input dimensions adjusted to comply with the MNIST dataset, the baseline model accuracy on unperturbed MNIST dataset is 99.42%. While the model accuracy is found to be 79.4% when perturbed with an $\varepsilon = 0.34$ attack. These are examples of the FGSM attack performance on the CIFAR10 and MNIST datasets using the ResNet-50 DNN model.

If we consider a metric to assess the complexity of a given dataset, such as the cumulative spectral gradient (CSG) method [166], we notice that the CSG complexity measure for the for the CIFAR10 dataset is 1.00 and MNIST dataset is 0.11. As we may expect, the FGSM attack is more effective on more complex data (e.g., CIFAR10) compared to less complex data (e.g., MNIST). This can be realised from the perturbation magnitude ε required for the model performance to decrease proportionally. For example, to decrease performance by approximately 20%, a lower value of ε (smaller perturbation magnitude) is required for CIFAR10 and a higher value of ε (larger perturbation magnitude) is required for the MNIST dataset.

3.3.2 Defense Formulation

To better understand how to form a suitable defense against an adversarial attack, we may consider how an adversary is able to form an effective attack. With the FGSM attack, a single step in the input space is taken in the direction of increasing loss. The perturbation is calculated using the network weights gradients, and subsequently the input data features are distorted in the direction of increasing loss with respect to the gradient updates. Then it is natural to consider that this informed way of creating adversarial perturbations may, even with relatively low magnitudes, affect the neurons that are more influential to the model performance (e.g., set of highly influencing neurons S).

We aim to show this effect of adversarial perturbations experimentally by comparing the output of the layer-wise convolution of the clean inputs x, and perturbed inputs x_ε that are computed using the pre-trained parameters θ. We define the model prediction $f(x,\theta)$ on clean inputs and the model prediction $f(x_\varepsilon, y, \theta)$ on perturbed input to not be equal ($f(x,\theta) \neq f(x_\varepsilon,\theta)$), such that the adversarial attack affects the network performance. In our defense formulation, we aim to modify the network layer parameters $\theta^{(l)}$ to $\tilde{\theta}^{(l)}$, such that a potential adversary is forced to distribute the attack targeting throughout the layer. We propose that this will make the network layer $\tilde{\theta}^{(l)}$ more robust against an adversarial attack.

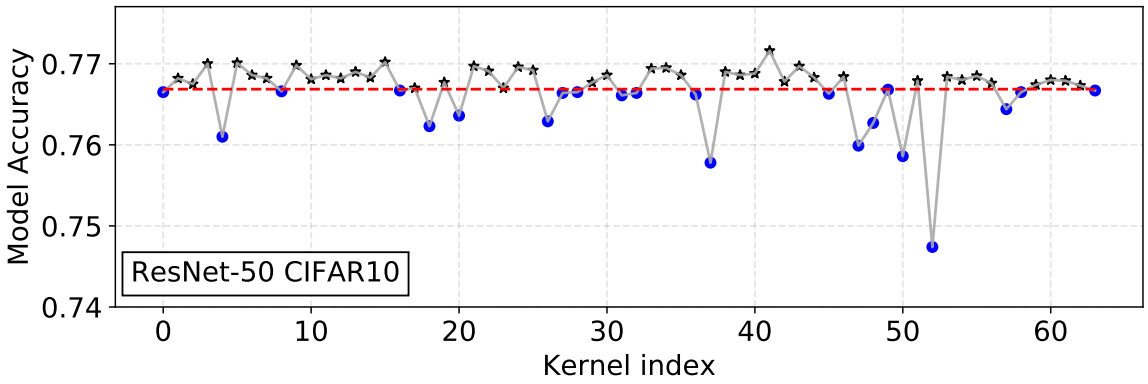


Fig. 3.1 Evaluated ResNet-50 model trained for 10 epochs. Fragile kernels S shown in blue below mean performance line in red and non-fragile kernels $\bar{S}$ are shown in black star above mean line in red.

3.3.3 Fragile and Non-fragile Kernels Identification

We identify fragile neurons (filter kernels) S and non-fragile neurons $\bar{S}$ through removing the neurons out systematically one-by-one, and measuring the variance in model performance. In Figure 3.1, we show the network performance for each removed neuron from the first convolutional layer of the network, represented by the indices along the x-axis. The indices of fragile kernels S are indicated with blue circled symbols and are observed to be less than the mean performance of the network. The mean network performance to neuron removal is indicated using a red dotted line and is computed over all neurons of the first convolutional layer. The filtering of the fragile neurons has a higher influence on the network performance when compared to the dropping of the non-fragile neurons, indicted by the black star symbol and observed to be above mean performance line.

3.4 Adversarial Targeting Algorithm

We assume that the sub-set of parameters $\theta_{\bar{S}}^{(l)}$ relating to non-fragile kernels $\bar{S}$ in layer l, carry within them artefacts of noise that render the overall influence of these kernels, with respect to the model performance, to be lower than that of the fragile kernel S. We propose to filter parameters $\theta_{\bar{S}}^{(l)}$ in order to remove this noise. We group the sub-set of identified fragile neurons of the layer parameters $\theta^{(l)}$ as a matrix of neurons K_S, and equally, the subset of layer parameters identified as non-fragile, as a matrix $K_{\bar{S}}$. We assume the distribution of the noise in the matrix $K_{\bar{S}}$ to be of a Gaussian distribution. Using this approach, we can use the works of Gavish and Donoho [167] to recover a lower rank matrix from $K_{\bar{S}}$ to retaining only

the most important features and remove noise from the neurons. The filtering of parameters $K_{\bar{S}}$ results in the modified parameters $\tilde{\theta}_{\bar{S}}^{(l)}$. The filtered parameters relating to non-fragile kernels $K_{\bar{S}}$, are considered to be more robust, if the probability of predicting the true class using modified model parameters $\tilde{\theta}$ is higher than θ as per:

$$P(\hat{y} = y|x_{\varepsilon}, \theta') > P(\hat{y} = y|x_{\varepsilon}, \theta). \tag{3.2}$$

We compose the matrix $K_{\bar{S}}$ by stacking flattened non-fragile kernel parameters $\tilde{\theta}_{\bar{S}}^{(l)}$ and propose compressing the features of $K_{\bar{S}}$ to remove noise or redundant information, thus increasing the influence of the non-fragile kernels $\bar{S}$ on the overall network performance. While filtering the non-fragile kernels $\bar{S}$, we proportionally amplify fragile kernel S. The amplification of S is carried out to maintain relative magnitude of local features of weights in layer l and to propagate only the essential representations to subsequent layers of the network.

3.4.1 Filtering Non-fragile Kernels

We decompose the non-fragile kernel matrix $K_{\bar{S}}$ using singular value decomposition (SVD) and reduce the complexity of the representations by clamping all singular values below a hard filtering threshold τ. The value of τ used, is computed using the method presented in Gavish and Donoho [167] that outlines the optimal hard threshold for singular values. We apply the proposed filtering method only to the first convolutional layer, due of the susceptibility to distortions having a higher influence on the overall network performance [165]. We use SVD to decompose our non-fragile kernel matrix $K_{\bar{S}}$, which is an $m \times n$ matrix, into its respective eigenvalues Σ along with the the eigenvectors matrices U and V as per:

$$K_{L,\bar{S}} = U\Sigma V^T. \tag{3.3}$$

We thereafter, compute a truncated matrix of singular values $\widetilde{\Sigma}$ by clamping all values to be at most equal to threshold value τ as per:

$$\widetilde{\sigma}_i = \arg\min(\sigma, \tau), \tag{3.4}$$

where σ is the diagonal of Σ and $\widetilde{\sigma}_i$ is the row up to which the matrix σ is truncated. The thresholding value τ, for $m \times n$ matrix is given as per:

$$\tau = \lambda(\rho) \cdot \sqrt{n}\varepsilon, \tag{3.5}$$

where $\rho = m/n$, ε is the noise level within the matrix, and the term $\lambda(\rho)$ is expressed as:

$$\lambda(\rho) = \sqrt{2(\rho+1) + \frac{c_1\rho}{(\rho+1) + \sqrt{\rho^2 + c_2\rho + 1}}}, \tag{3.6}$$

where constants c_1 and c_2 respectively are 8 and 14 as per [167].

We then find the noise level value ε in (3.5) experimentally through a systematic search method using a sample set of the parameters. As the final filtering step, we reconstruct the filtered weight matrix $\widetilde{K}_{\bar{S}}$ by using the clamped singular values and corresponding eigenvectors as per:

$$\widetilde{K}_{\bar{S}} = U\widetilde{\Sigma}V^T. \tag{3.7}$$

3.4.2 Amplification of Fragile Kernels S

The amplification of the fragile kernel parameter matrix K_S is carried out applying a scaling factor of η. The value of η is computed using (3.3) and (3.7), as per:

$$\widetilde{K}_S = \eta K_S, \tag{3.8}$$

where scaling factor of η is

$$\eta = 1 + ||K_S - \widetilde{K}_{\bar{S}}||_2. \tag{3.9}$$

The aim of this process is to amplify the features within fragile kernels S, such that a greater magnitude of adversarial perturbation is required to affect kernels. The magnitude of amplification being equal to the reduction in magnitude of kernels $\bar{S}$, such that the overall magnitude of the layer remains the same as before applying the framework.

3.4.3 Back-propagation Filters

Another method from that is proposed in Sections 3.4.1 and 3.4.2 is based on *back-propagation filters*; a method in which the back-propagation weight updates during network training are processed differently for fragile and non-fragile kernels. The premise of the regularisation technique can be succinctly described as a *learning filter* that governs the weight update of kernels dependent on the importance of the kernel through nodal dropouts. It should be noted that the mentioned back-propagation filters are not filters for the specific kernels, rather the method is a masked filter for the gradient updates during network training.

3.4.4 Adversarial Targeting of Fragile and Non-fragile Kernels

We assess the robustness of the fragile kernels S and non-fragile kernels $\bar{S}$ using our adversarial targeting algorithm, as shown in Algorithm 1. The FSGM attack is varied for a range of perturbations ε, which in turn, is used to compute the outputs of the evaluated first convolutional layer for both clean prediction $\hat{y}_x$ and the adversarial prediction $\hat{y}_\varepsilon$. The average difference between each kernel in the outputs $\hat{y}_x$ and $\hat{y}_\varepsilon$ is calculated and compared to see which is highest, indicating a greater average concentration of the attack.

Algorithm 1 Adversarial targeting

Initialise $f() \rightarrow f_L()$ ▷ $f_L()$ is the L-th layer of full network $f()$
Compute indices of fragile kernels S and non-fragile kernels $\bar{S}$ as per Sec 3.3.3
$S_{attack} = \{\}$ ▷ an empty list to store examples that attacks S
for perturbation $\varepsilon \in \mathbb{R}$ **do** ▷ where ε is perturbation magnitude
    $attack$ = FGSM($f^{(L)}, \varepsilon$)
    $S_{\text{count}} = 0$
    for (x, y) in (X_{test}, Y_{test}) **do**
        $x_\varepsilon = attack(x, y)$ ▷ create an adversarial example x_ε for input x and level y
        $\hat{y}_x = f^{(l)}(x)$ ▷ output of l-th layer on unperturbed input x
        $\hat{y}_\varepsilon = f^{(L)}(x_\varepsilon)$ ▷ output of l-th layer on perturbed input x_ε
        $\mathbf{d} = ||\hat{y}_x - \hat{y}_\varepsilon||_2$ ▷ Euclidean distance $\mathbf{d} = (d_1, \ldots, d_k)$ between $\hat{y}_x$ and $\hat{y}_\varepsilon$
        $S_f = (\sum_j^{|S|} d_{j,S})/|S|$ ▷ Average of distances $d_{j,S}$ of all S select from $\mathbf{d}$
        $S_n = (\sum_j^{|\bar{S}|} d_{j,\bar{S}})/|\bar{S}|$ ▷ Average of distances $d_{j,\bar{S}}$ of all $\bar{S}$ select from $\mathbf{d}$
        if $S_f > S_n$ **then**
            $S_{\text{count}} = S_{\text{count}} + 1$ ▷ increase counter of attacks for fragile kernels
        end if
    end for
    $S_{attack} \leftarrow S_{\text{count}}$ ▷ add S_{count} to the list S_{attack}
end for

3.5 Results and Discussion

In first series of experiments, we use the two sets S and $\bar{S}$ obtained as per Figure 3.1 on the ResNet-50 model and apply them to Algorithm 1 using the CIFAR10 dataset, resulting in Figure 3.2 and the MNIST dataset, resulting in Figure 3.3:

For Figure 3.2, we measure the robustness of ResNet-50 models and compare the percentage of examples attacking fragile kernels S when evaluated against the FGSM attack. In Figure 3.2 (*Left*), we observe that as the number of training epochs increases, the network

accuracy also increases for both the unperturbed ($\varepsilon = 0$) and perturbed ($\varepsilon > 0$) examples. In Figure 3.2 (*Right*), using the results from the adversarial targeting Algorithm 1, we also notice that the percentage of examples attacking fragile kernels S is higher for highly perturbed examples. However, for smaller perturbation magnitudes, 100 epoch model is more robust. This suggests that as the model becomes more robust (from epoch 10 to 100), the percentage of examples attacking fragile kernels S and non-fragile kernels $\bar{S}$ tends to distribute equally.

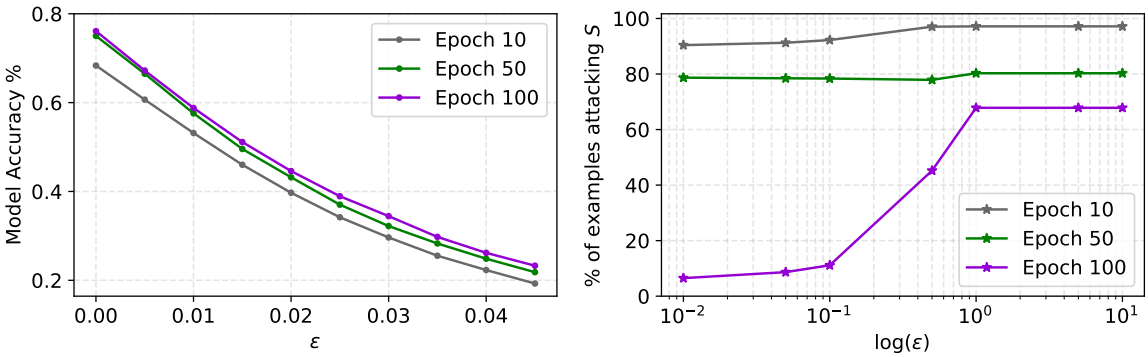


Fig. 3.2 *Left:* ResNet-50 model trained on the CIFAR10 dataset for epochs 10, 50 and 100 against the FGSM attack, with ε increasing linearly, marked by dots. *Right:* ResNet-50 model trained on the CIFAR10 dataset for epochs 10, 50 and 100 against the FGSM attack, with attack magnitude increasing logarithmically, marked by star symbols. Epoch 10, 50, 100 respectively indicated in colors grey, green, violet.

After applying our framework proposed in sections 3.4.1 and 3.4.2 using ε value of 0.015 to the first convolutional layer $\theta^{(l)}$, resulting in filtered layer parameters $\theta^{(l)}$, we observe the difference in attack distribution between original model and modified model using Algorithm 1.

We apply the parameter filtering framework to a ResNet-50 model trained on the MNIST for 10 epochs. The results of which is shown in Figure 3.3. In this experiment, although the number of fragile kernels S are 37% of the total kernels within the layer, these kernels show a larger average distance between the outputs of the original layer $\theta^{(l)}$ and modified layer $\theta^{(l)}$ for almost 89% of the tested input examples on original model. Furthermore, as the attack strength is increased, through a logarithmic increase of ε, the average magnitude of the attack on kernels S also increased. However, our method of filtering parameters $\theta^{(l)}$ resulted in a lower percentage of test examples attacking fragile kernels S, compared to the original model with parameters θ.

We observe from Figure 3.4 how the influence of kernels in the first convolutional layer varies during the training process while we systematically drop and assess the kernels. In Figure 3.4, red circles are the kernels that carry a higher influence through all stages of

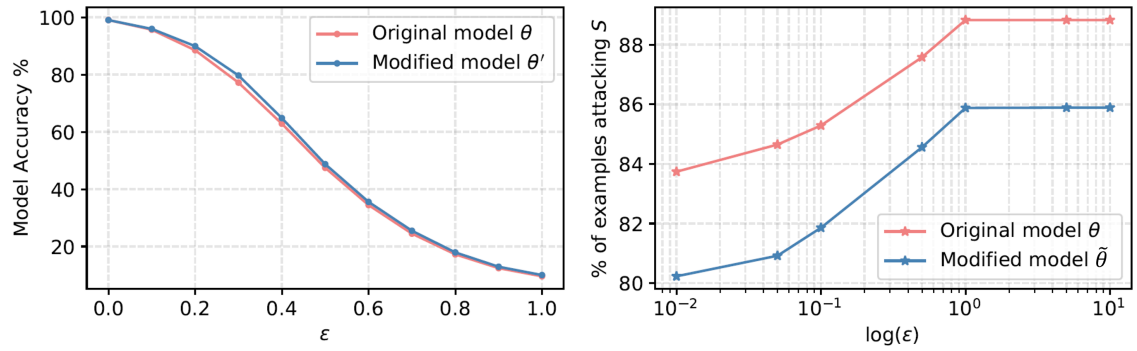


Fig. 3.3 Concentration of the adversarial attack on fragile kernels S for both the original model with parameters $\theta^{(l)}$ and the modified model, with $\theta^{(l)}$ in a ResNet-50 model trained on the MNIST dataset for 10 epochs, using the methods proposed in Sections. 3.4.1 and 3.4.2.

model training. We notice that as we change the model from ResNet-18 to ResNet-50 and ResNet-101, the number of influential fragile kernels increases on the CIFAR10 dataset. This is as we may expect; model architectures with greater complexities are able to learn the important features from the dataset faster than shallower model architectures. We notice from Figure 3.4, that the average model performance of the kernels in $\theta^{(l)}$ increases to a limit for models trained on the CIFAR10 dataset and shows to increase and then decrease for the models trained on the MNIST dataset. This characteristic invites a separate set of experiments to better understand how model overfitting affects nodal dropouts.

3.6 Summary

In this chapter we showed how an FGSM attack targets specific neurons within the first convolutional layer of ResNet-18, ResNet-50, and ResNet-101 models trained on both the CIFAR10 and NNIST datasets. To prove this property, we first identify fragile kernels S and non-fragile kernels $\bar{S}$ sets within the evaluated layer using an iterative dropout method and measuring the variance in model performance. We use the kernel indices of S and $\bar{S}$ to evaluate the highest average distance between the outputs of the layer using the original input x and perturbed example x_ε. In doing so, we find that for a ResNet-50 model trained on the CIFAR10 dataset for 50 epochs, the number of fragile kernels S account to 37% of the total number of kernels in the layer yet show to have a higher average difference for approximately 89% of the examples evaluated.

We also show how the robustness against the FGSM attack, and the targeting of fragile kernels S, varies as the model is trained, thus showing a correlation between a model becoming more robust and the targeting of fragile kernels. Furthermore, we propose a layer

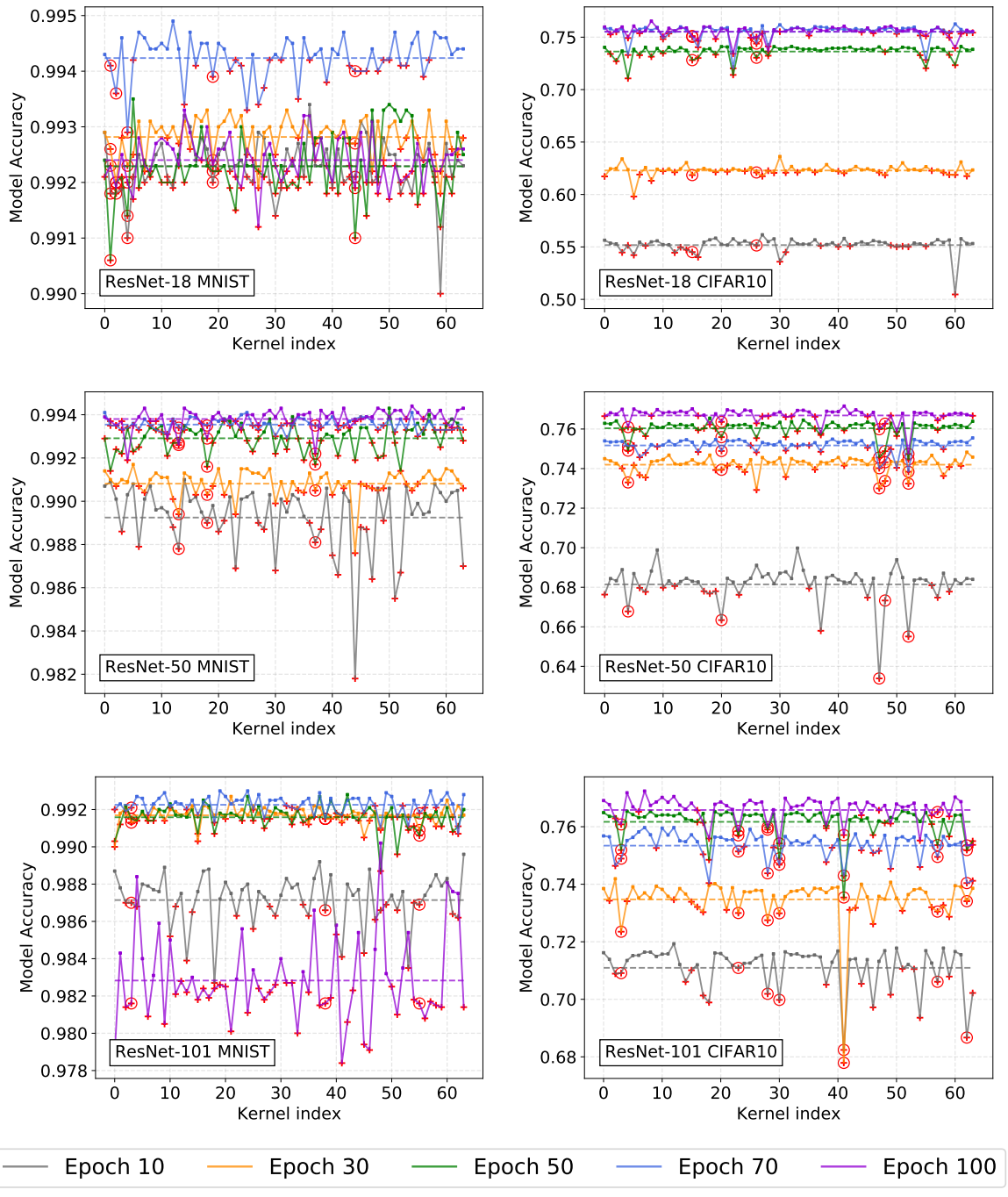


Fig. 3.4 Variance of model performance to individual kernels being dropped out within the first convolutional layer. Red circles indicate fragile kernels that remain fragile throughout the all training epochs, whereas red crosses indicate kernels that are observed as fragile for the specific training epoch length.

parameter filtering algorithm that improves robustness in a model by removing information

from non-fragile kernels $\bar{S}$ and amplifying the information in S. This simple method, despite only being applied to the first convolutional layer, improves the robustness of a model with less training. It should be noted that, although our study focuses on the first convolutional layer only, due to the layer being highly influencing on the network performance, other layers can also be evaluated using this proposed framework.

Chapter 4

Fragility, Robustness and Antifragility for DNNs

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.

4.1 Overview

Deep neural networks (DNNs) are extensively used in various tasks and domains, achieving noteworthy performances in both research and real-world applications [1, 3]. 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 [152, 168]. 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 [35, 104, 169] 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 [95]. Through analyzing the performance of a network to input perturbations (external stress) formed using an adversarial attack [19, 20], 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* [38] 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 [19, 20] due to their ability to exploit the learned representations of a network to decrease network performance [18]. In our study, we focus on the fast gradient sign method (FGSM) attack for its equal single-step perturbation calculation for increasing network loss [51]. Our synaptic filtering methodology (see Fig. 4.1) 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 *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 [53], we take a novel methodology that considers network susceptibility to adversarial perturbations in conjunction with network architecture and the learning processes (see Fig. 4.1). 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. 4.1 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 [170], SqueezeNet-v1.1 [171], and ShuffleNet V2 x1.0 [172] networks for the MNIST [151], CIFAR10 [5] and the Tiny ImageNet datasets [173].

The main contributions of this Chapter, therefore, are as follows:

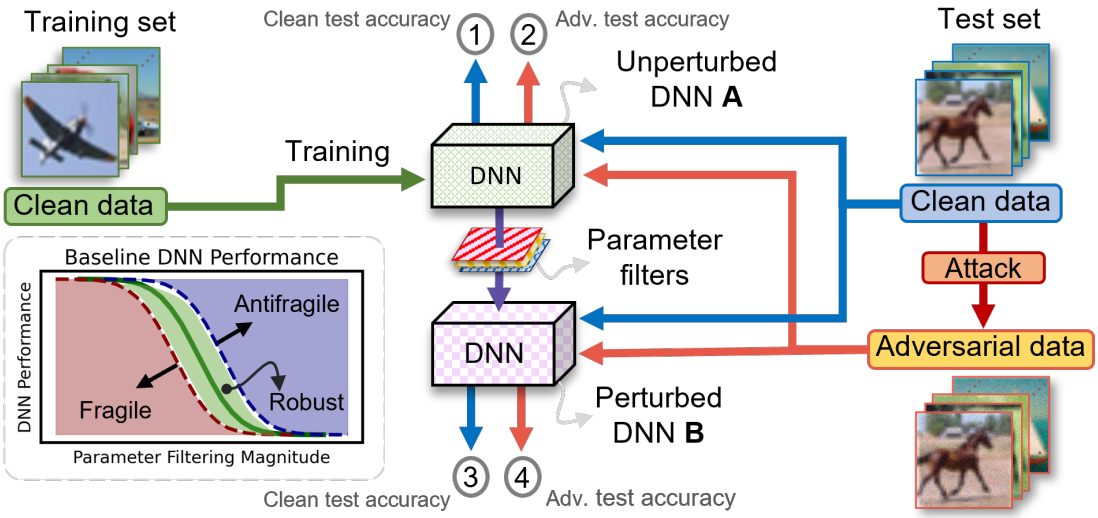


Fig. 4.1 Our methodology of parameter filtering and evaluating DNN performances on clean and adversarial datasets. Passing a DNN through parameter filters is equivalent to internal stress and applying an adversarial attack with various magnitudes on clean data is equivalent to external stress on a DNN. In this methodology, the DNN performances (labeled 1, 2, 3, and 4) are individually compared against a defined baseline DNN performance (sold green line 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).

- 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 param-

eters are retrained at any given epoch, thus proving a useful strategy for improving network robustness.

The following Section 4.2 gives insights into the background and related works. Section 4.3 offers definitions of the terms and concepts introduced in the proposed methodology. Section 4.4 reports the proposed methodologies. Section 4.5 shows the experimental results acquired using the proposed methodologies, and Section 4.6 concludes the work.

4.2 Related Work

We propose evaluating the resilience of DNNs using a physiologically inspired approach concerning the resilience of humans to stress on their physiology [95, 174]. 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) [96, 97] and we take external stress to be variations to the learning environment of the network (i.e., input perturbations) [175, 176, 54].

There exist various avenues of research when considering an analysis of DNNs to input perturbations [51, 139] and synaptic filtering [97, 104]. The works of Szegedy et al. [19] and Goodfellow et al. [51] 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 [26] and equally abundant adversarial defense techniques [125, 31], 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. [102] 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 [169, 97, 104] and adversarial attacks [53, 26, 31]. Our notion of antifragility in DNN is in line with the antifragility notion described by Taleb and Douady [38]) 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 [177–180]. Therefore, the works of Ilyas et al. [126] highlight the presence of robust and non-robust features within networks. In a similar context, we highlight the presence of fragile, robust and antifragile [38] 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. 4.1).

Identifying fragile, robust and antifragile parameters informs us about the *compressibility* of a network based on the variation and degradation in the network performance [47]. A central principle of network compression techniques is to reduce network size whilst retaining network performance [102]. A method of achieving network compression is through using network pruning techniques [101, 97, 104]. 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 [181], i.e., we systematically internal architecture of the DNN.

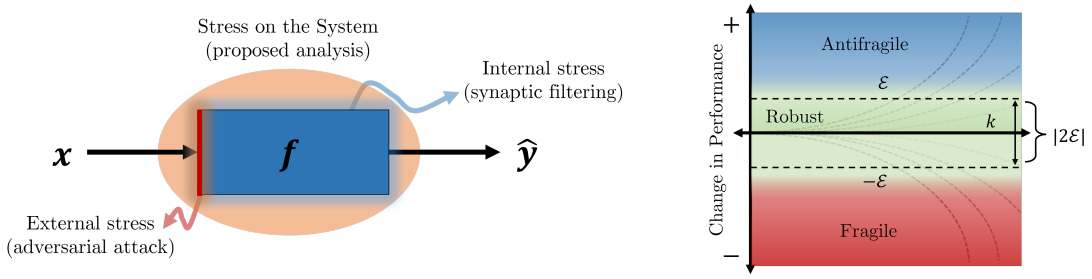
Siraj et al. [117] proposed a robust sparse regularisation method for network compactness while simultaneously optimizing network robustness to adversarial attacks. Similarly, we use our synaptic filtering methodology (a network parameter removal technique) to study the performances of a DNN on clean and adversarial datasets, which enable us to identify parameters that cause a decrease in network performance on the adversarial dataset [112] 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 [176, 54]. The primary objective of adversarial attacks is to deceive a network into misclassifying otherwise correctly classified inputs [19, 51]. The importance of the analysis of adversarial attacks on DNNs is significant due to the existence of adversarial examples in real world applications [182, 23]. 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 [112] it is possible to

observe an increase in network performance, even under an adversarial attack, thus requiring the notion of antifragility.

4.3 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. 4.2, where Fig. 4.2a shows the application of stress on a DNN and Fig. 4.2b shows the interpretation of DNN performance for parameter characterization. The ε bounds around robustness indicate a variable For detailed definitions of the above-mentioned concepts, we consider the following notations.



(a) Evaluation overview

(b) Response characteristics

Fig. 4.2 (a) Showing an overview of the proposed system evaluation method. (b) shows the characteristics of fragility, robustness and antifragility through analyzing the performance of a system 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. [101], 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 $\hat{y} = \arg\max_{1\leq k\leq K} f_k(x,\mathbf{W})$. The network parameters $\mathbf{W}$ are assumed to be optimized, either partially or fully, using back-propagation and a loss function $\mathscr{L} : \mathbb{R}\times\mathbb{R} \to \mathbb{R}$ given by $\mathscr{L}(\hat{y},y)$ to calculate the network error.

4.3.1 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 [102] and $\theta = \mathbf{W}$ is the global parameter analysis, as mentioned in [183].

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:

$$\tilde{\theta}_\alpha = B_\alpha \odot \theta_\alpha, \quad B_\alpha \in \{0,1\}^{|\theta_\alpha|}, \tag{4.1}$$

where $\alpha = \{\alpha_0, \alpha_1, \ldots, \alpha_A\}$ is the synaptic filtering threshold with a lower bound α_0, upper bound α_A and step size $\alpha_1 = \alpha_0 + \Delta_\alpha$. 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. 4.1, 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 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 $\mathscr{S}_\alpha$:

$$\mathscr{S}_\alpha = \{f(x,\tilde{\theta}_{\alpha_0}), f(x,\tilde{\theta}_{\alpha_1}), \ldots, f(x,\tilde{\theta}_{\alpha_A})\}. \tag{4.2}$$

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 $x_\varepsilon = x + \delta_\varepsilon$ as the *perturbed example* of x with an adversarial perturbation $\delta_\varepsilon \in \mathbb{R}^d$, where $\varepsilon = \{\varepsilon_0, \varepsilon_1, \ldots, \varepsilon_E\}$ is the perturbation magnitude with minimum perturbation magnitude ε_0 and maximum perturbation magnitude ε_E with step size $\varepsilon_1 = \varepsilon_0 + \Delta_\varepsilon$. Using a single adversarial attack formulation method δ we define $\hat{y} = f(x, \theta)$ as the network predictions on clean dataset and $\hat{y}_\varepsilon = f(x_\varepsilon, \theta)$ as the network predictions on the adversarial dataset [Fig. 4.1(Left)]. When dealing with external stress only, θ is taken as the complete set of network parameters $\mathbf{W}$.

The performance of $f(x_\varepsilon, \theta)$ 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 [176, 54]. 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. 4.1).

When applying external stress with various magnitude ε, we get a set of perturbed inputs for the network $\mathscr{S}_\varepsilon$:

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 $|\varepsilon|$ inputs to the network $\mathscr{S}_\varepsilon$:

$$\mathscr{S}_\varepsilon = [f(x_{\varepsilon_0}, \theta), \ldots, f(x_{\varepsilon_E}, \theta)]. \tag{4.3}$$

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 ($\mathscr{S}_\alpha$ and $\mathscr{S}_\varepsilon$) may offer equal performance to the unperturbed network, i.e., for all inputs x in test set, we observe:

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

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 $\hat{f}(x_\varepsilon, \theta)$ 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 $\hat{f}(x_\varepsilon, \theta_\alpha)$, 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 an example when we apply internal stress, it produces a set of various filtered networks $\mathscr{S}_\alpha$. 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 $\tilde{\theta}_{\alpha_A} = \mathbf{0}$. Hence, the performance of 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 Section 4.4.1.

For a given stress function, we must define a corresponding baseline network performance such that the unperturbed network performance can be compared to the baseline. The baseline network performance is defined as a dependent to the stress magnitude parameters, α for internal stress and ε for external stress. This is so that we can define the baseline network response to have different characteristics for different magnitudes and types of stress.

4.3.2 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 ($\mathscr{S}_\alpha$) or external ($\mathscr{S}_\varepsilon$), 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 $\mathscr{S}_\alpha$ to $\mathscr{S}_\varepsilon$, from $\hat{f}(x,\tilde{\theta}_\alpha)$ to $\hat{f}(x_\varepsilon,\theta)$, and 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:

$$\sum_{i=\alpha_0}^{\alpha_A} [\hat{f}(x,\tilde{\theta}_\alpha) - \mathscr{S}_\alpha]\Delta_\alpha < -\varepsilon, \tag{4.4}$$

where Δ_α is the change in synaptic filtering threshold α, $\varepsilon \geq 0$ and asserts a variable fragility measure, as shown in Fig. 4.2b (red shaded region). When the threshold $\varepsilon = 0$, we have a strict fragility condition. Equation (4.4) 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:

$$-\varepsilon \leq \sum_{i=\alpha_0}^{\alpha_A} [\hat{f}(x,\tilde{\theta}_\alpha) - \mathscr{S}_\alpha]\Delta_\alpha \leq \varepsilon, \tag{4.5}$$

where Δ_α is the change in synaptic filtering threshold α, $\varepsilon \geq 0$ and asserts a variable robustness measure, as shown in Fig. 4.2b (green shaded region). When the threshold $\varepsilon = 0$, we have a strict robustness condition. Equation (4.4) 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:

$$\varepsilon < \sum_{i=\alpha_0}^{\alpha_A} [\hat{f}(x,\tilde{\theta}_\alpha) - \mathscr{S}_\alpha]\Delta_\alpha, \tag{4.6}$$

where Δ_α is the change in synaptic filtering threshold α, $\varepsilon \geq 0$ and asserts a variable robustness measure, as shown in Fig. 4.2b (blue shaded region). When the threshold $\varepsilon = 0$, we have a strict antifragility condition. Equation (4.4) computes the discrete area difference

between the stressed network performance and the baseline network performance for all stress magnitudes of α.

4.4 Methodology of DNN parameters characterization

In this Section, we present the methodology of DNN parameter characterization that is shown in Fig. 4.1. Concisely, Fig. 4.1 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 Section 4.4.1 and then introduces parameter scores that characterize the parameters in Section 4.4.2. Finally, we discuss the experiment setting in Section 4.4.3.

4.4.1 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. 4.3, 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.

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 Section 4.4.1). In this study, we work primarily with the FGSM attack [51] 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. 4.3(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 follow:

Definition 7 (minimum attack bound ε_0 – *constraint 1.*)**.** We limit the adversarial attack to follow $p(\hat{y}_\varepsilon = y|x+\delta_{\varepsilon_0}) < p(\hat{y} = y|x)$, for all inputs x in the test dataset. This constraint

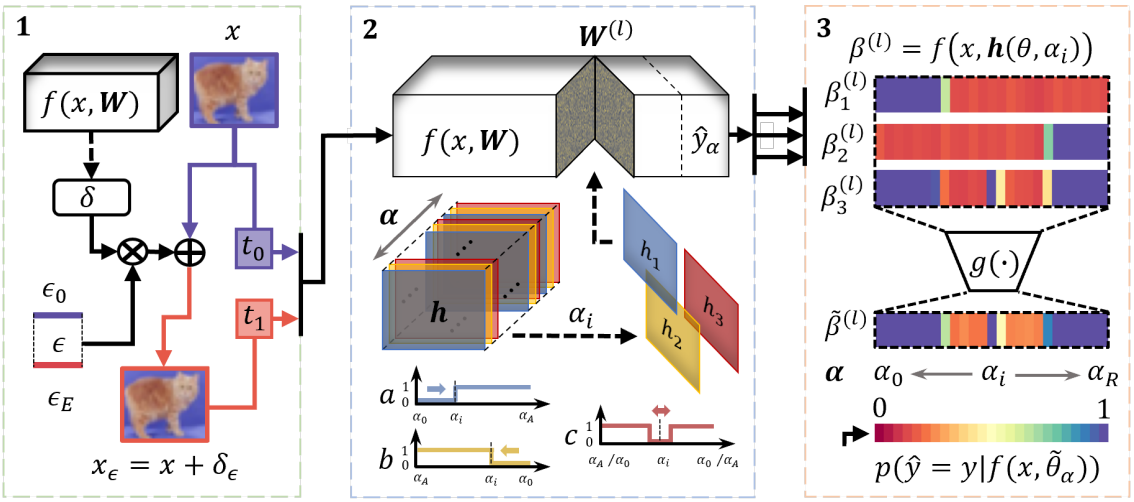


Fig. 4.3 Synaptic filtering framework. *Left* block (**1**) shows the input x at time t_0; network $f(x, \boldsymbol{W})$ with parameters $\boldsymbol{W}$; the adversarial attack δ [this study computes δ using $f(x, \boldsymbol{W})$]; the perturbation magnitude ε and the resultant adversarial example x_ε at time t_1. The perturbation magnitude ε is bounded by $(\hat{y}_\varepsilon \approx \hat{y})$ and $(\hat{y}_\varepsilon K > 1)$ for K classes; $\hat{y}$ and $\hat{y}_\varepsilon$ 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 R sets of $\boldsymbol{h}$ for each $\alpha_i \in [\alpha_0, \alpha_A]$. *Right* block (**3**) shows $\beta^{(l)} = f(x, \boldsymbol{h}(\theta, R))$ 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 $\hat{\beta}^{(l)}$.

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 $(\hat{y}_{\varepsilon_E} K > 1)$, i.e., we have the constraint: $p(\hat{y}_{\varepsilon_E} = y | x + \delta_{\varepsilon_E}) K > 1$, 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 $f_1(x, \theta) - f_1(x + \delta_{\varepsilon_1}, \theta) \approx f_2(x, \theta) - f_2(x + \delta_{\varepsilon_2}, \theta)$ thus ensuring that the adversarial perturbations affect the network performances equally.

Synaptic filtering of DNNs

We investigate a set of synaptic filters $h = \{h_1, h_2, h_3\}$ containing three different synaptic filters [Fig. 4.3(Middle block)]: h_1, the ideal *high-pass* filter; h_2, the ideal *low-pass* filter and h_3 the *pulse wave* filter.

We apply filter h_1 to the learned (unperturbed) network parameters θ, resulting in perturbed network parameters $\tilde{\theta}_{1,\alpha_i}$:

$$\tilde{\theta}_{1,\alpha_i} = h_1(\theta, \alpha_i) = \begin{cases} 0 & \text{if } \theta \leq \alpha_i, \\ 1 & \text{otherwise} \end{cases}, \tag{4.7}$$

for an $\alpha_i \in \alpha$, where $\alpha = \{\alpha_0, \alpha_1, \ldots, \alpha_A\}$ and 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 $\mathscr{S}_\alpha$ with each threshold defined as $\alpha_i = \alpha_{i-1} + \Delta_\alpha$ for a step length $\Delta_\alpha = [\max(\theta) - \min(\theta)]/A$ and $i = \{i \in \mathbb{N} : 0 \leq i \leq |\alpha|\}$.

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 $\alpha_i \in \alpha$:

$$\tilde{\theta}_{2,\alpha_i} = h_2(\theta, \alpha_i) = \begin{cases} 0 & \text{if } \theta \geq \alpha_i, \\ 1 & \text{otherwise} \end{cases}, \tag{4.8}$$

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 is given by:

$$\tilde{\theta}_{3,\alpha_i} = h_3(\theta, \alpha_i) = \begin{cases} 0 & \text{if } \alpha_i - \frac{\Delta_\alpha}{2} < \theta \leq \alpha_i + \frac{\Delta_\alpha}{2}, \\ 1 & \text{otherwise} \end{cases}, \tag{4.9}$$

where $\alpha_i = \alpha_{i-1} \pm \Delta_\alpha$, $\alpha_0 = \min(\pm\theta)$, and $\alpha_A = \max(\pm\theta)$. In Eq. (4.9), 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.

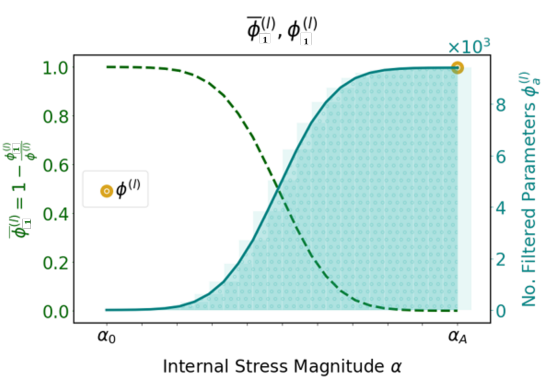
Baseline Network Performances We denote ϕ_1, ϕ_2 and ϕ_3 to be the number of parameters filtered out by the synaptic filters h_1, h_2 and h_3 corresponding to filtering threshold α_i. If the synaptic filtering procedure is only applied to a local layer l (e.g., only on a convolutional layer or a linear 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 $\bar{\phi}_{1,\alpha_i}^{(l)}$ at threshold $\alpha_i$is given as:

$$\overline{\phi}_{1,\alpha_i}^{(l)} = 1 - \frac{\phi_{1,\alpha_i}^{(l)}}{\phi^{(l)}}, \tag{4.10}$$

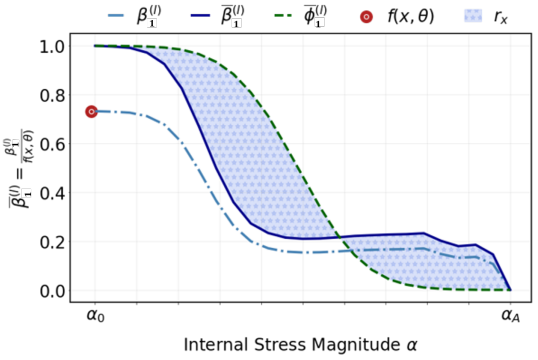
Similarly, the baseline network performances for filters h_2 and h_3 are $\overline{\phi}_{2,\alpha_i}^{(l)}$ and $\overline{\phi}_{3,\alpha_i}^{(l)}$. In Eq. (4.10), the fraction $\frac{\phi_1^{(l)}}{\phi^{(l)}}$ is a ratio between the number of parameters removed to the total number of parameters in 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. 4.4a). 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. 4.4b). The underling 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 [117]. The characteristics of the baseline network performance, describes a network with parameters that, when filtered, proportionally influence the network performance. From Eq. (4.10), the proposed baseline network performance is linked to the the compactness of the network/layer; the characteristics of the baseline network performance is inversely proportional to the compactness ratio of the network/layer weights.



(a) Baseline Network Performance.



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

Fig. 4.4 (a) Baseline network performance (green dotted line) $\bar{\phi}_1^{(l)}$ [Eq. (4.10)] 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 network response $\bar{\phi}_1^{(l)}$ (green dotted line) is the area [Eq. (4.14) and Eq. (4.15)] of synaptic robustness.

For a specific non-random synaptic filtering method, the compactness characteristics of a network is 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. (4.7)], h_2 [Eq. (4.8)] and h_3 [Eq. (4.9)] 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:

$$\beta = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} = \begin{bmatrix} p(\hat{y}_\alpha = y|f(\tilde{\theta}_1,x)) \\ p(\hat{y}_\alpha = y|f(\tilde{\theta}_2,x)) \\ p(\hat{y}_\alpha = y|f(\tilde{\theta}_3,x)) \end{bmatrix}. \tag{4.11}$$

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.

$$\beta_\varepsilon = \begin{bmatrix} \beta_{1,\varepsilon} \\ \beta_{2,\varepsilon} \\ \beta_{3,\varepsilon} \end{bmatrix} = \begin{bmatrix} p(\hat{y}_\alpha = y | f(\tilde{\theta}_1, x_{\delta_\varepsilon})) \\ p(\hat{y}_\alpha = y | f(\tilde{\theta}_2, x_{\delta_\varepsilon})) \\ p(\hat{y}_\alpha = y | f(\tilde{\theta}_3, x_{\delta_\varepsilon})) \end{bmatrix} \tag{4.12}$$

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 compared. Thus, through a comparison of β and β_ε (see Fig. 4.5), we expose the specific parameters (targeted parameters) that are either negatively, invariently or positively affecting the synaptic filtering performances for the adversarial dataset, compared to the clean dataset.

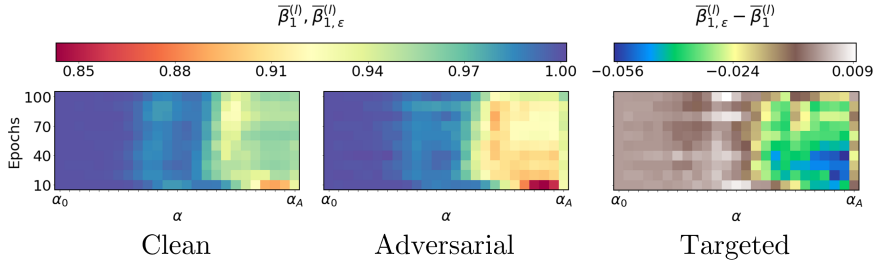


Fig. 4.5 Learning *landscape* of layers and the *regime change* of test accuracy. Targeted parameters of ResNet-18 trained on MNIST using filter h_1. Showing the combined responses for layer 'layer3.0.conv2', measured every 10 epoch up to 100 epochs. The difference between clean (left) and adversarial (middle) responses results in the targeted parameters (right). Every pixel on the clean and adversarial images represents the network test accuracy and for targeted image it is the difference between former two over all evaluated epochs and α.

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. 4.3(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 $\bar{\beta}$ as the performance 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:

$$\tilde{\beta}_i = g(\bar{\beta}_i) = \frac{1}{|h|} \sum_{j \in h} \bar{\beta}_{j,i} \quad \text{for } i = 1, \ldots, |\alpha|. \tag{4.13}$$

Similarly, the combined adversarial network performance $\tilde{\beta}_\varepsilon$ is computed by replacing $\bar{\beta}$ with $\bar{\beta}_\varepsilon$ in Eq. (4.13), where $\bar{\beta}_\varepsilon$ is the performance of the perturbed network (synaptic filtering performance) relative to the unperturbed network performance $f(x_\varepsilon, \theta)$ on adversarial perturbed datasets x_ε. Although the combined network performances $\tilde{\beta}$ and $\tilde{\beta}_\varepsilon$ offer a 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 $\tilde{\beta}_\varepsilon$ 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 $\tilde{\beta}_\varepsilon$ for all network parameters (i.e., global analysis).

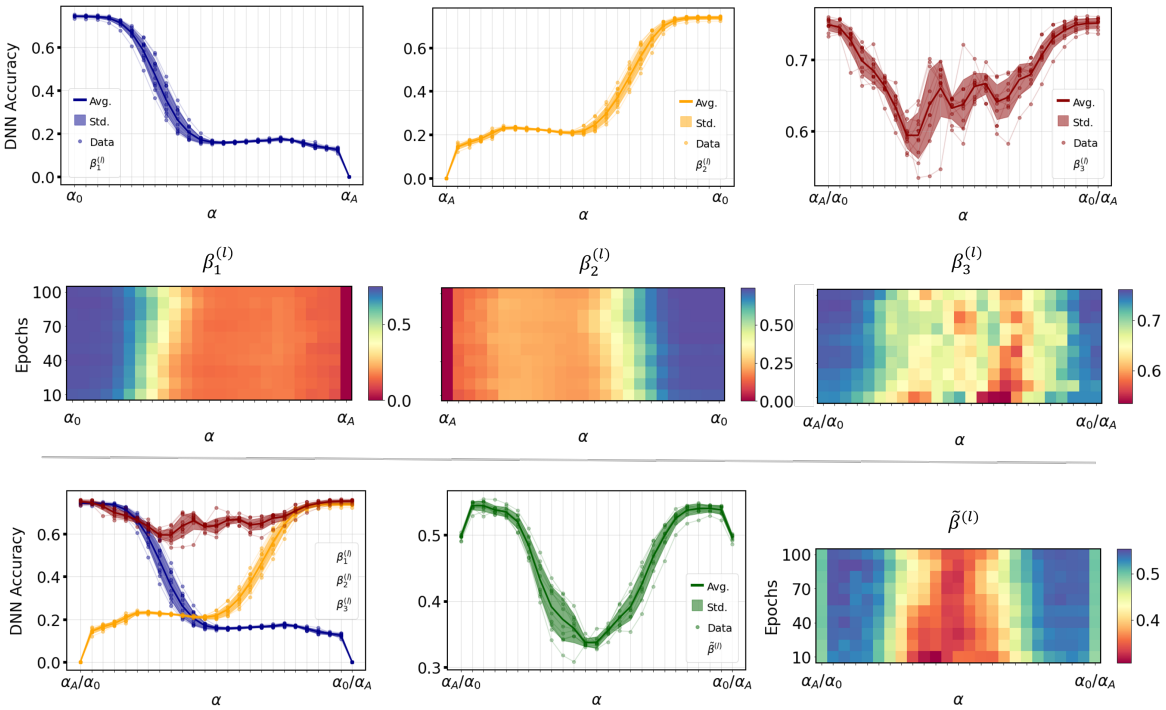


Fig. 4.6 Network accuracy results of the synaptic filtering procedure applied to layer 'conv1' of ResNet-18 trained on CIFAR10, shown to illustrate the combined system response (green line and shaded area). Synaptic filtering performance results for filter h_1 (blue lines and shaded area), h_2 (yellow lines and shaded area), and h_3 (red lines and shaded area).

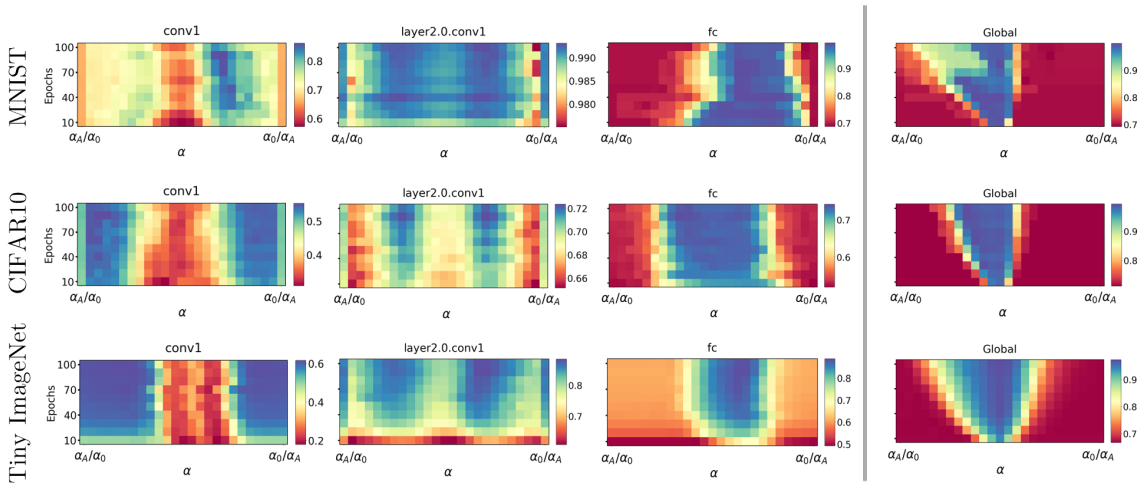


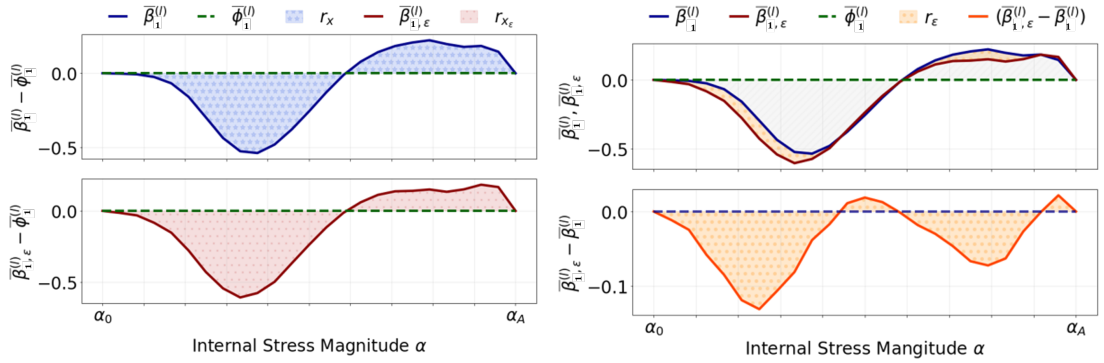
Fig. 4.7 Combined synaptic filtering performances for ResNet-18 trained on CIFAR10, MNIST and Tiny ImageNet datasets every 10 epochs up to 100 epochs. (**1**) Local layer-wise system response to the filtering methods for all α values. (**2**) Global network responses using the full network for all α values. Pixel intensities on the shown images represents the average network accuracy using the different synaptic filters on the clean test dataset, for each α_i in α.

4.4.2 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. (4.10)] and the procedure is captured in Fig. 4.4. 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 $\hat{\overline{\beta}}^{(l)}$ [Eq. (4.13)].

Parameter score for clean data

We take the baseline network performance $\bar{\phi}_1^{(l)}$ [Eq. (4.10)] 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 and Adversarial Performances

(b) Parameter Score Computation

Fig. 4.8 (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 $\bar{\beta}_{1,\varepsilon}^{(l)}$ (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 $\bar{\beta}_{1,\varepsilon}^{(l)}$ datasets over normalised α (top). The area r_ε [Eq. (4.16)] is the adversarial parameter score (bottom).

a clean dataset is r_x is shown in Fig. 4.8a(top)] and given as:

$$r_x = \sum_{i=0}^{|\alpha|} (\bar{\beta}_1^{(l)} - \bar{\phi}_1^{(l)}) \Delta_\alpha, \tag{4.14}$$

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.

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 $\bar{\beta}_{1,\varepsilon}^{(l)}$ to give the parameter

characterization score r_{x_ε}, as per:

$$r_{x_\varepsilon} = \sum_{i=0}^{R} (\bar{\beta}_{1,\varepsilon}^{(l)} - \bar{\phi}_1^{(l)}) \Delta_\alpha, \tag{4.15}$$

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

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 $\overline{\beta}_{1,\varepsilon}^{(l)}$ 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. 4.8b (bottom) and expressed as:

$$r_\varepsilon = \sum_{i=0}^{R} (\overline{\beta}_{1,\varepsilon}^{(l)} - \overline{\beta}_1^{(l)}) \Delta_\alpha. \tag{4.16}$$

If the network performs equally to clean and adversarial datasets for all filtering thresholds α the value of $r_\varepsilon = 0$. Where $r_\varepsilon < 0$, 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, $r_\varepsilon > 0$ 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.

4.4.3 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 three datasets; MNIST [151], the collection of handwriten digits from 0-9, CIFAR10 [5], which is a cluster of 32x32 RGB pixel images representing 10 different classes, and the Tiny ImageNet [173] datasets, which is a group of 64x64 RGB pixel images representing 200 different classes. 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 [173].

State-of-the-art DNNs studies On the benchmark datasets, we train ResNet-18, ResNet-50 [170], SqueezeNet v1.1 [171] and ShuffleNet V2 x1.0 [172]. 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 [184] method. We use a cross-entropy loss function and the Adam optimizer [185] configured with $\gamma = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\lambda = 0$ and $\varepsilon = 1 \times 10^{-08}$ 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 Section 4.5 are shown for the saved networks.

Adversarial attack on datastes For the adversarial attack, we use the single-step FGSM attack [51] and analyze the difference in network performance on the test set to the proposed synaptic filtering methods (Section 4.4). The effectiveness of an adversarial attack on a given dataset can be attributed

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

Collection of results We normalize the r_x and r_{x_ε} parameter score values from Section 4.4.2 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 Section 4.4.2 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 [186]) and trained networks. In order to satisfy constraint 3 from Section 4.4, we use a line search algorithm to find the optimal ε value for each model and dataset, that satisfies: $f(x+\delta_\varepsilon, \mathbf{W}) \approx 0.5 \cdot f(x, \mathbf{W})$.

4.5 Results and Analysis

The results of global (full network parameters) and local (network layer parameters) analysis shown in Figs. 4.9 and 4.10 describes the fragility, robustness, and antifragility characteristics of parameters (cf. Section 4.4.2 and Figs. 4.10 and 4.9). Furthermore, the results show the adversarially targeted (r_ε) parameters (cf. Section 4.4.2 and Figs. 4.10 and 4.9). We identify parameter characteristics that are invariant for clean and adversarial datasets across different datasets and networks. Further local layer results on the network performances to synaptic filtering and adversarial attacks are presented in Appendix A.

Fragile, Robust, and Antifragile The global parameter scores for networks on different datasets are shown in Fig. 4.9. 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 provides a *learning landscape* to examine individual network parameters (Fig. 4.10). 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

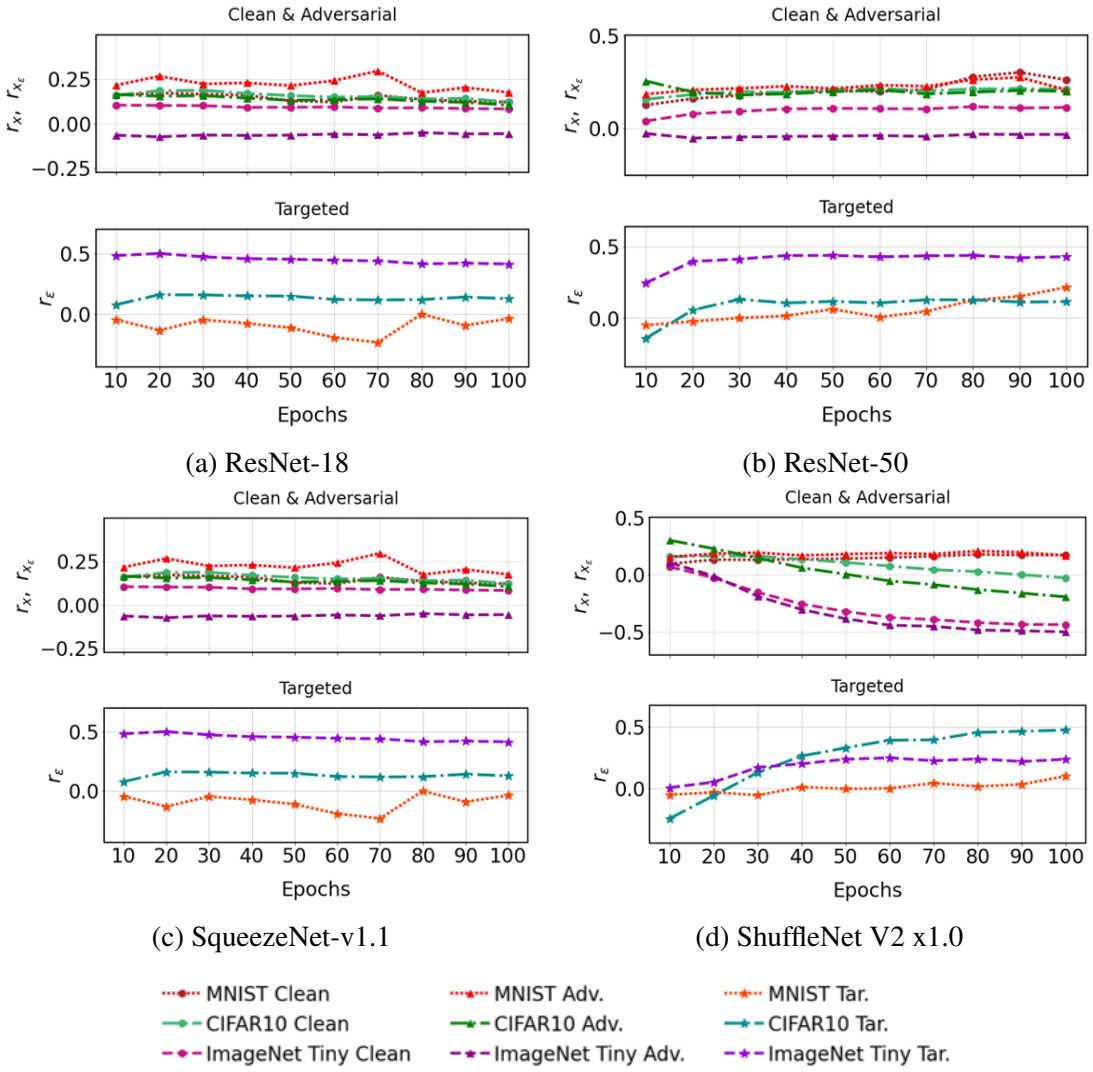


Fig. 4.9 Global parameter scores of (a) ResNet-18, (b) ResNet-50, (c) SqueezeNet-v1.1 and (d) ShuffleNet V2x1.0 over all datasets are r_x, r_{x_ε} and r_ε, measured every 10 epochs up to 100 epochs for the whole network parameters using synaptic filter h_1. The parameter score interpretation is given in Section 4.4.2 and Section 4.4.2.

shown in Fig. 4.10a ImageNet Tiny; Fig. 4.10c MNIST, CIFAR10 and ImageNet Tiny, and Fig. 4.10d ImageNet Tiny. We see the presence of robust parameters (marked as 'Ro') in Fig. 4.10a CIFAR10 and ImageNet Tiny; Fig. 4.10b ImageNet Tiny; Fig. 4.10c MNIST and CIFAR10, and Fig. 4.10d CIFAR10 ImageNet Tiny. Antifragile parameters (marked as 'Af') are distinctly visible in Fig. 4.10a MNIST and CIFAR10; Fig. 4.10b MNIST, CIFAR10 and ImageNet Tiny; Fig. 4.10d MNIST and CIFAR10. Furthermore, periodic robustness characteristics are shown in Fig. 4.10a ImageNet Tiny; Fig. 4.10b MNIST, CIFAR10 and

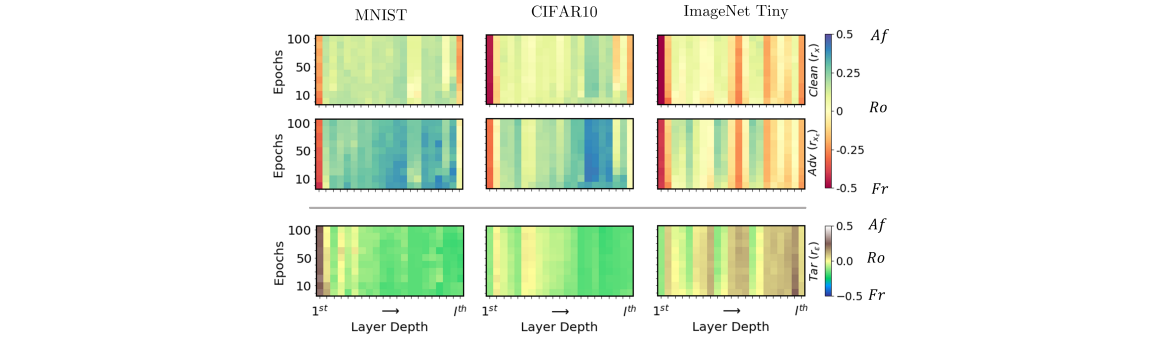
ImageNet Tiny; Fig. 4.10c MNIST, CIFAR10, ImageNet Tiny, and Fig. 4.10d CIFAR10 and ImageNet Tiny.

Adversarially Targeted Parameters In Fig. 4.5, 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 (Section 4.4.2) from Fig. 4.9 and Fig. 4.10. In Fig. 4.10, 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. 4.10c 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. 4.10a MNIST, CIFAR10 and ImageNet Tiny; Fig. 4.10b MNIST, CIFAR10 and ImageNet Tiny; Fig. 4.10c MNIST, and Fig. 4.10d 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. 4.10a MNIST and CIFAR10; Fig. 4.10b MNIST, CIFAR10 and ImageNet Tiny, and Fig. 4.10d MNIST and CIFAR10.

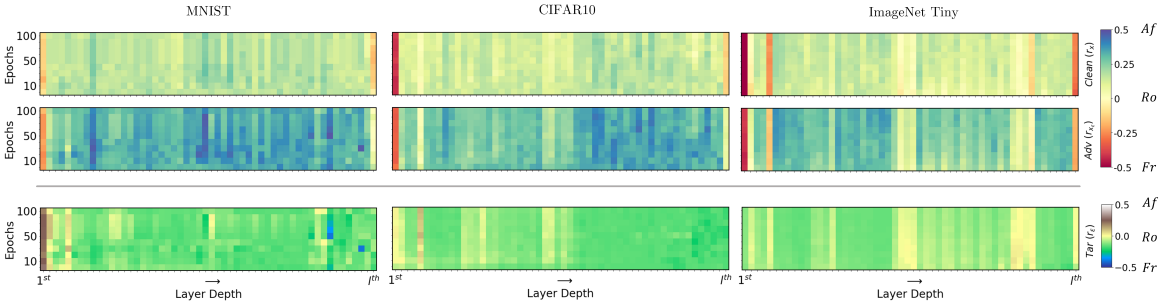
Adversarial Robustness and DNN Compression The results for the synaptic filtering procedure applied to the different networks and datasets evaluated, in relation to network compression, is presented in Figure 4.11 and Figure 4.12. The results on compression show the network test accuracy plotted against the fraction of parameters retained, alternatively, the number of parameter unaffected by the synaptic filtering procedure of network parameters over the total number of parameters. We can observe that when network compression is considered, the performance of networks is similar across datasets.

In Figure 4.11, we can notice a similar gradual drop off in network test accuracy for the ImageNet Tiny dataset measured from all networks, as the fraction of parameters retained approaches 0, signifying the removal of all evaluated parameters. We also find that we are able to retain small fractions of parameters and still maintain sufficient network performance, as shown in the results for ResNet-18 (MNIST, CIFAR10), ResNet-50 (MNIST, CIFAR10, ImageNet Tiny), SqueezeNet-v1.1 (CIFAR10), ShuffleNet V2 x1.0 (MNIST).

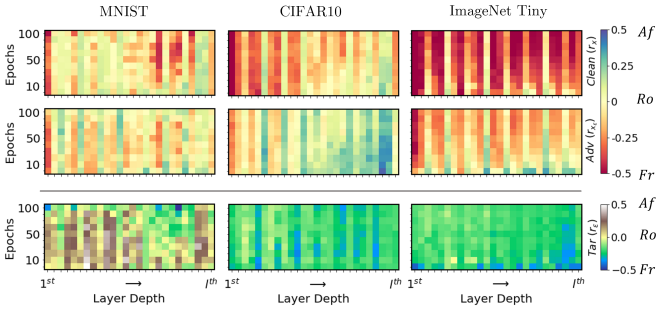
The response to synaptic filter h_2, as presented in Figure 4.14, shows similar gradually declining performance characteristics to that of filter h_1 on the ImageNet Tiny dataset, for all networks. We also observe difference in the responses to filters h_1 and h_2. In the performances of ResNet-18 (MNIST, CIFAR10), ResNet-50 (MNIST, CIFAR10), ShuffleNet V2x1.0 (CIFAR10), we notice imrpoved performances when using the h_2, compared to the



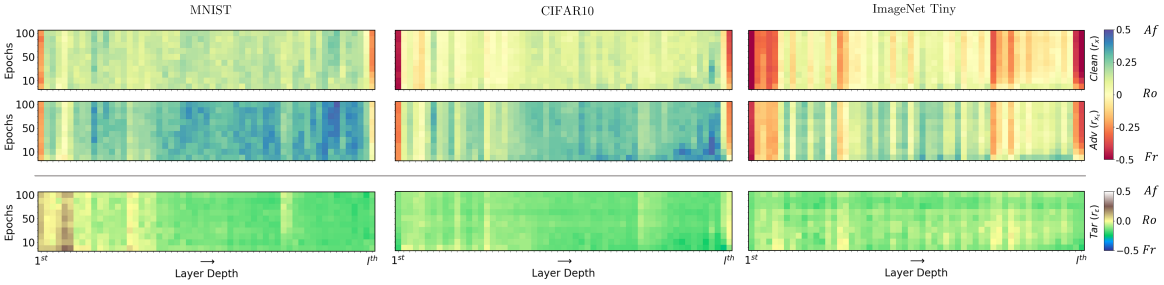
(a) ResNet-18



(b) ResNet-50



(c) SqueezeNet-v1.1



(d) ShuffleNet V2 x1.0

Fig. 4.10 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 Section 4.4.2 and Section 4.4.2.

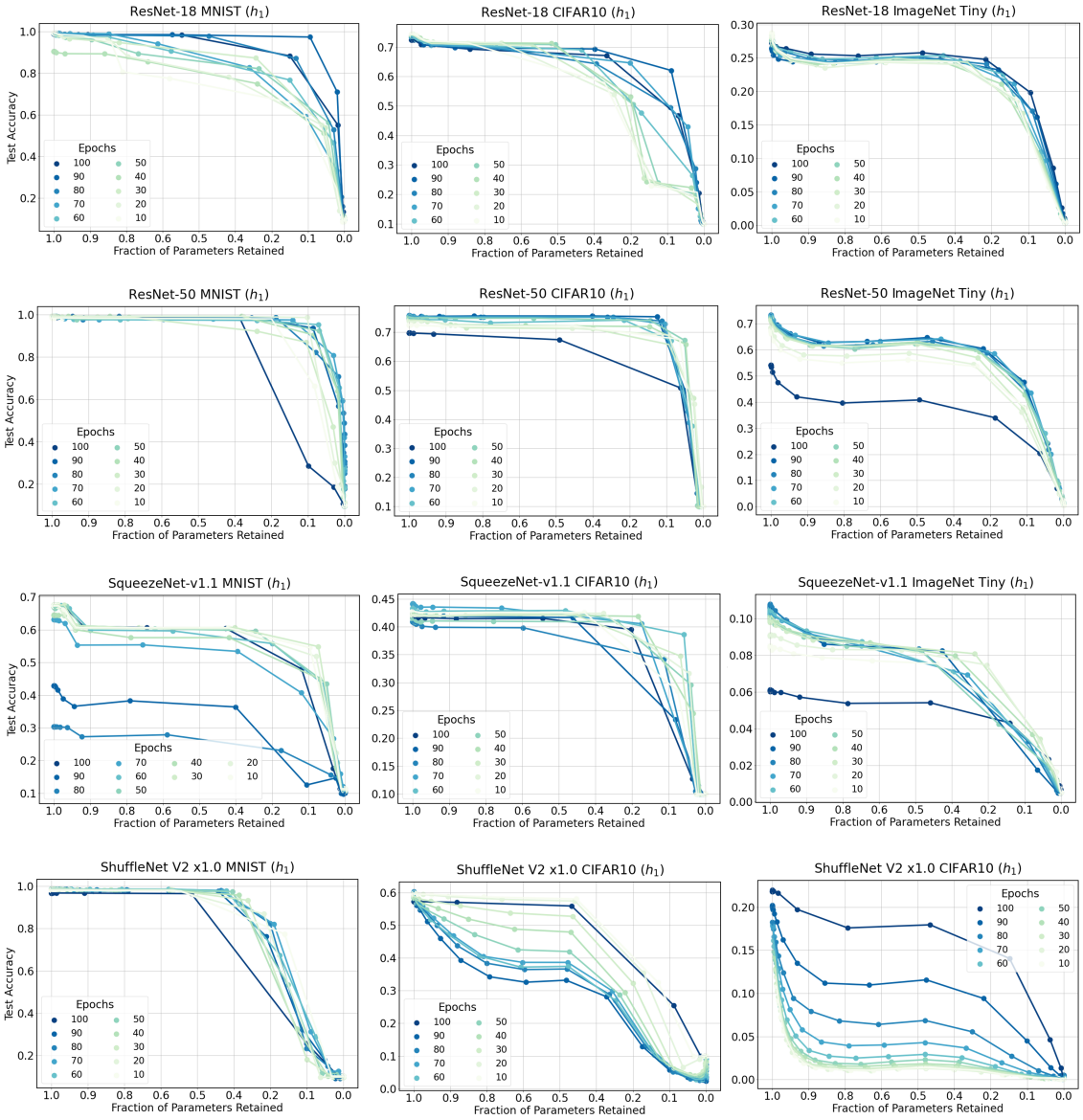


Fig. 4.11 Network performance to the synaptic filtering procedure showing test accuracy as the dependent variable to the fraction of parameters retained post-filtering. Results are presented for filter h_1 for the global filtering procedure on all network layers, showing the averaged results from 10 individual trials on 10 initialised and trained networks. The results are given for networks trained to 100 epochs (dark blue plots) and measured at 10-epoch intervals beginning from epoch 10 (light green plots).

h_1 filter, for all epochs and lower fractions of parameters retained. This signifies that the two filters affect the network differently and this is evident from the performance responses.

Furthermore, we present examples of local layer synaptic filtering results for ResNet-18 on the MNIST dataset in Figure 4.13 for filter h_1 and Figure 4.14 for h_2. The results are

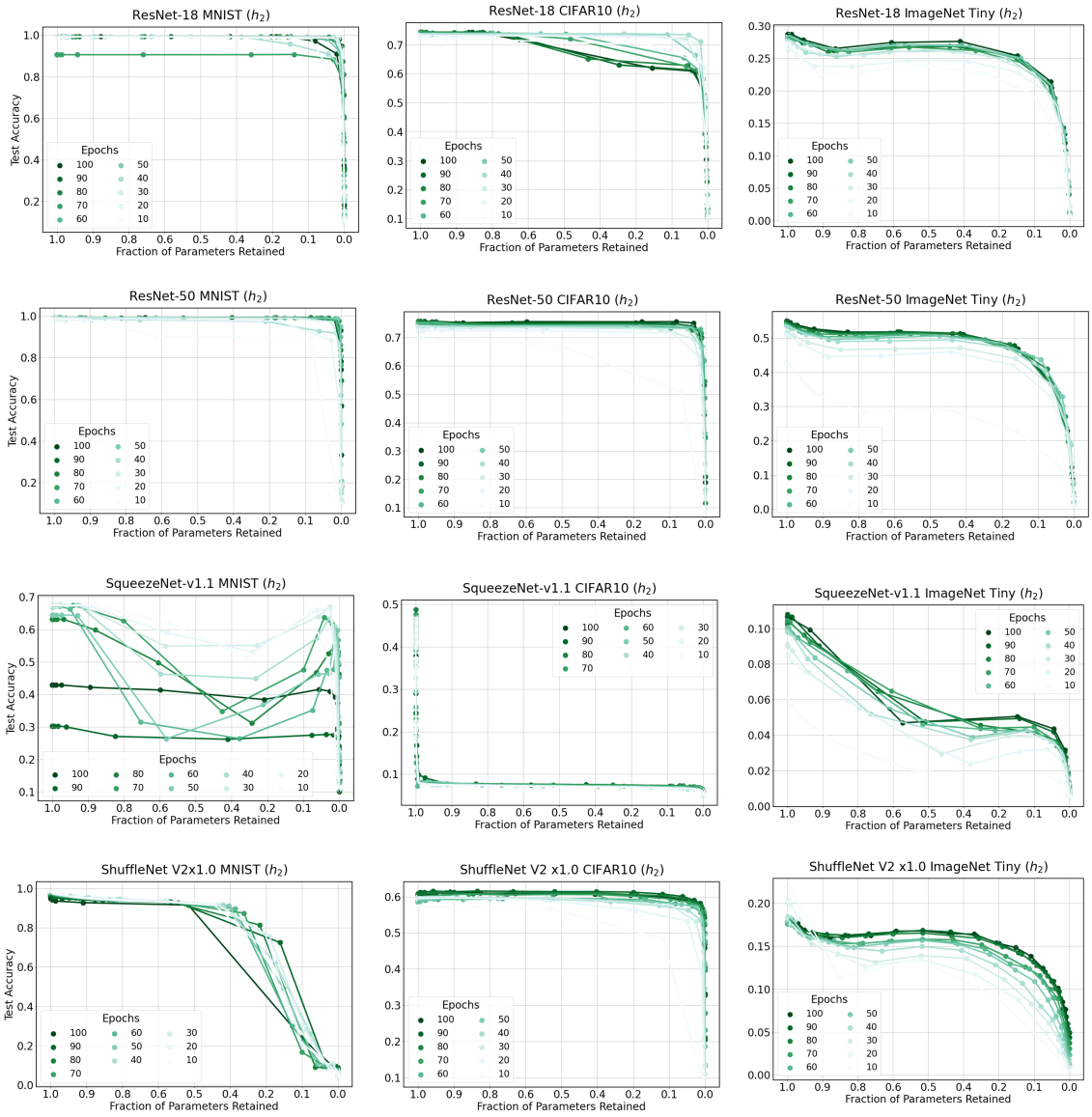


Fig. 4.12 Network performance to the synaptic filtering procedure showing test accuracy as the dependent variable to the fraction of parameters retained post-filtering. Results are presented for filter h_2 for the global filtering procedure on all network layers, showing the averaged results from 10 individual trials on 10 initialised and trained networks. The results are given for networks trained to 100 epochs (dark green plots) and measured at 10-epoch intervals beginning from epoch 10 (light green plots).

shown for convolutional layers 'conv1', 'layer2.1.conv1' and the fully connected 'fc'. We present both the regular test dataset accuracy, as well as the adversarial dataset accuracy for comparison and give the scaled difference of clean and adversarial performances (c.f. Fig-

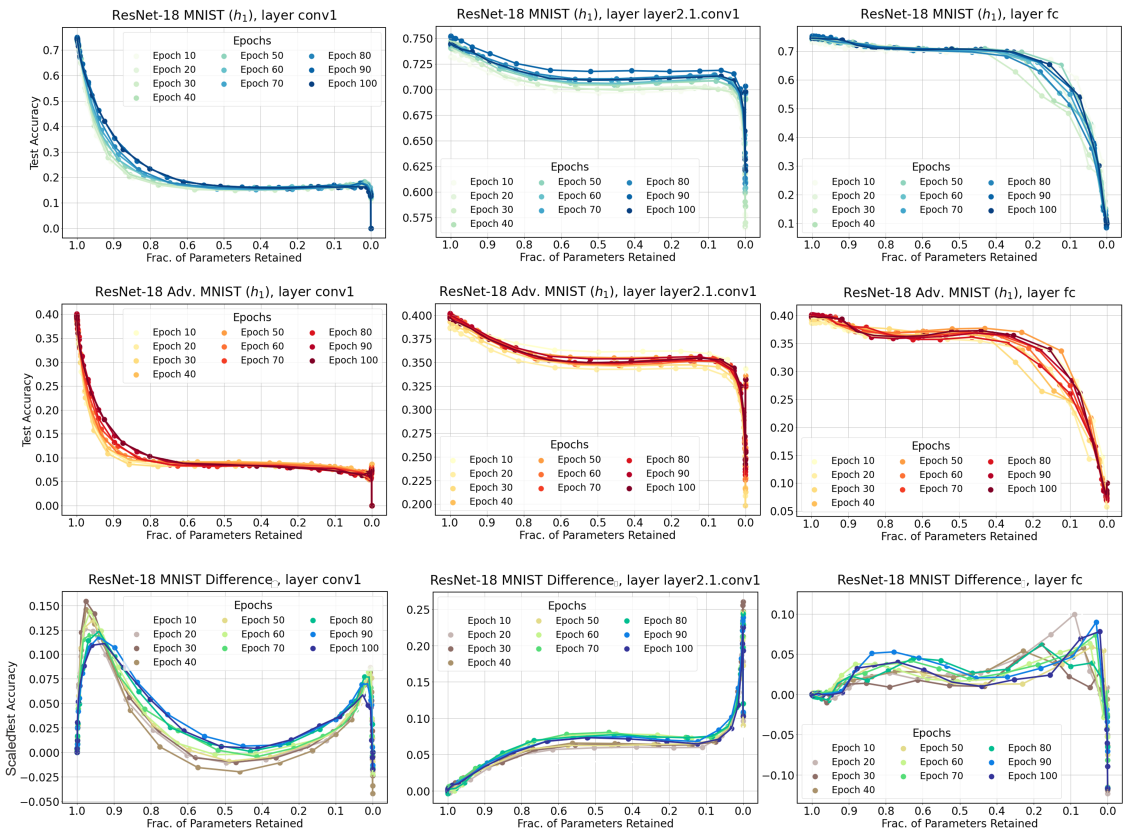


Fig. 4.13 Network performance to local layer-wise application of the synaptic filtering procedure, showing the test accuracy as the dependent variable to the fraction of parameters retained post-filtering. Results are presented for filter h_1 applied to layers 'conv1', 'layer2.1.conv1' and 'fc'. Showing the averaged results from 10 individual trials on 10 initialised and trained networks. Results are shown for on the clean dataset performance (light blue to dark blue), the adversarial dataset (light yellow to dark red), and the scaled performance difference (white to blue).

ure 4.4). The scaled difference between the adversarial and clean dataset performances show how the network responds to both the adversarial attack and synaptic filtering simultaneously.

For both filters h_1 and h_2, we can observe similarities in the scaled difference of network performances, for the same layer. Taking layer 'conv1' as example, in Figure 4.13 we notice a sharp and large scaled difference between clean and adversarial dataset, which gradually decreases and increases before sharply decreasing to as we filter all parameters. A similar response characteristic is also evident from Figure 4.14. Layers 'layer2.1.conv1' and the 'fc' layer show similar behaviour for both filters. Furthermore, we can notice that we are able to reduce the fraction of parameters retained in the layer, particularly layer 'fc', and still maintain sufficient test performance.

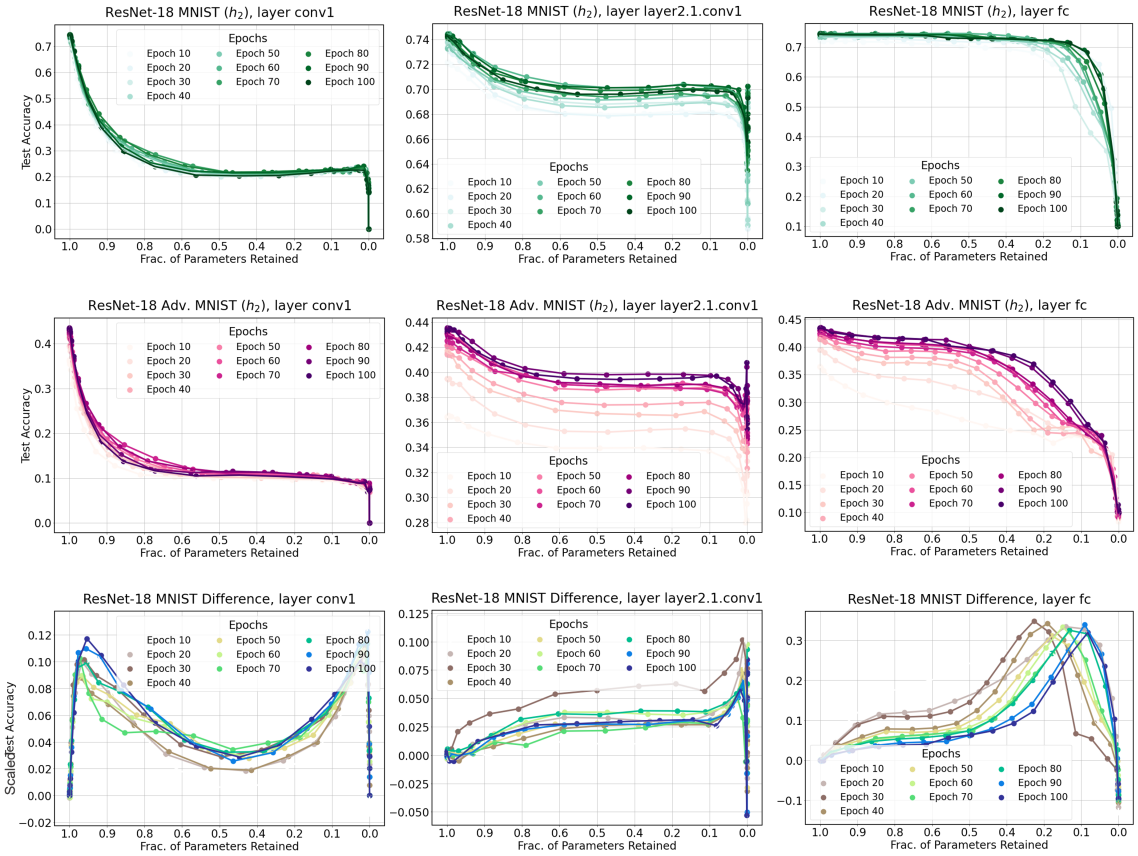


Fig. 4.14 Network performance to local layer-wise application of the synaptic filtering procedure, showing the test accuracy as the dependent variable to the fraction of parameters retained post-filtering. Results are presented for filter h_2 applied to layers 'conv1', 'layer2.1.conv1' and 'fc'. Showing the averaged results from 10 individual trials on 10 initialised and trained networks. Results are shown for on the clean dataset performance (light green to dark green), the adversarial dataset (light pink to dark purple), and the scaled performance difference (white to blue).

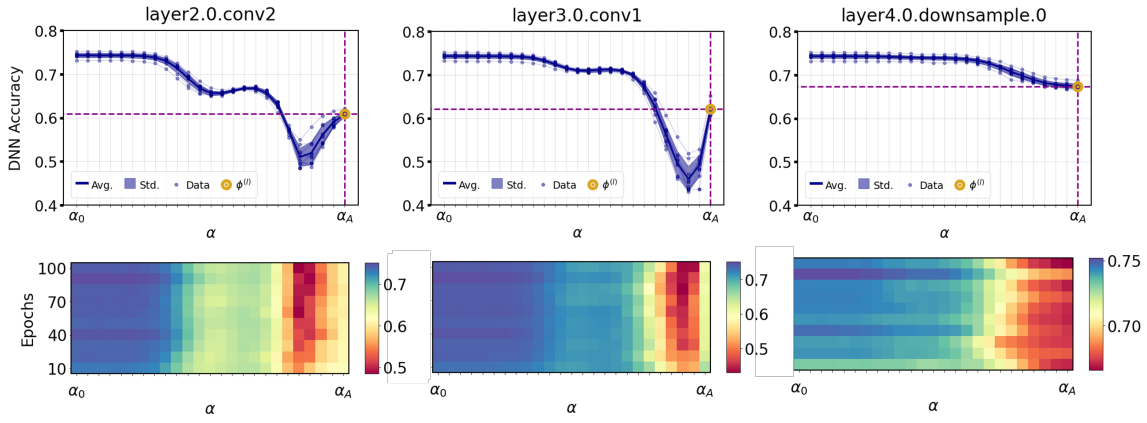
Effects of Batch Normalization We investigated the phenomenon of the network retaining classification performance despite all features at layer l being removed (see column α_A in Fig. 4.5). Even as the network layer parameters are filtered maximally, resulting in a null layer, the network is able to achieve sufficient performances on the test set, respective to both the clean and adversarial dataset performances pre-filtering.

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 [80]. When implementing a network architecture, we utilise the standard models in accordance

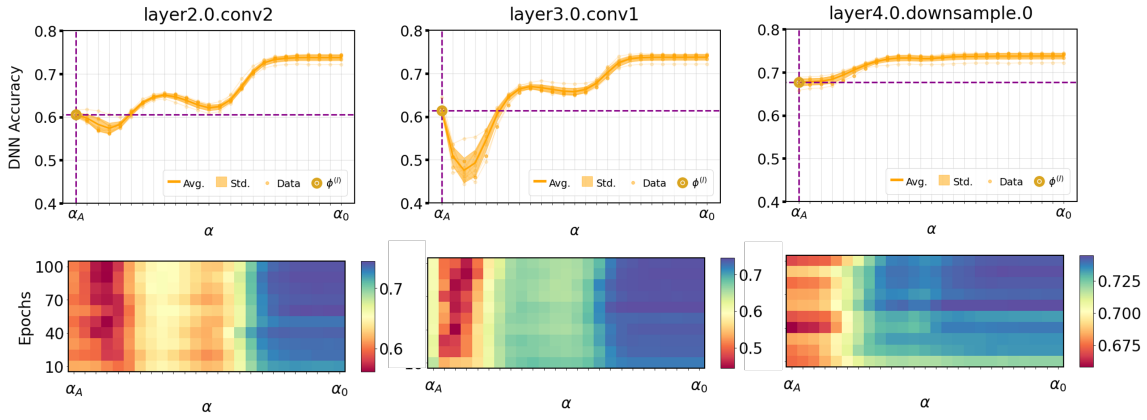
with literature; the functionality of batch normalization layers are also predefined and remain unaltered in our analysis. Consider the condition where a convolutional layer l has been filtered maximally using a synaptic filter, the subsequent batch normalization computation is given as:

$$\hat{y}^{(l)} = \frac{x^{(l)} - \mathbb{E}[x]}{\sqrt{\text{Var}[x] + \varepsilon}} * \gamma^{(l)} + \beta^{(l)} \tag{4.17}$$

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}).



(a) Filter h_1



(b) Filter h_2

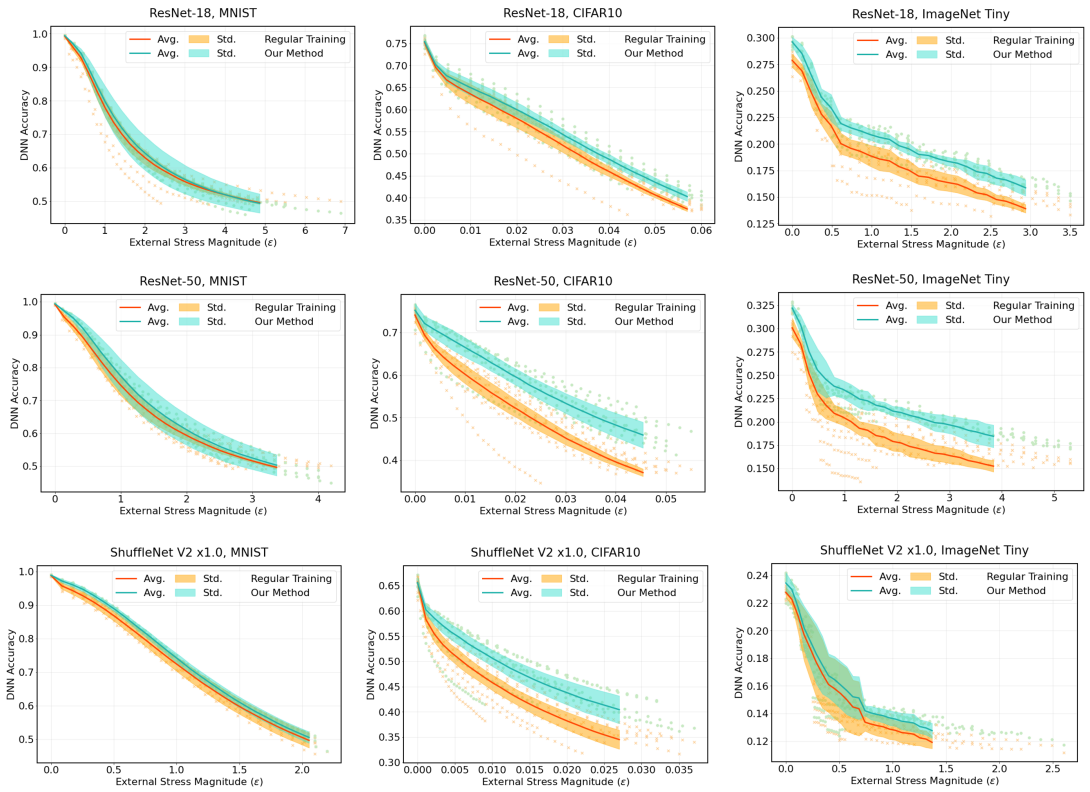
Fig. 4.15 Synaptic filtering network performances of ResNet-18 trained on CIFAR10 for layers 'layer2.0.conv1', 'layer3.0.conv1' and 'layer4.0.downsample.0'. Purple dotted lines marked at $\Phi^{(l)}$ (gold dot marker) show the network performance when the maximum number of parameters are filtered using the synaptic filters.

Implementations of networks compute the expectation and variance from Eq. (4.17) 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. 4.15, 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 of Robust and Antifragile Parameters Upon identifying robust, fragile and antifragile parameters using the difference in parameter scores (see Section 4.4.2) we consider fragile parameters to be parameters that, when perturbed, result in a 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 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.

The results from our selective backpropagation method is shown in Fig. 4.16 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 been seen from the results, our proposed method, shown in teal, out-performs 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. 4.16b) and ImageNet Tiny (Fig. 4.16c) dataset compared to the MNIST (Fig. 4.16a) dataset. The effectiveness of the selective backpropagation method on CIFAR10 and ImageNet Tiny



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

Fig. 4.16 Network accuracy of ResNet-18 on (a) MNIST, (b) CIFAR10, (c) ImageNet Tiny datasets to external stress (adversarial attack) with magnitude ε using the FGSM attack.

compared to MNIST can be attributed to the complexity of the datasets [166], where MNIST can be considered to have a lower complexity relative to CIFAR10 and ImageNet Tiny.

4.6 Summary

We can examine deep neural networks using our proposed synaptic filtering technique to characterize parameters of the network as fragile, robust and antifrgaile 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. $\pm 2\%$ 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 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 analyse the performance of DNN architectures through a selective backpropagation technique, where only the identified robust and antifrgaile parameters are retrain upon identification of parameter characterisations, at the evaluated epoch. The selective backpropagation technique is compared to the regular training procedure to show that retraining only robust and antifrgaile parameters improves DNN robustness to adversarial attacks on all evaluated datasets and network architectures. Furthermore, we also see instances of the test accuracy on the standard dataset increase with the use of selective backpropagation. We successfully implement a novel method of making DNNs more robust that, built on knowledge of the parameter characterisations, entails less computations than regular training. The selective backpropagation technique, and the prerequisite task of identifying fragile, robust, and antifrgaile parameters is applied to the DNN tasks outlined in Chapter 5 to evaluate and improve the performance of custom DNN architectures and original datasets.

Chapter 5

Robustness of Deep Learning in Real-World Applications

In the following chapter, we apply the robustness analysis methods developed in Chapters 3 and 4, on real-world problems that are comprised of custom datasets and novel network architectures. The two tasks we consider are (1) The preprocessing of inherently noisy radar signals using a DNN-based denoising filter selection algorithm, and (2) the classification of human activities using 24 GHz radar and deep learning. Thus, the chapter is divided into Section 5.1 and Section 5.2 corresponding to tasks (1) and (2) respectively. It should be noted that the filter selection algorithm detailed in Section 5.1 acts as a preliminary preprocessing stage of the activity classification network developed in Section 5.2. The robustness analysis method developed in Chapter 3 and Chapter 4 are applied to both the filter denoising selection algorithm presented in Section 5.1 and the activity classification network presented in Section 5.2.

5.1 Signal Denoising Filter Selection Algorithm

In this section, we propose a novel deep learning based denoising filter selection algorithm for noisy Electrocardiograph (ECG) signal preprocessing with the aim of applying the DNN-based denoising method for filtering noisy radar signals. ECG signals measured under clinical conditions, such as those acquired using skin contact devices in hospitals, often contain baseline signal disturbances and unwanted artefacts; indeed for signals obtained outside of a clinical environment, such as heart rate signatures recorded using non-contact radar systems, the measurements contain greater levels of noise than those acquired under clinical conditions. In this section we focus on heart rate signals acquired using non-contact radar

systems for use in assisted living environments. Such signals contain more random noise than those measured under clinical conditions, and thus require a novel signal noise removal method capable of adaptively determining denoising filters. Currently the most common method of removing noise from such waveforms is through the use of filters, the most popular filtering method of which is the wavelet filter. There are, however, circumstances in which using a different filtering method may result in higher signal-to-noise-ratios (SNR) for a waveform. In this section, we investigate the wavelet and elliptical filtering methods for the task of reducing noise in ECG signals acquired using a convolutional neural network (CNN). Our proposed CNN architecture classifies (with an initial accuracy of 92.8%) the optimum filtering method for noisy signal with respect the the expected SNR value, using which the network is trained.

We further modify the network using the proposed selective backpropagation training procedure detailed in Chapter 3 and Chapter 4 that increases the test accuracy of the network from 92.8%, using regular training, to 96.3%, using the proposed methodology. Furthermore, we evaluate the network to two different types of adversarial attacks; the fast gradient sign method (FGSM) attack and the projected gradient descent (PGD) attack to anlayse the adversarial robustness of the CNN. WE find that the selective backpropagation methodology is successful in making the CNN more robust to the FGSM and PGD adversarial attacks. We further apply the signal denoising selection algorithm presented in this section, to the application of activity signal denoising recorded using UWB radar systems detailed in Section 5.2. The purpose of the CNN methodology developed in this section is to be used as an automated preprocessing tool by which noisy, real-world recorded signals can be denoised using the optimal denoising filter.

5.1.1 Overview of The Application

With the emergence of embedded devices capable of artificial intelligence and learning abilities, there remains the problem of effective data processing, which has been a task for engineers and data analysts alike since the development of embedded systems entirely [187]. Acquiring signals and analysing information (signal processing) from the natural phenomena that occurs in the real-world comes with various difficulties; none more so prevalent than the task of differentiating between information that is important for analysis, from that which is not. Signals acquired using assistive technologies from applications outside of a laboratory setting are inherently subjected to greater levels of signal fluctuations from the recording environment. These fluctuations, considered as noise to the system, are often detrimental to

the analysis of the objective event. The role of a digital signal processor, for the majority of the data used in modern applications is considered electronically, is to adequately remove unwanted artefacts or disturbances from the recorded signals that represent some target, real-world event. We propose a Deep Neural Network (DNN) architecture to recognise minute variations of an objective signal and classify the optimum method for removing noise from the waveform. The objective application of this study is to outline an effective and automated denoising method for Electrocardiograph (ECG) signals measured using ultra-wideband (UWB) radar systems, such that the method can be used in real world settings, such as assisted living environments.

The application of the ECG signal denoising filter selection algorithm is extended to the task of activity signal denoising in Section 5.2. The work outlined in this section considered ECG signals as the target signal, however, the proposed methodology is able to be adapted for denoising other types of signals, such as activity signals recorded using UWB radar. Measurements made using UWB radar systems use micro-Doppler signatures to identify subtle movements from an individual's body, such as movement [142, 72], heart rate and respiration rate [148, 40]. Due to the unobtrusive method of data acquisition, recorded signals are often contaminated with varying levels of noise from different sources that need to be removed in order to analyse the vital signs effectively [39] efficiently. We propose an automated method of removing noise from inherently noisy ECG signal recordings using a DNN architecture to learn the subtle variations of signals.

Oftentimes the disparity between studies carried out under clinical environments and the technology being used in real-world applications stem from the attention paid to the data being considered, or there lack of. Classical machine learning models, and more recently deep learning models, have already made significant strides in recognising important information from noisy and often weak signals [188, 189]. The need for an automated model to recognise random, seemingly unpredictable variances in the environment is therefore needed to understand how best to process raw signals. For this study, the signals considered as the base ECG waveforms are from the MIT-BIH normal sinus rhythm database [190]. In this section, we propose a model to predict the best method for removing unwanted artefacts from a noisy ECG waveform. In regular applications, disturbances in a recorded waveform are removed using various signal processing techniques; methods primarily based around filtering-out unwanted noise perturbations and retaining the important features from the signal. The task of filtering a raw signal involves determining the optimum filter, along with other hyperparameters, such that the filter is able to effectively remove noise present in the signal. This requires careful consideration and analysis of the data at hand, often in the

frequency domain. Furthermore, static methods of filtering where the system function does not adapt in behaviour to changes in the input, are at risk of attenuating important features from the waveform, or equally, being unable to sufficiently remove certain aspects of noise, both of which may affect further analysis of the evaluated signal [191].

Signals measured outside of a laboratory, such as assisted living spaces where environmental factors are seldom controlled, the acquired waveforms are intrinsically noisy and irregular, thus such signals require a filtering methods that are able to adapt to random variations. Haykin [192] details the process of adaptive filtering where an optimisation algorithm is used to determine the adjustable filter parameters, however, such a method is still frequently inadequate in dealing with subtle deviations of the noise distribution, particularly where the noise in a signal varies irregularly, requiring a more stochastic approach to filtering noise [193]. Deep learning models in contrast, have the innate capacity for recognising detailed patterns, and as such, make them ideal for differentiating signals with minute differences between the waveforms. The intended real-world applications for such models are assistive technologies that are used to remotely monitor the cardiac health of individuals in an environment. This task also introduces a constraint of signal windowing in order to reduce the number of elements to be considered at a given time, such that the signals may be analysed in real-time. This constraint consequently, works to reduce the computational complexity of the overall process [194].

In Section 5.1.4, we detail how commonalities can be found between different signals that result in higher SNR values for a given filter, thus also outlining the training process of the proposed model. The architecture outlined in this study is that of a binary classifier and is designed to predict an optimum filtering method, between wavelet filtering and elliptical filtering, given a noisy ECG input signal. The Elliptical filter was chosen as an alternative to the wavelet filter due to its narrow transition response at the cut-off frequency, respective to other similar finite impulse response (FIR) filter functions such as the Chebyshev and Butterworth filters [191]. After classifying the optimum filter, a newly presented waveform is filtered using the filter label and predefined filter coefficients, which are determined during the model training process.

The section is organised as follows: Section 5.1.2 contains a review of relevant techniques and methods explored in this Section, Section 5.1.3 and Section 5.1.3 outlines a definition of the signals used, model training algorithm is defined in Section 5.1.3, classification model parameters in Section 5.1.3, feature reduction techniques used in Section 5.1.3 and finally, the experimental set-up in Section 5.1.3. This is followed by the results and discussions in Section 5.1.4, which details the experimental results 5.1.4 and a discussion of the results 5.1.4.

In conclusion, we make remarks on the study carried out and possible future avenues of research are presented in Section 5.1.6.

5.1.2 Related Work

Physiological signals have been recorded and studied extensively for many years, none more so than human vital signs, particularly ECG signals. The analysis of ECG signals is vital in diagnosing, and often treating, the cardiac health of a person, which has until recently been a task carried out by medical professionals exclusively [195]. With the power of modern machine learning tools and the availability of assistive technologies to monitor, process and analyse ECG signals remotely and in real-time, the task of detecting and predicting cardiac abnormalities have become a task for medical professionals and machine learning practitioners alike [70]. We focus on the noise perturbations of ECG signals, which varies from one segment of the signal to another and is often unable to be removed effectively with standard filtering methods [194]. For such instances, machine learning and deep learning networks have previously been applied to remove noise [150, 69]. The effectiveness of these models, however, decreases as the noise power within the signal increases, as is the nature signals acquired using UWB radar systems.

With instances of vital signs monitoring using UWB radar systems, various investigations have been carried out into ECG signal denoising and classification tasks [40, 148]. The studies described go as far as providing a holistic overview of the data acquisition process, and demonstrating various signal processing techniques to extrapolate vital signs form radar signatures. Similar to ECG detection, Liang et al. [196] proposed a method of detecting respiration signs using a frequency accumulation algorithm followed by the discrete short-time-Fourier transform to suppress random signal harmonics that are the combined products of heartbeat and respiration signals. A model proposed by Shikhsarmasr et al. [149] makes use of the wavelet packet decomposition method to suppress random noise in ECG signals, and subsequently makes use of a vital sign estimation model on a defined region of interest to improve the overall system efficiency. The wavelet transform has shown to be used in many similar studies [41, 197], where a thresholding method is used to attenuate certain frequency amplitudes and suppress others.

There exist studies using deep learning and machine learning algorithms in various forms, to carry out the task of ECG signal denoising. Antczak [150], proposed using synthetic data to train a deep algorithm for signal denoising and fine-tuning the network parameters to learn higher-level features using real data. More recently, a layer-by-layer denoising neural

network has been developed based on factor analysis [198], where the model attempts to learn Gaussian noise present in the signal, thus being able to remove it. A similar study has been carried out for an audio equalisation task by Pepe et al. [199], in which, given a noisy signal, the FIR filter coefficients are predicted using a DNN architecture. Indeed, various attempts have also been made into developing learning models capable of determining optimum filtering coefficients for reducing noise in ECG signals [197, 200]. To the best of our knowledge and through a review of relevant literature, we for the first time propose a novel study detailing the use of machine learning techniques for the task filter selection in ECG signal denoising.

5.1.3 Denoising Filter Classification Modelling

The primary application for the model proposed in this study is to classify the optimum method of attenuating unwanted artefacts from noisy ECG signals acquired using an UWB radar system. To simulate additional noise to the signals, as would be expected from radar data of this nature [201], noise perturbations drawn from a Gaussian distribution were added to the original dataset. The new signals were then processed using a wavelet filter and a low-pass elliptical filter to determine the optimum method for reducing the noise in a signal. Finally, a classification model was trained and tested using the labelled dataset for the purpose of predicting the optimum filtering method that results in the highest SNR value. This process is possible due to the fact that the additive noise is known pre-classification, and thus the SNR values can be calculated accurately.

ECG Signal Definition

A raw ECG waveform is a continuous-time signal and through sampling, can be viewed as a discrete time signal $x(k)$, as per the following definition:

$$x(k) \triangleq x(t) \mid t = kT, \tag{5.1}$$

where $k = \{1, \ldots, K\}$ and represents the number of discrete data points in the waveform. The parameter T is the sampling period of the discrete-time signal and thus, the sampling frequency is given as $\nu_s = 1/T$. The raw signal is represented by $x(k)$, however, having been originally recorded under clinical conditions, is regarded as the optimum ECG waveform for this investigation. This primary aim of this study is to improve the signal quality of ECG

signals recorded using UWB radar systems, which exhibit greater levels of noise than the raw signal used in this study.

To modify the raw signal in order to make it more noisy, artificial noise perturbation $g(k)$ is generated using the Gaussian distribution function as per:

$$g(k) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(r(k)-\mu)^2}{2\sigma^2}}, \tag{5.2}$$

where $r(k)$ is a random value between 0 and 1 and μ and σ, are the mean and standard deviation of the signal $x(k)$. The noise is modelled on the statistical characteristics of the original signal itself. The waveform to be considered by the classifier is $z(k)$; a noisy ECG signal that is generated by combining the original signal $x(k)$ with the generated Gaussian distributed noise $g(k)$, as per:

$$z(k) = x(k) + g(k). \tag{5.3}$$

The objective of the classification model is to determine the optimum method to reduce the noise perturbation $g(k)$, from the noisy signal $z(k)$.

Filter Label Identification Algorithm

The resultant signal $z(k)$ is foremost normalised using min-max normalisation, and subsequently processed through the SNR optimisation function $\Omega(\cdot)$. The SNR optimisation function returns the signal label y, as shown in (5.6), which is a composition of α, the optimum filtering label; β which is maximum SNR value from the optimisation function, and δ, which the filter hyperparameter value. We reshape the clean signal $x(k)$ and noisy signal $z(k)$ of lengths K into a $M \times N$ shaped matrices as follows:

$$\mathrm{X} \in \mathbb{Q}^{M\times N}, \mathrm{Z} \in \mathbb{Q}^{M\times N} \mid \mathbb{Q}^{M\times N} \leftarrow \mathbb{Q}^{1\times K}, \tag{5.4}$$

where $\mathrm{X} = [\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^M]^T$ and $\mathrm{Z} = [\mathbf{z}^1, \mathbf{z}^2, \ldots, \mathbf{z}^M]^T$ are the clean and noisy signals reshaped into a matrix of windowed signals. The reshaping parameters M and N are non-zero and are defined as $M = K/\lambda$ and $N = \lambda$. The chosen window period is given as λ. The variables m and n are the matrix indices given as natural numbers, such that $m = \{1, 2, \ldots, M\}$ and $n = \{1, 2, \ldots, N\}$. This process is carried out to reshape the original signal of length K into

M windowed signals of length N, an example for Z is shown in the following form:

$$\mathrm{Z} = \begin{bmatrix} \mathbf{z}^1 \\ \mathbf{z}^2 \\ \vdots \\ \mathbf{z}^M \end{bmatrix} \equiv \begin{bmatrix} z_1^1 & z_2^1 & \dots & z_N^1 \\ z_1^2 & z_2^2 & \dots & z_N^2 \\ \vdots & \vdots & \ddots & \vdots \\ z_1^M & z_2^M & \dots & z_N^M \end{bmatrix}, \tag{5.5}$$

where the noisy windowed signal is given as $\mathbf{z}^m = (z_1^m, z_2^m, \dots, z_N^m)$, and equally, the windowed clean signal is given as $\mathbf{x}^m = (x_1^m, x_2^m, \dots, x_N^m)$. The function $\Omega(\cdot)$ is applied to the resultant windowed signals and returns the corresponding filter labels y^m, given by:

$$y^m = (\alpha^m, \beta^m, \delta^m) = \Omega(\mathbf{x}^m, \mathbf{z}^m, \theta_f). \tag{5.6}$$

The maximum value of the filter boundary parameter is θ_f and f is the filter label. We assign $f = 0$ for elliptical filter and $f = 1$ for wavelet filter. The variable θ_f depends on the filter being used, since the elliptical and wavelet filters have different types of hyperparameters. When considering the elliptical filter, θ_0 represents the maximum cut off frequency for a low pass filter. As for the wavelet filter, θ_1 represents the maximum number of wavelets to be investigated and must satisfy $\theta \leq \Theta$, where Θ is the maximum number of wavelets available. The clean signal $\mathbf{x}^m$, is used within the SNR optimisation function for calculating the individual SNR values of the filtered signals.

For a given filter f and the windowed signal index m, the maximum SNR value β_f^m, as shown in (5.7), and optimum filter variable δ_f^m that returns β_f^m, as shown in (5.8), is obtained by:

$$\beta_f^m = \max[\mathrm{SNR}(\mathbf{x}^m, \mathbf{r}_f^m)] \tag{5.7}$$

$$\delta_f^m = \underset{\omega_c}{\operatorname{argmax}}[\mathrm{SNR}(\mathbf{x}^m, \mathbf{r}_f^m)]. \tag{5.8}$$

The resultant signal of function $R_f(\mathbf{z}^m, \omega_c)$ for a filter f, and windowed signal with index m, is $\mathbf{r}_f^m$. The filtered signal $\mathbf{r}_f^m$ carries the same length as $\mathbf{z}^m$, such that $\mathbf{r}_f^m = (r_{f,1}^m, r_{f,2}^m, \dots, r_{f,N}^m)$. The filter hyperparameters is $\omega_c \in \mathbb{N}$ and $\theta_{f,0} \geq \omega_c > \theta_f$, where the term $\theta_{f,0}$ holds the initialising value of ω_c. As the elliptical and wavelet filters accept different function hyperparameters, the values of ω_c and $\theta_{f,0}$ carry different representations for each filter. For the elliptical filter, where $f = 0$, the value $\theta_{0,0}$ is the lowest frequency to be tested and is chosen to be 1Hz. Given the condition where $f = 1$, the value of $\theta_{1,0}$ is the first element from an ordered set containing numerically encoded wavelets, such that each element from the set can

be decoded to retrieve its corresponding wavelet type. The function $R_f(\mathbf{z}^m, \omega_c)$ is defined as:

$$\mathbf{r}_f^m = R_f(\mathbf{z}^m, \omega_c) = \mathbf{T}_f^{-1}[H_f(\omega_c) \cdot \hat{\mathbf{z}}^m], \tag{5.9}$$

where $\mathbf{T}_f^{-1}$ is the general inverse transform operator. The condition that when $f = 0$, the inverse transform operator $\mathbf{T}_f^{-1}$ is the inverse fast Fourier transform (FFT), and when $f = 1$, the inverse transform operator is the inverse discrete wavelet transform (DWT). This transformation is applied to a convolution of the filter function $H_f(\omega_c)$ and $\hat{\mathbf{z}}^m$. Where $\mathbf{z}^m$ is the windowed signal transformed in the Fourier domain, given the condition $f = 0$, and in the wavelet domain, given the condition $f = 1$, and ω_c represents the filter hyperparameters.

When using the elliptical filter function $H_0(\omega_c)$, the Nyquist theorem must be satisfied before applying the filter, as shown:

$$H_0(\omega_c) = \{\Psi_p(\omega, \omega_0) \mid \omega_0 = \frac{\omega_c}{0.5\ \nu_s})\}\ , \tag{5.10}$$

$$\Psi_p(\omega, \omega_0) = \frac{1}{\sqrt{1 + \varepsilon^2 R_p^2(\xi, \frac{\omega}{\omega_0})}} \tag{5.11}$$

where ω_0 is the cut-off frequency, ω is the angular frequency given as $2\pi\nu$ (where ν is the ordinary frequency in Hz), ε is the ripple factor and ξ is the selectivity factor. The p-th-order elliptical filter is indicated by $\Psi_p(\omega, \omega_0)$, and the function R_p referred to as a Chebyshev rational function that controls the stopband ripple response. The ripple factor specifies the passband ripple, whereas the stopband ripple is given by the combination of the ripple factor and selectivity factor.

Similarly, an example function $H_1(\omega_c)$ representing a wavelet filer is used. When we use the wavelet filtering method, we apply DWT using a wavelet, denoted by ω_c, that results in a transformed signal $\hat{\mathbf{z}}^m$. Following this, we apply thresholding to the wavelet detail coefficients u using a soft threshold $\hat{u}$, defined in [202], as per:

$$u = \begin{cases} [sgn(u)](|u| - \hat{u}) & |u| \geq \hat{u} \\ 0 & |u| < \hat{u} \end{cases} \tag{5.12}$$

$$\hat{u} = \sigma\sqrt{2\log N}\ , \tag{5.13}$$

where N and σ are the length and standard deviation of the input signal $\mathbf{z}^m$ respectively. The noise present in the signal is then attenuated using a thresholding function, given by (5.13), before reconstructing the original signal using the inverse of DWT, as shown in (5.9).

From (5.7) the value of β_f^m is given as the maximum SNR value achieved for a windowed signal m being processed through filter f and it can be seen from (5.8) that δ_f^m takes the value of ω_c for which β_f^m is achieved. The standard SNR function used in both (5.7) and (5.8) is given by:

$$\mathrm{SNR}(\mathbf{x}^m, \mathbf{z}^m) = 20\log_{10}\left(\frac{\sum_{n=1}^{N}[x_n^m]^2}{\sum_{n=1}^{N}[x_n^m - z_n^m]^2}\right), \tag{5.14}$$

where $\mathbf{x}^m$ is the clean signal and $\mathbf{z}^m$ is the signal to be compared. The root-mean-square error (RMSE) calculation is used in Section 5.1.4 to compare the average power difference between noisy signals and their corresponding clean signals, as given by (5.15). The definition of RMSE is the standard deviation of the residuals (predicted errors) between two signals $\mathbf{x}^m$ and $\mathbf{z}^m$, given by:

$$\mathrm{RMSE}(\mathbf{x}^m, \mathbf{z}^m) = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(x_n^m - z_n^m)^2}\,. \tag{5.15}$$

The equivalent values of the window label y^m, comprised of the filter label α^m, the maximum SNR value β^m, and the optimum filter variable δ^m, are given as per:

$$\alpha^m = \begin{cases} 0 & \text{if } \beta_0^m \geq \beta_1^m \\ 1 & \text{if } \beta_0^m < \beta_1^m \end{cases} \tag{5.16}$$

$$\beta^m = \max\{\beta_0^m, \beta_1^m\}. \tag{5.17}$$

Here the filter label α^m and the maximum SNR value β^m are shown by (5.16) and (5.17) respectively. Note that the individual signal filter labels are assigned using (5.16), depending on which maximum SNR value, for a given filter β_f^m, results in the highest overall SNR value, when comparing signals $\mathbf{x}^m$ with $\mathbf{r}_f^m$ according to (5.7).

The values of δ^m are given by (5.18) and show that for instances where $\alpha^m = 0$, the value of δ_0 is assigned to δ^m and conversely, where $\alpha = 1$ the value of δ^m is given as δ_1.

$$\delta^m = \begin{cases} \delta_0 & \text{if } \alpha^m = 0 \\ \delta_1 & \text{if } \alpha^m = 1 \end{cases} \tag{5.18}$$

$$\delta_0 = \frac{1}{M} \sum_{m=1}^{M} (\delta_0^m) \tag{5.19}$$

$$\delta_1 = mode(\delta_1^m) \ \forall m \ . \tag{5.20}$$

Equation (5.18) and (5.19) show that δ^m is equal to δ_0, which is the average value of δ_0^m for all values of m. It can be seen from (5.18) and (5.20) that δ^m is is equal to δ_1, and is the modified mode function $mode(\delta_1^m)$ that returns the most frequent element of δ_1^m for all values of m. Given the condition where multiple values appear equally as frequently in δ_1^m, the $mode(\delta_1^m)$ function returns a randomly selected δ_1^m from the subset of most frequent appearing values. For the case where all values of δ_1^m appear equally as frequently, signifying that $M \leq \Theta$, the modified mode function returns a randomly chosen value of δ_1^m.

For convenience in Section 5.1.3, the labelling algorithm details operations for one signal $z(k)$ of length K reshaped into a matrix of M windowed signals of length N, however, it should be understood that the complete model is trained on multiple noisy signals reshaped into windowed waveforms. Thus, in the training process there will be a concatenation of multiple signals $z(k)$, each windowed into MxN matrices. Further details on this matter are presented in Section 5.1.3.

Classification Model

The task of identifying the optimum filtering method between a low-pass elliptical filter and a wavelet filter can be formulated as a binary classification problem. As such, various machine learning models, including DNN models, can be applied to this problem. In this section, we propose a convolution neural network (CNN) classification model for the given task. The CNN model is chosen due to its ability to learn unknown variations in the input distribution in the input data [203], such as noise. This model is compared against other machine learning models, such as support vector machines (SVM), logistic regression, K-nearest neighbours (KNN) and a DNN.

The hyperparameters for DNN is be shown in Fig. 5.1, which details the number of layers, activation functions, pooling layers and flattening layers used. To summarise, a 128 dense layer, followed by a max-pooling layer of size 2 with an equal stride value with a stride value, and a 64 dense layer were added, the dense layer using rectified linear unit (ReLu) activation functions. This is followed by a 32 dense layer and a flattened 16 layer, all using the ReLu activation. The ReLu activation function was chosen based on fine tuning of the model on the training dataset. The final layer consisted of 2 dense layers using a *SoftMax*

activation function. This model was configured with a kernel size of 3. The optimisation function chosen for this model was Adam: a method for stochastic optimization [204] with parameters: learning rate = 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 1e-07$. A categorical cross-entropy function was used to calculate loss and the model was trained for 20 epochs of time and a batch size of 16 was used.

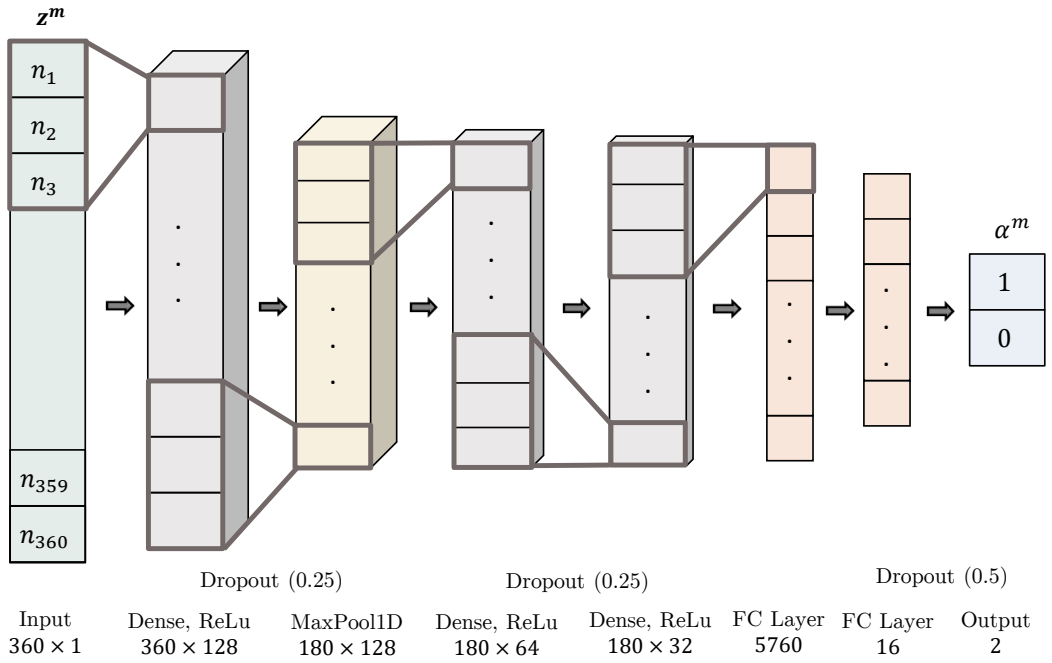


Fig. 5.1 The DNN was configured with the following parameters in sequential order: a 128 dense layers, a maximum pooling layer with pooling size of 2, a 64 dense layers, a 32 dense layers and a 16 dense layers, all using ReLu activation functions. A unit dropout rate of 25% was used after the 128 dense layer and 64 dense layer, followed by a 50% rate after the flattened 16 dense layer, this was applied in order to avoid overfitting [35]. An stochastic gradient decent (SGD) optimizer using backpropagation was used for the learning method and the model was trained for 20 epochs.

Feature Reduction Techniques

Two different feature reduction techniques, principal components analysis (PCA) and independent analysis (ICA), were used for the models evaluated for this task. Reducing the dimensions of the data being considered, particularly using techniques that retain any significant information from the data such as PCA and ICA, have shown to be effective in improving classification accuracies in tasks involving ECG signal analysis [205].

When applying PCA, the dimension of the data was reduced by 95% such that only the top 5% of the principal components were used. For ICA, 36 independent components were used, being 10% of the initial signal length. These parameters for PCA and ICA were chosen to retain the most important features whilst reducing the length of the data, thus improving the performance of the classification models.

Experimental Set-up

For the classification task at hand only the filter labels α^m from y^m for a windowed noisy signal $\mathbf{z}^m$ are required for the filter classification model. The parameter δ^m is used for selecting the data being applied to the models and β^m is used to assign the optimum filter with an appropriate filter parameter. Particularly when calculating δ^m, the complete dataset of windowed signals should be used after construction, such that the value of M used for (5.19) and (5.20) is replaced by the total number of windowed signals for all full signals, shown in (5.21).

Using the elliptical filter, given by (5.10) and (5.11), requires the presetting of parameters such as the filter order, passband ripple and stopband ripple. For this study, a 7-th-order filter with a 3 dB passband and 4 dB stopband was chosen. These parameters control the elliptical filter response characteristics and were chosen based on prior testing of signals that returned the best average performance by the filter.

The value of β^m for a windowed signal is used to remove any signal anomalies; there are cases where clean signals from the dataset are corrupted, such that the original waveform shows zero-amplitude and in some cases the clean signal itself exhibits noise. We require an ideal, clean signals in order to control the variables of the study, thus we choose remove signals where the underlying waveforms are distorted. It was observed, from prior analysis of the training data, that instances where signals are corrupted have a β^m value less than -3.00 dB approximately, thus signals where $\beta^m \leq -3.00$ dB are removed from the dataset being used on the models.

Detailed in 5.1.3, the signal reshaping and labeling method is specified for one noisy signal $z(k)$ reshaped into a matrix Z of windowed signals. The proposed models are trained and tested on multiple noisy signals $z^l(k)$, generated from multiple different clean signals $x^l(k)$, where $l = \{1,2,\ldots,358\}$ and the resultant noisy windowed waveforms are given as per:

$$\mathrm{Z}_{total} = [\{\mathrm{Z}_1, \mathbf{y}_1^T\}, \{\mathrm{Z}_2, \mathbf{y}_2^T\}, \ldots, \{\mathrm{Z}_{358}, \mathbf{y}_{358}^T\}] \,. \tag{5.21}$$

A λ value of 360 was chosen, thus M and N values in (5.4) are 10 and 360 respectively. For all 358 signals in our dataset, the total training sets of signal matrix Z is given in (5.21) which adds up to 3580 labelled waveforms (examples of noisy signals. For our classification models, we split the data into training and test sets respectively of sizes 67% and 33%, which gives us 2399 windowed waveforms in training set and 1181 windowed waveforms in the test set.

5.1.4 Results and Discussions

Table 5.1 Classification accuracies of filter selection classifier. Models tested include SVM, logistic regression, KNN, DNN and CNN. Learning models were applied to the regular dataset and dataset after having applied feature reduction techniques such as PCA and ICA, see 5.1.3.

Classification model	Classification accuracy (%)
SVM	89.77
Logistic regression	84.20
KNN	82.06
DNN	89.82
CNN	**92.80**
SVM (PCA)	89.89
Logistic regression (PCA)	84.62
KNN (PCA)	82.23
DNN (PCA)	90.05
CNN (PCA)	92.27
SVM (ICA)	89.78
Logistic regression (ICA)	84.20
KNN (ICA)	84.46
DNN (ICA)	89.81
CNN (ICA)	91.56

The results presented in table 5.1 lists the performance (classification accuracies) of classifiers. It can be seen from Fig. 5.2 that SVM, logistic regression and DNN showed an improved

model performance using only the principal components, compared to using the full data or independent components only. The CNN model achieved the highest classification accuracy for all three forms of input data; regular signal, principal components and independent components.

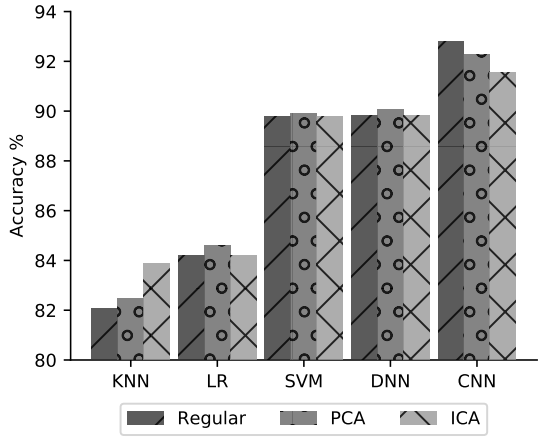


Fig. 5.2 Classification accuracies of SVM, KNN, logistic regression, DNN and CNN for the task of classifying the optimum filtering method for a given windowed signal.

Experimental Results

The results of a test signal $\mathbf{z}^a$, representing a noisy signal with an arbitrary index a, such that $a \in m$ and is applied to both the wavelet filter and elliptical filters; this can be observed from Table 5.2 along with their corresponding waveforms in Fig. 5.3. Comparatively, the filter responses for a signal $\mathbf{z}^b$, where b is an arbitrary signal index such that $b \in m$ and $a \neq b$, is shown in Fig. 5.4 with its corresponding data displayed in Table 5.2. It should be noted that the two signals considered, $\mathbf{z}^a$ and $\mathbf{z}^b$, both result in differing optimum filtering methods.

Discussion

It can be observed from Table 5.2 where the SNR and RMSE values, given by (5.14) and (5.15), for noisy signals $\mathbf{z}^a$ and $\mathbf{z}^b$ are presented with their corresponding filtered signals. The signals $\mathbf{r}_0^a = R_0(\mathbf{z}^a, \delta_0)$ and $\mathbf{r}_1^a = R_1(\mathbf{z}^a, \delta_1)$ are the elliptical and wavelet filtered responses respectively for the noisy signal $\mathbf{z}^a$. Whereas the signals $\mathbf{r}_0^b = R_0(\mathbf{z}^b, \delta_0)$ and $\mathbf{r}_1^b = R_1(\mathbf{z}^b, \delta_1)$ represent the elliptical and wavelet filtered signals respectively for noisy signal $\mathbf{z}^b$.

For noisy signal $\mathbf{z}^a$, the lowest RMSE and highest SNR values were obtained using the elliptical filter response $\mathbf{r}_0^a$. Contrastingly, given the noisy signal $\mathbf{z}^b$, the lowest RMSE and

Table 5.2 Comparing the wavelet filter and elliptical filter on two signals: $\mathbf{x}^a$ and $\mathbf{x}^b$. Where the signals $\mathbf{r}_0^a$ and $\mathbf{r}_0^b$ represent the elliptical filtered responses for $\mathbf{z}^a$ and $\mathbf{z}^b$ respectively. The filtered signals $\mathbf{r}_1^a$ and $\mathbf{r}_1^b$ represent the wavelet filtered responses of $\mathbf{z}^a$ and $\mathbf{z}^b$ respectively. Showing the SNR and RMSE values of the noisy signal, signal filtered by wavelets and signal filtered through elliptical filtering.

Signal	RMSE	SNR (dB)
$\mathbf{x}^a$	-	-
$\mathbf{z}^a$	0.154	6.846
$\mathbf{r}_0^a$	**0.122**	**8.505**
$\mathbf{r}_1^a$	0.138	7.809
$\mathbf{x}^b$	-	-
$\mathbf{z}^b$	0.125	8.914
$\mathbf{r}_0^b$	0.122	9.591
$\mathbf{r}_1^b$	**0.112**	**10.024**

highest SNR values were given by the filtered signal $\mathbf{r}_1^b$, being the wavelet filter response. This proves that for a noisy signal, the successful classification of an optimum filter would indeed result in a response with a higher SNR value and lower RMSE value compared to the alternative filtering method. The results for which presented in Table 5.2 and thus reaffirm the understanding that different windowed noisy signals ultimately result in differing optimum filtering methods, this can be further noticed from Figure 5.3 and Figure 5.4. Such a finding can be attributed to various signal characteristics; the two waveforms presented show differences in noise levels, baseline characteristics and average signal power, which all show to affect the optimum filtering method used for the signal.

From Table 5.1 it can be found that feature reduction methods, such as PCA and ICA, do not necessarily result in the highest classification accuracy for all models. This is evident particularly for the proposed CNN model, where the best classification performance was obtained with the original dataset, without having applied any feature reduction methods. Reduction of the dataset dimension should be dismissed entirely, as the DNN model performance in Fig. 5.2 shows that PCA being used as the input data resulted in the highest classification accuracy for that model. As stated by [199], tasks involving signals with high levels of noise, where filter parameters are to be determined by a learning model, require fine tuning and experimentation. The findings in this section show that it is possible to develop a DNN model to recognise noisy signals based on their optimum predicted filtering method.

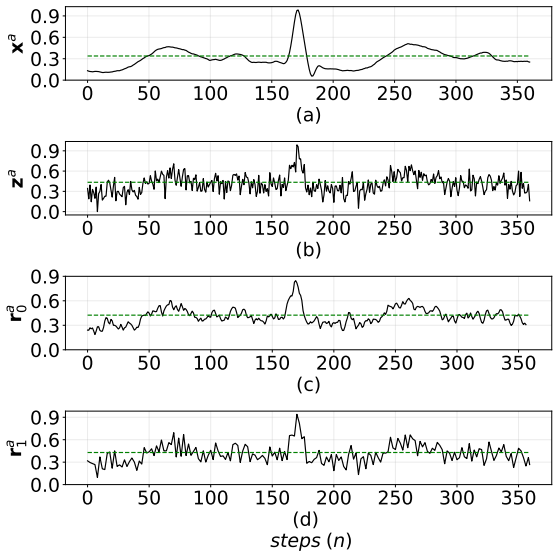


Fig. 5.3 Signals shown where the optimum filter is determined to be the elliptical filter ($\alpha^a = 0$). Showing the clean signal $\mathbf{x}^a$ (a), the signal with additive Gaussian noise perturbations applied to it $\mathbf{z}^a$ (b), the noisy signal after having been processed through an optimum elliptical filter $\mathbf{r}_0^a$ (c), and finally the noisy signal after having been filtered by an optimum wavelet filter $\mathbf{r}_1^a$ (d). Outlined in black are the target waveforms and the green dotted lines represent the RMS values of each signal.

This investigation is an exploratory venture into developing and applying deep learning techniques to tasks involving digital signal processing, specifically for the selection of digital filtering methods. For the task of removing noise from ECG signals we investigate the wavelet filter, as proposed by various studies carried out for similar applications [202, 149, 41, 197], and an elliptical filter, due to its frequency response at the defined cut-off frequency [192, 194]. Consequently, various avenues of research are yet to be explored, such as using novel complex networks capable of processing complex signals and developing graphical representations of signals to be used is conjunction with machine learning models.

5.1.5 Selective Backpropagation for Signal Denoising Filter Selection

Here we present the parameter scores and adversarial robustness of the ECG signal denoising filter selection algorithm following the application of the robustness improvement methods using selective backpropagation, using the three synaptic filters h_1, h_2, and h_3 as outlined in Chapter 4. It should be noted that the identification of of fragile, robust and antifrgaile network parameters, which is a prerequisite of applying the selective backpropagation method, is carried out using on the FGSM attack only. We test the effectiveness of the selective

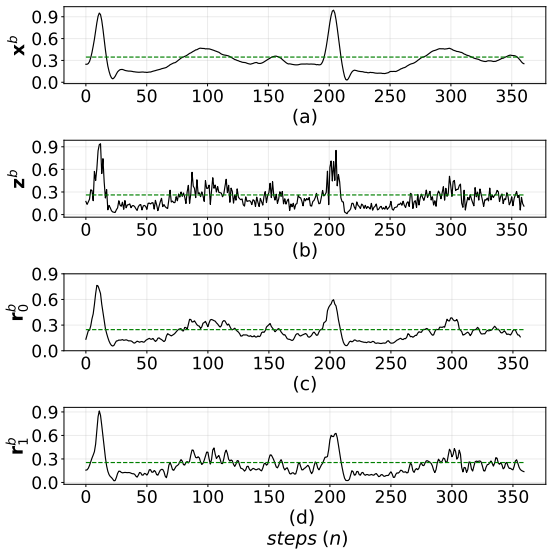


Fig. 5.4 Signals shown where the optimum filter is determined to be the wavelet filter ($\alpha^b = 1$). Showing the clean signal $\mathbf{x}^b$ (a), the signal with additive Gaussian noise perturbations applied to it $\mathbf{z}^b$ (b), the noisy signal after having been processed through an optimum elliptical filter $\mathbf{r}_0^b$ (c), and finally the noisy signal after having been filtered by an optimum wavelet filter $\mathbf{r}_1^b$ (d). Outlined in black are the target waveforms and the green dotted lines represent the RMS values of each signal.

backpropagation method on the fast gradient sign method attack and the projected gradient descent attack (PGD). The PGD attack is considered to be a universal first-order adversarial attack and stronger than the FGSM attack, due to the increased steps taken to increase loss. We test the trained networks on the PGD attack to evaluate the ability of the network to generalise on other adversarial datasets.

From Figure 5.6 and Figure 5.7 we can see that the adversarial robustness of the ECG signal denoising filter selection network is improved as the network is subjected to both the FGSM and PGD attacks. The local layer-wise parameter scores used to identify which of the specific network layers to include, and which to omit, from the training procedure is presented in Figure 5.5.

The three parameter scores for each layer of the network are r_x, r_{x_ε}, and r_ε represent the robustness of the network performance on the clean dataset, adversarial dataset and the difference in the clean and adversarial dataset performances respectively. Using the parameter score r_ε, which is calculated using the difference between r_x and r_{x_ε}, we select only the robust and antifragile network layers ($r_\varepsilon \geq 0$) to include in the selective backpropagation procedure. The purpose of applying the proposed selective backpropagation procedure to the signal denoising filter selection algorithm, is so only the specifically the robust and antifragile

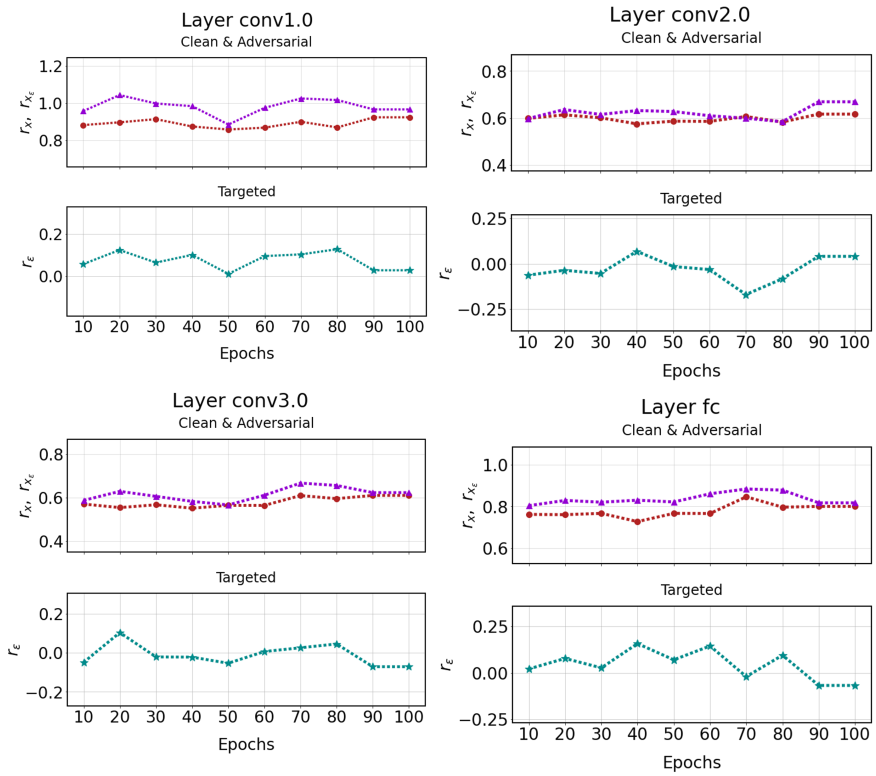


Fig. 5.5 The layer wise parameter scores for the ECG signal denoising filter selection networks applied on the clean dataset (r_x and marked bu the purple dotted lines), the adversarial dataset (r_{x_ε} and marked by the red dotted lines) and the difference in scaled performances (r_ε marked by the teal dotted lines).

parameters of the network that are retrained after identification. The identification of robust and antifragile parameters allow us to focus on the parameters that are most influential to increasing the adversarial performance of the network, which are identified using the FGSM attack during the proposed analysis.

The results observed in Figure 5.6 and Figure 5.7 show that the identification and selective backpropagation of only the robust and antifragile parameters increases the adversarial performance of the network to both the FGSM attack, using which we identify the robust and antifragile parameters to retrain during selective backpropagation, as well as a different adversary; the PGD attack formulation. We show that the proposed analysis and selective backpropagation method applied to a new DNN developed for a custom dataset increases in performance to our method. Furthermore, we notice that the regular network accuracy is also increased using the proposed methodology ($\varepsilon = 0$), compared to regular training. The reasoning for this increase in regular accuracy, as well as the adversarial performance, is due to the dataset of UWB radar signals containing artefacts of additive Gaussian noise that may

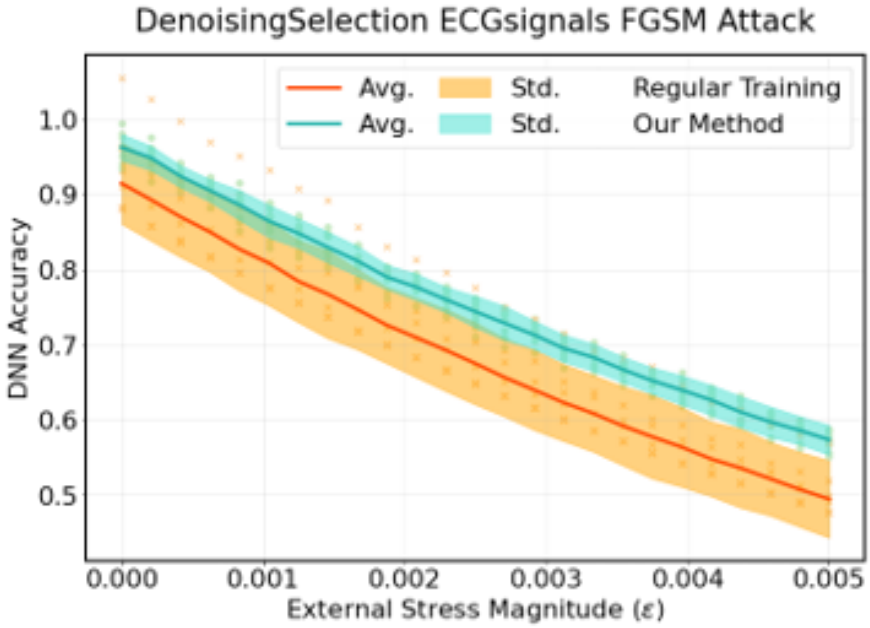


Fig. 5.6 Performances of the proposed ECG signal denoising filter selection network, trained using a regular training procedure and the selective backpropagation method, as presented in Chapter 4. Both the regularly trained network (shown in orange) and the network trained using selective backpropagation (shown in teal) were tested on the FGSM attack with varying perturbation magnitudes to evaluate the effects of the proposed training. Showing the average (labelled 'Avg.' and shown as a solid line) and standard deviation (labelled 'Std.' and shown by the shaded coloured regions) of the results all trained networks evaluated.

limit the ability of the network to perform optimally on the regular, unperturbed, dataset. The consequences of this is an increase in both the adversarial and regular network performances, using a method that retrains specific parameters of the network that are fewer than those retrained using a regular training scheme.

5.1.6 Summary of Signal Denoising Filter Selection Algorithm

We propose a convolutional neural network (CNN) architecture, a variant of a DNN, for classifying an optimum signal denoising filter for a given noisy ECG signal. Furthermore, we introduce an algorithm for labelling of signal waveforms with the optimum denoising filters (elliptical filter and wavelet filter). Our three versions of labelled datasets (full features dataset and reduced features datasets based on principal component analysis and independent component analysis) was fed to various classifiers such as, support vector machine, K-nearest neighbour, logistic regression and deep neural network and CNN. Our CNN model was able to classify the optimum filter with an accuracy of 92.8% when using the full feature dataset.

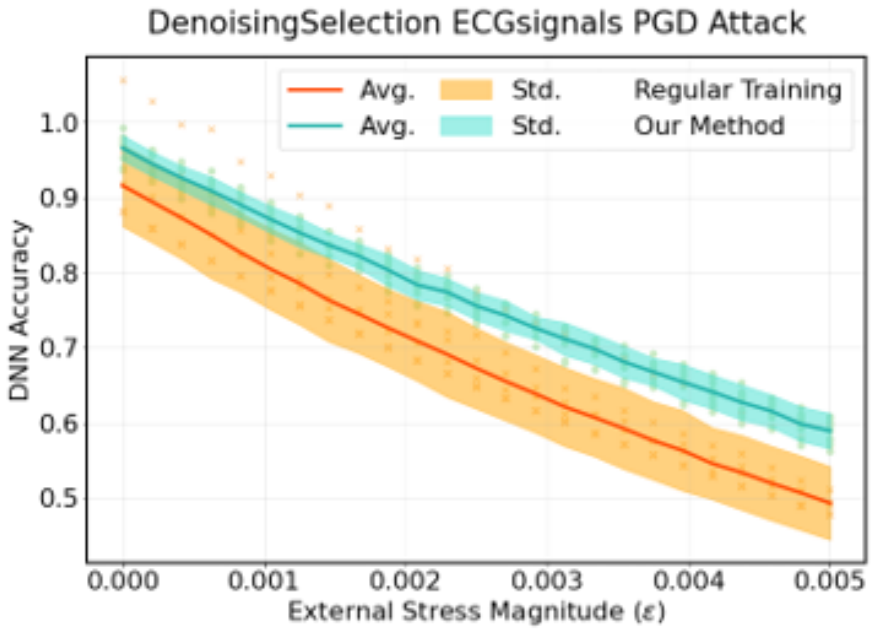


Fig. 5.7 Performances of the proposed ECG signal denoising filter selection network, trained using a regular training procedure and the selective backpropagation method, as presented in Chapter 4. Both the regularly trained network (shown in orange) and the network trained using selective backpropagation (shown in teal) were tested on the PGD attack with varying perturbation magnitudes to evaluate the effects of the proposed training. Showing the average (labelled 'Avg.' and shown as a solid line) and standard deviation (labelled 'Std.' and shown by the shaded coloured regions) of the results all trained networks evaluated.

Such a high classification accuracy for determining the optimum denoising filter enables us to effectively remove noise from a signal without affecting the underlying signal characteristics. We show that when presented with different windowed signals, the optimum filter can be either the elliptical filter (shown in Figure 5.3) or the wavelet filter (shown in Figure 5.4).

The contents of this section were directed at firstly developing a new dataset that consisted of an existing dataset [190], corrupted with additive Gaussian noise to represent noisy data acquired using UWB radar systems. Using the known additive noise, we designed an algorithm to identify the optimum signal filter for denoising the Gaussian perturbations and subsequently trained a CNN model to be able to identify the optimum denoising method, given a noisy raw waveform. The findings of this section are used to preprocess the UWB radar signals used in Section 5.2. We subsequently apply the selective backpropagation methodology derived from the works presented in Chapter 3 and Chapter 4, to the denoising filter selection CNN model and observe an increase to the adversarial robustness of the network, as well as the regular accuracy of the network, increasing from 92.8% to 96.4%. We show that applying selective backpropagation successfully increases both the adversarial

robustness and regular test accuracy of the network. We apply the selectively backpropagated denoising filter selection network developed in this section, to the task of activity signal denoising in the following Section 5.2.

5.2 Activity Classification Application

In this section, we present a method of classifying human activities and activity intensities using an UWB radar system and deep learning. We conduct a custom experimental procedure to collate a novel dataset of human activities captured using the radar system. Furthermore, we design a new DNN architecture to classify human activities and intensities from radar signals. The task of automated activity classification has previously attracted various avenues of research, and has inspired different methodologies in solving the problem. We outline an unobtrusive method of detecting and classifying different activities and exercises using a 24 GHz UWB radar transceiver and a DNN. The radar transceiver module is used to record the data of a single individual carrying out 6 different activities within a closed environment, and the subsequently processed radar signals are used to train a CNN, which is used to classify the human activities and the intensity of the activities. We define the methods used to record the activity data using the radar transceiver, and the techniques used to process the raw radar signals in this section using the denoising filter selection method proposed in Section 5.1.

To add, we detail the proposed CNN model, including training regime and parameters used to learn the custom built dataset. We observe that the developed CNN model achieves a test accuracy of 92% for 3 classes of activities (squat, star jump and standing still), and a test accuracy of 74.2% on unseen data for classifying 6 different activities. A classification accuracy of 80.3% for classifying the intensity of the particular activity. Furthermore, we evaluate the developed activity and activity intensity classification networks to adversarial attacks, and using the selective backpropagation method detailed in Chapter 3 and Chapter 4, we improve the adversarial robustness of the developed networks. For the activity classification task, the selective backpropagation methodology is also able to imcrease the test accuracy on the clean, unperturbed, dataset. We summarize the complete system and show that deep learning can be used in conjunction with radar systems to classify human activities for application as an assisted exercise tool.

5.2.1 Overview of Activity Classification

Significant developments have been made in technologies relating to home assistants and artificial intelligence enabled systems for the home [206]. Home assistants are now trained to carry out complex tasks such as speech-to-text translation [207], searching the internet for useful information [206, 208] and automation of other household devices connected to a network [209]. With the ever growing demand for home assistants collecting data to make

complex decisions, there exists a real security concern regarding collection, storage and analysis of data recorded using automated systems. Within the scope of this project, we focus our attention to the task of an automated home gym assistant capable of classifying various exercises and differentiating between different intensities of exercise, such that it can inform the user about their performance to the exercises. We investigate the use of an unobtrusive 24 GHz radar system in conjunction with a deep neural network (DNN) for the classification of physical activities and exercises. The DNN model we focus on in our study particularly, is a convolutional neural network (CNN).

Global and national guidelines on physical activity constitute as one of the primary components in a comprehensive framework for public health management and action [210]. The World Health Organisation outline general guidelines for physical activity uptake and promote regular physical activity amongst the general population. The lack of regular physical exercise has wide reaching impacts on the greater society, as regular exercise is considered as a mitigating factor of various diseases and illnesses [211]. Oftentimes, it is apparent that guidelines alone are not sufficient in increasing the uptake of regular physical exercise amongst the general population [212], and further communication is required in order to encourage regular physical activity. From the perspective of public health, research suggests that activity tracking devices provide a cost-effective method of increasing physical activity motivation. We propose a system to utilise automated technology, in the form of a deep learning agent, to recognise different physical activities and exercises whilst also suggesting recommendations of improving or maintaining the activity intensity and movement.

There exists various technologies providing methods of physical activity and movement tracking, many of which using wearable devices equipped with accelerometers, vibration sensors and visual sensors [213]. One of the major concerns with on-person devices capable of regular activity and exercise monitoring are the privacy vulnerabilities of wearable technologies [214, 213]. The system proposed within the context of this study is capable of circumventing the issue of data security by utilising the unobtrusive 2.4 GHz radar system for data measurement. The main contributions and innovation points of this work are as follow:

- We collect an *original* dataset consisting of radar measurements recording basic activities such as squatting, star jumps, walking and standing. The collected dataset also includes three different levels of exercise intensity relating to the range of motion for each exercises, from light motion, medium motion and full motion.

- We develop a novel CNN based methodology, outlined in Section 5.2.4 and train the model to classify different activities using radar signals. We validate the CNN model on unseen data and show that automated classification of different activities is possible.
- We extend the CNN model to classify three levels of motion intensity for each exercise and present our findings in Section 4.5.
- We evaluate and increase the adversarial robustness of the developed network using selective backpropagation, as detailed in Chapter 3 and Chapter 4, which also increases the test accuracy of the network on un-perturbed data.

5.2.2 Related Work

The following section presents state-of-the-art work with regards to using deep learning in conjunction with radar systems to classify/predict various properties from the radar measurements. The current commercially available state-of-the-art devices for physical activity classification and monitoring can be broadly categorised within the domains of (i) wearable technologies [215, 213] that are equipped with an array of sensors to measure various motions, and (ii) visual systems that use video data to classify specific activities. Wearable and visual technologies, however, pose concerns regarding data security, as the data collected may consist of sensitive information about the users and environment [214]. We circumvent this security concern by using unobtrusive radar system as the method of acquiring data.

There exists various studies and models proposed to applying deep learning for the task of physical exercises recognition [213, 216, 40]. It should be noted that the majority of methods proposed for the task of activity recognition, use forms of wearbale technologies to acquire data. Ravi et al. [217] also suggest a deep learning approach using on-node sensor data analytics for classifying activities using wearable devices. The primary direction of the work presented in this section relates to the task of human activity classification using radar systems. We direct the reader to the works of [72] which details an overview of deep learning for human activity classification using radar systems. To add, in their work Gurbuz and Amin [141] detail numerous different applications of deep learning for radar-based human-motion recognition. There has also been research conducted into using deep learning enabled radar systems for safety critical activity classification amongst elderly people [142, 145]. The inherently noisy and random nature of the data acquired using radar systems also pose questions on the robustness of developed learning models, such as not to misclassify, or fail to classify safety-critical activities.

The task of exercise recognition using DNNs poses distinct problem specifications, particularly when considering repetitions, exercise intensities and response to unpredictable noise from the environment. In their work, Soro et al. [218] present an end-to-end deep learning approach to classifying complex physical exercises using data recorded from wearable technologies. The primary divergence between the the works of Soro et al. [218] and work and the work proposed in this research is the type of data used and the method of recording the data. We aim to utilise the ability of radar systems to recognise minute and often complex motions, similar applications can be found in the works of [145, 142, 141], where they outline a method to recognise and classify different exercises.

5.2.3 Signal Processing and Experimental Set-Up

In the following section we outline the methods used to process the raw noisy radar signals, such that we extract important features from the measurements and prepare the signal to be applied to the CNN model detailed in Section 5.2.4. We also outline the experimental set-up used to acquire the radar measurements, detailing the procedures designed for acquiring radar signatures of exercises from individuals.

Signal Processing

We begin by considering the noisy radar signal recorded for an specific activity. The raw signal contains unwanted features (noise) that we foremost remove using a 2^{nd}-order low-pass ChebyShev Type II filter with a critical frequency of 1000 Hz and 3 dB ripple to remove any high frequency components that do not represent regular activities, as mentioned in the works of Jokanovic et al. [145]. This can be seen in Figure 5.16. Due to the nature of the recordings, splitting the signal into equal individual repetitions from the full signal cannot be done efficiently due to the overlapping nature of each activity repetition signal, and noise contained within the signal. To segment the full signal into 5 repetitions we must manually identify the discrete time points at which the signal can be split. To aid this process, the signal is first transformed into frequency-time using mel-spectrogram [68].

As can be seen from Figure 5.8 (bottom), the time-series signal is noise, even post-filtering and as such, definitive repetition breaks cannot be easily identified. When the signal is transformed into a mel-spectrogram, as shown in Figure 5.8 (Top), identification of individual repetitions, as well as specific motions of the repetitions, can be identified with less effort. From the signal spectrogram, we can see the frequency spectrum of the signal with respect to time and measure the points in time where the signal shows to contain

lower average frequency components, indicating a still motion or relative lack of activity. We separate each repetition from the full signal containing 5 repetitions using the signal spectrogram. When recording the activity measurements, the motion routine is noted and referenced when carrying out the signal segmentation procedure. Upon windowing the

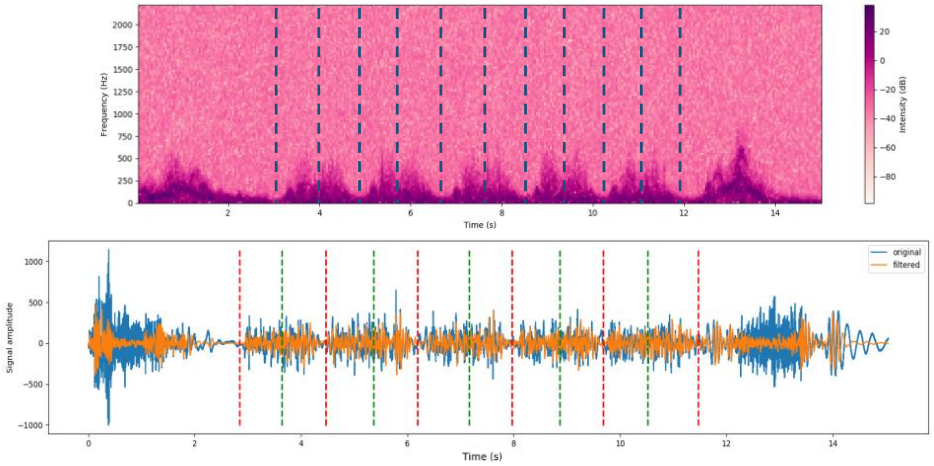


Fig. 5.8 (**Top**) figure shows the radar signal transformed into a spectrogram, such that we can analyse the signal over time at various frequencies and identify each repetition of an exercise. (**Bottom**) figure shows how the noisy, full signal is split into its constituent repetitions using the spectrogram.

full signal into 5 repetitions, the window lengths are not expected to be the equal for all repetitions. This is due to the fact that certain repetitions may be longer or shorter in duration, and thus different repetitions are expected to yield varying lengths of windowed signals. In order to standardise the windowed dataset, we take the length of all signals to be of length N, where N is the maximum length from all individual windowed signals.

Once the radar signals have been segmented and standardised to have equal length, the dataset is then manually filtered using a Chebyshev Type II bandpass filter, and a discrete wavelet filter to identify the optimum filtering method that results in the highest SNR, indicating a more effective removal of noise. The calculation of SNR is carried out, similar to that detailed in Equation 5.14, is given as $20\log_{10}(z/(z-g))$, where z is the noisy signal and g is the background noise. The noise in question is the measured from the environment during the data collection process with no participants in the field of view of the radar module. We use the procedures outlined in Section 5.1, which is based on Reference [42], to carry out the processes of effective signal denoising.

The network presented in Section 5.1.3 is used to identify the optimum filtering method for a given noisy windowed waveform. It can be seen from Figure 5.9 and Figure 5.10 that two different waveforms, x^a and x^b, result in different levels of filtering using a Chebyshev Type II band-pass filter and a wavelet filter, using the same filtering parameters for both operations. The filtered waveforms r_0^a and r_1^a in Figure 5.9, are the result of waveform z^a

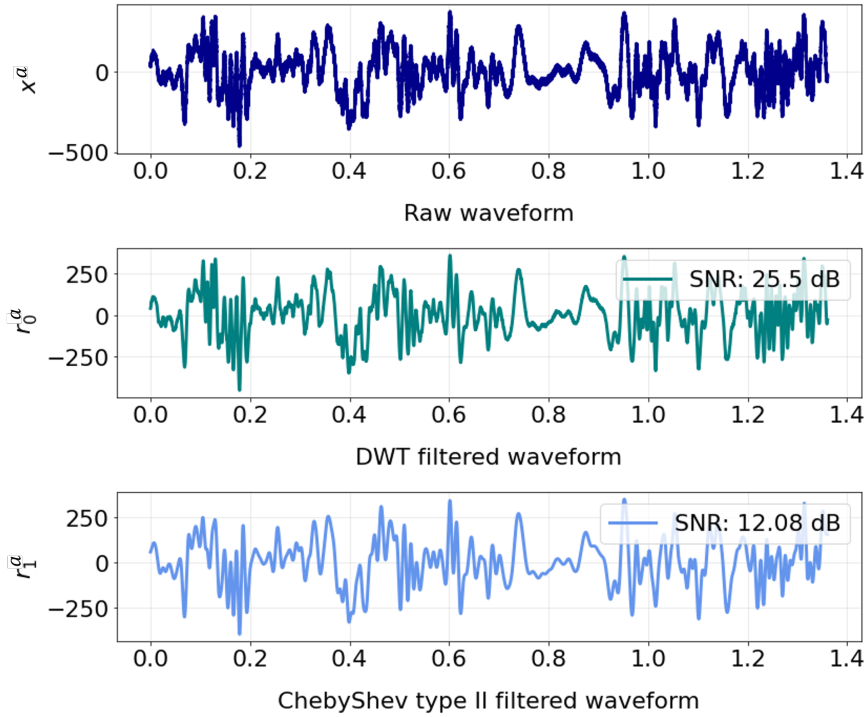


Fig. 5.9 Filtering example of waveform x^a using a wavelet filter, the resultant waveform for which is labelled r_0^a, and a 5-th order Chebyshev Type II band-pass filter, the resultant waveform for which is labelled as r_1^a.

filtered using a Chebyshev Type II band-pass filter and a wavelet filter respectively. The filtered waveforms r_0^b and r_1^b in Figure 5.10, are the result of waveform z^b filtered using a Chebyshev Type II band-pass filter and wavelet filter respectively. It can be seen from the Figure 5.9, that the optimum SNR value of 25.5 dB calculated using the raw waveform x^a was observed to result from the DWT filtered signal r_0^a, compared to the Chebyshev type II filtered signal, which obtained an SNR value of 12.08 dB. Whereas for the raw waveform x^b, the optimum calculated SNR value of 19.37 dB was calculated for the Chebyshev Type II filtered waveform r_1^b, compared to an SNR value of 8.59 dB for the DWT filtered waveform. It can be seen from the presented examples that an optimum filter selection has the potential of significantly improving the signal quality. We employ the CNN-based denoising filter selection algorithm to automate the task of identifying an optimum pre-processing filter for

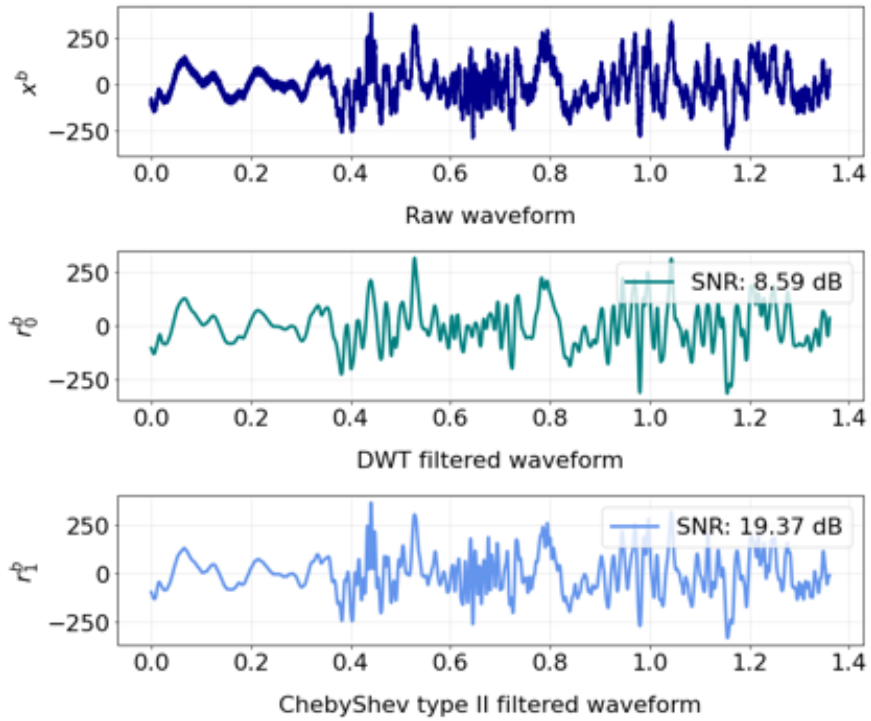


Fig. 5.10 Filtering example of waveform x^b using a wavelet filter, the resultant waveform for which is labelled r_0^b, and a 5-th order Chebyshev Type II band-pass filter, the resultant waveform for which is labelled as r_1^b.

signal removing noise. We achieve this objective through training the network on noisy waveforms labeled with the optimum denoising filter that are identified using a manual denoising procedure of the waveforms.

From the example waveforms shown Figure 5.10 and Figure 5.9, we can observe that selecting an effective filtering method results in filtered signal with higher SNR values. In Figure 5.9 we see that the noisy waveform x^a The CNN model outlined in Section 5.1 is applied to the task of removing noise from human activity data recorded using an UWB radar system. As shown in Figure 5.11, classification accuracy of 89% is achieved for predicting the optimum denoising filter, between a Chebyshev Type II and wavelet filter. The effects of selecting the optimum filter for a given noisy waveform.

5.2.4 Activity and Intensity Classification

Within this section we outline the proposed activity and intensity classification framework, and the accompanying novel CNN architecture and dataset processing algorithms for exercise recognition and intensity classification. We also outline the signal preprocessing techniques

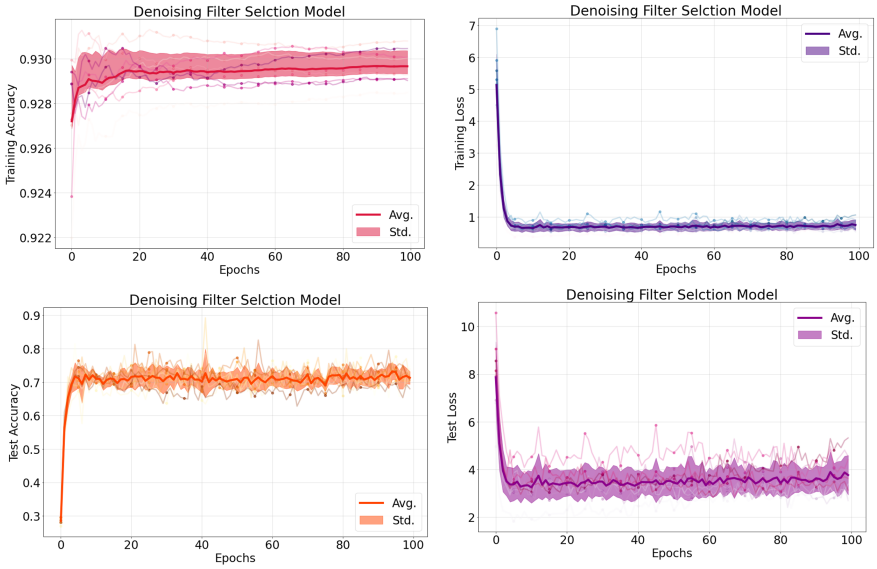


Fig. 5.11 Activity classification signal denoising filter selection training results.

used to to prepare the raw radar data for the subsequent classification models, as each model requires different data transformations to be applied to the waveforms for effective classification, prior to application. The activities measured were: squats, star jumps, wall push-ups, sitting down on a chair, standing upright, and walking across the field of measurement.

Activity and Intensity Datasets

From the measured activities, each exercise signal set contained within it, 5 individual repetitions of the exercise. For the walking, sitting, and standing activities a continuous signal of the participant carrying out the activities was recorded, without specified repetitions. Each repetition signal is further processed prior to the classification task. We are left with a resultant of 100 signals for each exercise (squats, star jumps, wall push-ups, sitting still, walking and standing) and each person. The procedure for measuring activity and intensity data for each participant, is summarised in Table 5.3.

We take a signal x and apply a digital 4^{th}-order Chebyshev Type II bandpass filter to remove frequencies below 0.1 Hz, for the purpose of stabilising the signal, and 500 Hz, which from analysis of the signals, found to not contain significant activity features. The enclosed frequency bandwidth is then filtered using a 3^{th}-order Chebyshev Type I notch filter with critical frequencies at $[(15-30),(30-60),(60-120),(120-240),(240-280)]$, resulting in 5 filtered signals from one original repetition signal. The notch filter critical frequencies are decided upon through an analysis of a Fourier transformed log-magnitude spectrum of a

Table 5.3 Experiment set up for activity intensity recording

Activity	Experiment
Squats	5 repetitions of 3 sets
Star jumps	5 repetitions of 3 sets
Wall push ups	5 repetitions of 3 sets
Sitting on a chair	continuous signal of 3 sets
Standing for a period of 10 seconds	continuous signal of 3 sets
Walking across the field of measurement	continuous signal of 3 sets
Low intensity star jumps	5 repetitions of 3 sets
Medium intensity star jumps	5 repetitions of 3 sets
High intensity star jumps	5 repetitions of 3 sets

sample dataset. The purpose of the notch filter windowing on the raw signal is to carry out a transformation of the raw signal through removing prominent frequencies, such that we may better understand the nature of motions and their accompanying frequencies.

Activity and Intensity Classification Network Architectures

The DNN was configured with the following parameters in sequential order: a 32 dense layers, a maximum pooling layer with pooling size of 2, a 64 dense layers, a 32 dense layers and a 16 dense layers, all using ReLu activation functions. A unit dropout rate of 25% was used after the 128 dense layer and 64 dense layer, followed by a 50% rate after the flattened 16 dense layer, this was applied in order to avoid overfitting. An Adam optimizer [204] using backpropagation was used for the learning method and the model was trained for 20 epochs. The motions of activities are presented in the example Figure 5.12

Presented in Figure 5.13 is the network architecture of the activity classification network with an output of 6 classes representing the different activities outlined in Table 5.3. Presented in Figure 5.14 is the network architecture of the intensity classification network with 3 different classes representing low, medium, and high intensity activities. It should be noted that the intensity classification network is designed for the squat and star jump activities only, with both activities exhibiting the 3 different intensities of the activity.

Experimental Set-Up

The radar system used in this study is the 24 GHz I/Q channel K-LC2 radar transceiver with the ST100 evaluation board to control system power, data transfer and radar tuning. The radar transceiver, and the accompanying development board used for and data transfer, were

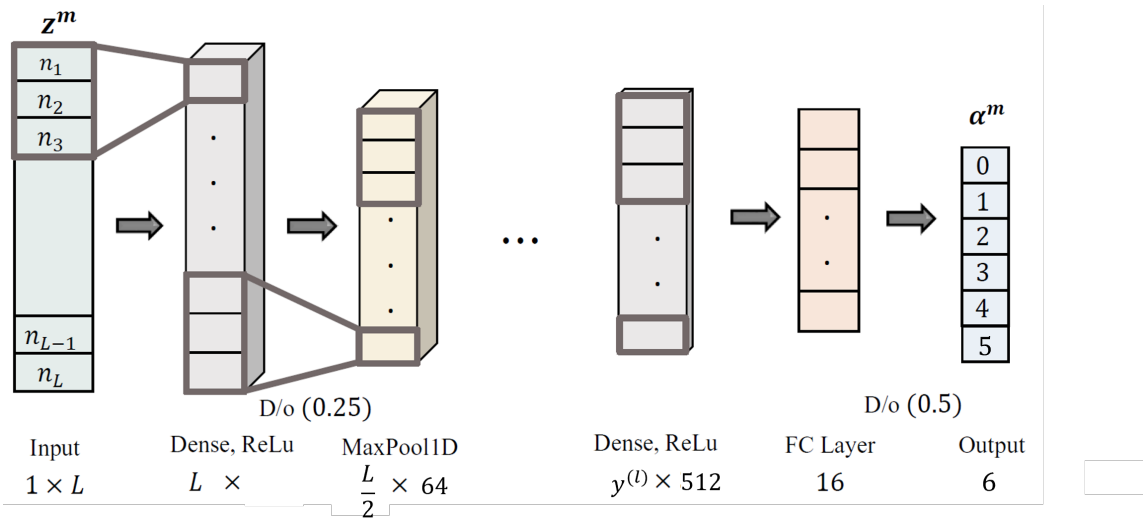


Fig. 5.12 Shown here is a condensed figure for the CNN model developed in this study for the classification of different exercises. The input of the model is controlled by the parameter L and represents the length of the signal. The output of the model is a vector of length 5 representing sit ups, squatting, star jumps, body twisting and standing.

powered using a 5V DC power supply. All radar recordings were carried out under laboratory conditions and the raw dataset consisted of seven individuals (3 males and 4 females).

Each full-length recording signal consists of 5 repetitions of each exercise and there are 3 recording trials per exercise measured. All recordings were carried out with the individual being recorded carrying out the 6 different activities with a distance of 2m away from he radar transceiver. From Figure 5.15 we show two example exercises and the signal segmentation method in which we manually classify the different motions that constitute each exercise. The resultant signal is segmentation of the original radar signal and represents one instance of a given activity.

5.2.5 Results and Analysis

In this section we begin by describing the resultant methodology from analysis of the 24 GHz radar signals for the purpose of human activity classification. The complete methodology for processing the radar signals can be found in Figure 5.16, containing initial signal preprocessing (**A**), the signal transformation through windowed filtering resulting in different signals from the original signal (**B**) and the notch filter windowing carried out to remove specific frequencies (**C**).

```
ActivityClassifier(
  (conv1): Sequential(
    (0): Conv1d(1, 64, kernel_size=(3,), stride=(3,), padding=(1,))
    (1): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
  )
  (conv2): Sequential(
    (0): Conv1d(64, 64, kernel_size=(3,), stride=(1,), padding=(1,))
    (1): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool1d(kernel_size=3, stride=3, padding=0, dilation=1, ceil_mode=False)
  )
  (conv3): Sequential(
    (0): Conv1d(64, 64, kernel_size=(3,), stride=(1,), padding=(1,))
    (1): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool1d(kernel_size=3, stride=3, padding=0, dilation=1, ceil_mode=False)
  )
  (conv4): Sequential(
    (0): Conv1d(64, 64, kernel_size=(3,), stride=(1,), padding=(1,))
    (1): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool1d(kernel_size=3, stride=3, padding=0, dilation=1, ceil_mode=False)
  )
  (conv5): Sequential(
    (0): Conv1d(64, 128, kernel_size=(3,), stride=(1,), padding=(1,))
    (1): BatchNorm1d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool1d(kernel_size=3, stride=3, padding=0, dilation=1, ceil_mode=False)
  )
  (conv6): Sequential(
    (0): Conv1d(128, 128, kernel_size=(3,), stride=(1,), padding=(1,))
    (1): BatchNorm1d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool1d(kernel_size=3, stride=3, padding=0, dilation=1, ceil_mode=False)
  )
  (conv7): Sequential(
    (0): Conv1d(128, 128, kernel_size=(3,), stride=(1,), padding=(1,))
    (1): BatchNorm1d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool1d(kernel_size=3, stride=3, padding=0, dilation=1, ceil_mode=False)
  )
  (conv8): Sequential(
    (0): Conv1d(128, 128, kernel_size=(3,), stride=(1,), padding=(1,))
    (1): BatchNorm1d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool1d(kernel_size=3, stride=3, padding=0, dilation=1, ceil_mode=False)
  )
  (conv9): Sequential(
    (0): Conv1d(128, 256, kernel_size=(3,), stride=(1,), padding=(1,))
    (1): BatchNorm1d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool1d(kernel_size=3, stride=3, padding=0, dilation=1, ceil_mode=False)
  )
  (conv10): Sequential(
    (0): Conv1d(256, 256, kernel_size=(3,), stride=(1,), padding=(1,))
    (1): BatchNorm1d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
  )
  (conv11): Sequential(
    (0): Conv1d(256, 512, kernel_size=(3,), stride=(1,), padding=(1,))
    (1): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
  )
  (conv12): Sequential(
    (0): Conv1d(512, 512, kernel_size=(3,), stride=(1,), padding=(1,))
    (1): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
  )
  (fc): Linear(in_features=3072, out_features=6, bias=True)
)
```

Fig. 5.13 Activity classification network architecture designed to classify between 6 different activities: squats, star jumps, wall push ups, sitting, standing and walking.

```
IntensityClassifier(
  (conv1): Conv2d(3, 12, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (relu1): ReLU()
  (bn1): BatchNorm2d(12, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (do1): Dropout(p=0.25, inplace=False)
  (conv1_1): Conv2d(12, 12, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (bn1_1): BatchNorm2d(12, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relu1_1): ReLU()
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv2): Conv2d(12, 20, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (bn2): BatchNorm2d(20, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relu2): ReLU()
  (conv2_2): Conv2d(20, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (bn2_2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relu2_2): ReLU()
  (conv3): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (bn3): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relu3): ReLU()
  (conv4): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (bn4): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relu4): ReLU()
  (conv5): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (bn5): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relu5): ReLU()
  (fc): Linear(in_features=261120, out_features=3, bias=True)
)
```

Fig. 5.14 Intensity classification network architectures designed to recognise between the different intensities of 6 activities, from low, to medium, and high intensity of activities.

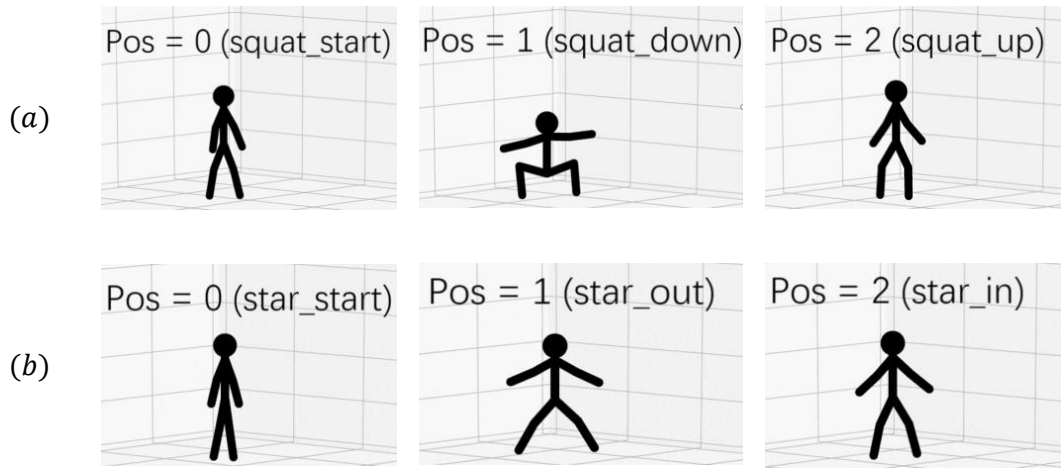


Fig. 5.15 (**a**) shows the expected motions when recording for a squat exercise with accompanying labels for the three motions considered to make up the squat exercise. (**b**) Shows the expected motions when recording for a star jump exercise with accompanying labels for the three motions considered to make up the squat exercise.

Classification Results

The model detailed in Section 5.2 results in a classification accuracy of 92% when tested against 3 classes (squats, star jumps and standing still). The network accuracy for the six

activity classes of squats, star jumps, wall push-ups, sitting still, standing still and walking reduces to 74.2%. For motion intensity classification we consider a signal activity and measure three different levels of intensity; high intensity, mid intensity and low intensity. The custom recorded dataset consists of 300 segmented signals (100 signals of each intensity range) which are the result of the moving notch filter transformation, as described in Section 5.2.4. The dataset is split into a test set (30% of total signal) and train set (70% of total signal), furthermore the base signals for standing still are also added into the dataset, resulting in 135 test signals and 255 training examples. The intensity classification task using the CNN model detailed in Section 5.2.4) results in a classification accuracy of 84% over a test set of 135 signals.

Filter Bank

From Figure 5.16 part **A** we have (1.) two exercises a (squat) and b (star jump) which are both filtered using a (2.) low pass filter with a critical frequency of 1000 Hz, as detailed in Section 5.2.3. The resultant signals (3.) of a and b are filtered for high frequency components. Following this, part **B** shows the two signals a and b transformed using $T^(f,t)$ into the signal frequency f and time t, resulting in a spectrogram. We use the spectrogram (1.) to segment the signal in time to extract 5 repetitions from the original signal. An example of a segmented signal (2.) is shown with frequency components. This results in (3.) two signals which have the same motion label as the original, but contain different information. We utilise this behaviour to form **C**, where we take an input signal x (1.) and systematically remove specific frequency bands (2.,3.) and form the final dataset (4.,5.), with only specific frequency components remaining (6.).

Upon retrieving the windowed waveforms, we filter each waveform using a Chebyshev Type II filter and a wavelet filter to remove background noise of the same frequencies. The selection of the filters, between a wavelet filter and a Chebychev type II filter, is chosen using the signal denoising filter selection method presented in Section 5.1 and further detailed in Section 5.2.3. Using this method we are able to filter all waveforms using the optimal window filtering method, as discussed in Section 5.2.3.

5.2.6 Selective Backpropagation for Radar Signal Classification

Here we present the parameter scores and adversarial robustness of the developed activity and intensity classification networks. This is achieved through the application of the robustness improvement methods, using the three synaptic filters h_1, h_2, and h_3 outlined in Chapter 4,

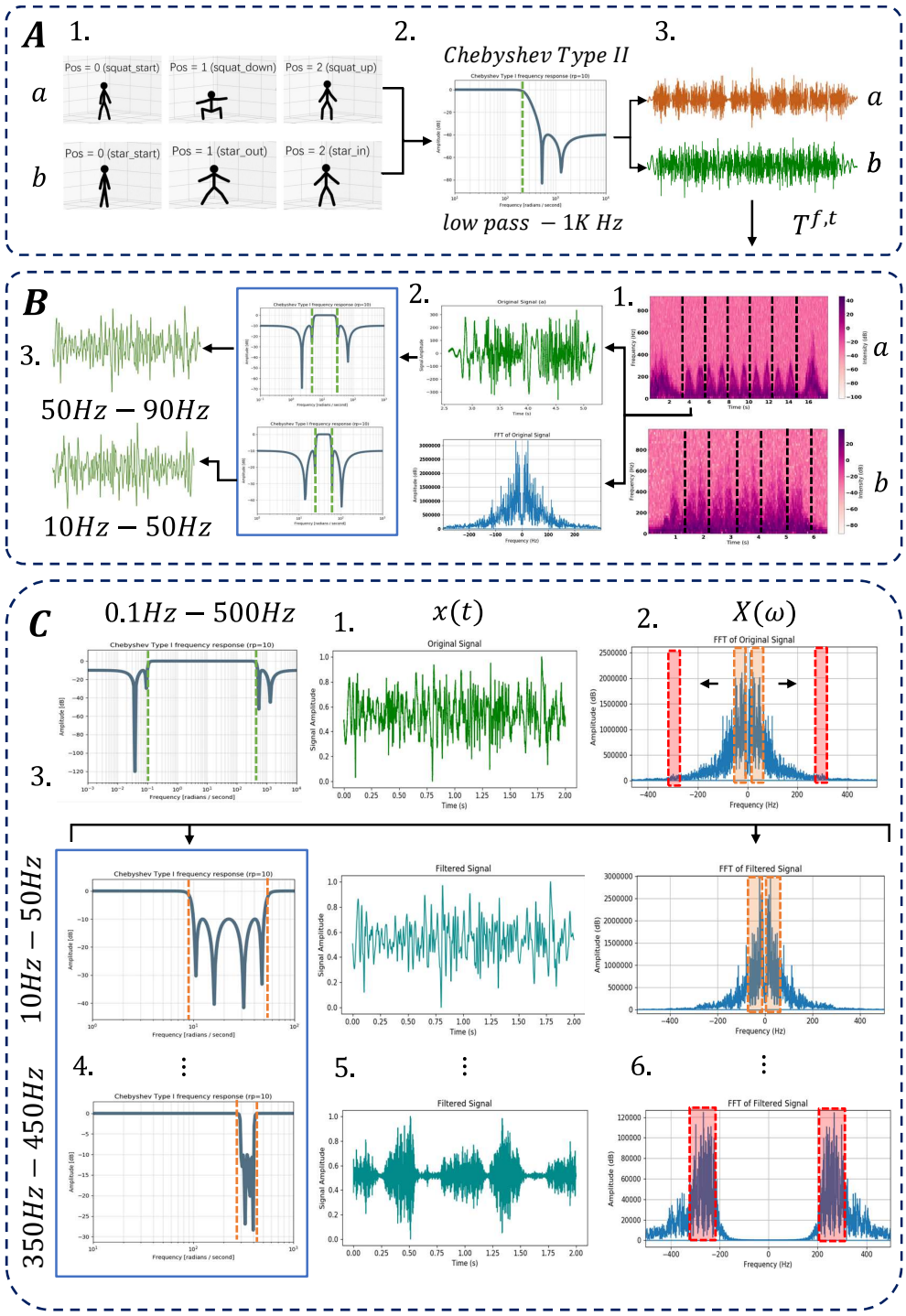


Fig. 5.16 Diagram showing data prepossessing **A**, the result of isolating specific frequencies **B** and the moving notch filter **C** used to extract different features form the original signal. Further details on the diagram can be found in Section 5.2.5.

that use selective backpropagation to retain only the robust and antifragile parameter update gradients. It should be noted that the identification of of fragile, robust and antifrgaile network parameters, which is a prerequisite of applying the selective backpropagation method, is carried out using on the FGSM attack only. We test the effectiveness of the selective backpropagation method on the fast gradient sign method attack and the projected gradient descent attack (PGD) [135]. The PGD attack is considered to be a universal first-order adversarial attack and stronger than the FGSM attack, due to the increased steps taken to increase loss. We test the trained networks on the PGD attack to evaluate the ability of the network to generalise on other adversarial datasets. The results presented in this section are averaged over 10 different trained and initialised networks.

Activity Classification The local layer-wise parameter scores for the custom-designed activity intensity classification DNN are shown in Figure 5.20. It can be noticed form the clean dataset parameter scores (r_x), that the parameters of the network with less than 50 epochs training show to be more fragile overall, compared the parameters of networks that were trained for longer. This relative fragility of parameters at earlier stages in training can be observed at all layers of the network. Similarly, the adversarial parameter scores (r_{x_ε}) show a concentration of more fragile parameters at earlier stages in network training. From the adversarially targeted parameter score difference (r_ε), we can see that the most fragile parameters are located in the middle layers of the network. We use the information from Figure 5.17 to identify fragile, robust and antifragile network parameters, such that the selective backpropagation procedure only propagated robust and antifragile parameter weight updates. It should be noted that the activity classification network is designed for one-dimensional inputs and thus the convolutional layers are one-dimensional in width.

Upon identification of fragile, robust and antifragile parameters we apply the proposed selective backpropagation training procedure in parallel to regular training 10 epoch intervals in the classification model training procedure. The results of the selective backpropagation method is displayed in Figure 5.18 for the network under an FGSM attack, and Figure 5.19 for the network under a PGD attack. We can observe the activity classification network improves in performance against both the FGSM attack and PGD attack as the external stress (perturbation magnitude ε) is increased.

Furthermore, for the FGSM attack, we also observe an increase in network performance for the clean dataset ($\varepsilon = 0$). The increase in performance on the clean dataset is also observed on the PGD attack for the activity classification model. This suggests that the robust and antifragile network parameters relate to both the adversarial robustness, as well as random

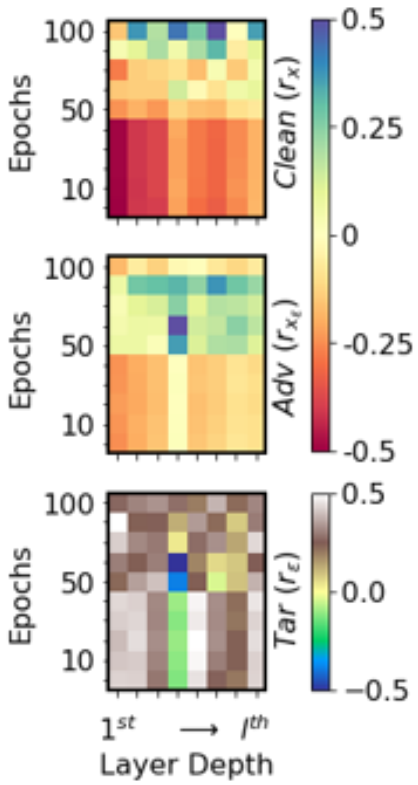


Fig. 5.17 Local layer-wise parameter scores for the activity classification model. Results measured periodically every 10 epochs from epoch 10 to epoch 100. Presented are the parameter scores r_x (clean dataset), r_{x_ε} (adversarial dataset), and r_ε (scaled difference in clean and adversarial dataset scores). The layer depth represents the layers of the network ranging from the 1^{st} convolutional layer to the l^{th} fully-connected layer.

noise artefacts that may be found in the 'clean' dataset that is hindering the performance of the network to reach optimal accuracy. The results in Figure 5.18 and Figure 5.19 show that the proposed analysis to identify fragile, robust, and antifragile parameters, along with the subsequent selective backpropagation method, is capable of improving both the regular clean dataset and the adversarial dataset.

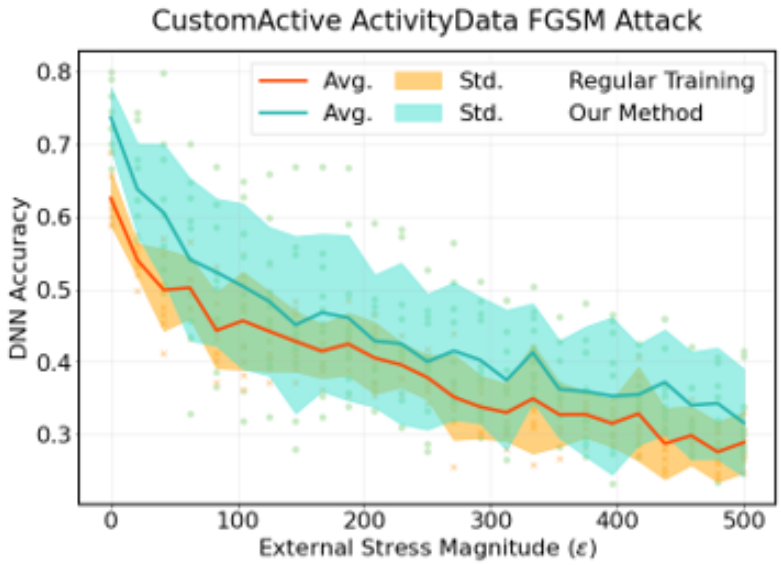


Fig. 5.18 Performances of the activity classification DNN, trained using a regular training procedure and the selective backpropagation method. Both the regularly trained network (shown in orange) and the network trained using selective backpropagation (shown in teal) were tested on the FGSM attack with varying perturbation magnitudes. Showing the average (labelled 'Avg.' and shown as a solid line) and standard deviation (labelled 'Std.' and shown by the shaded coloured regions) of the results for all networks evaluated.

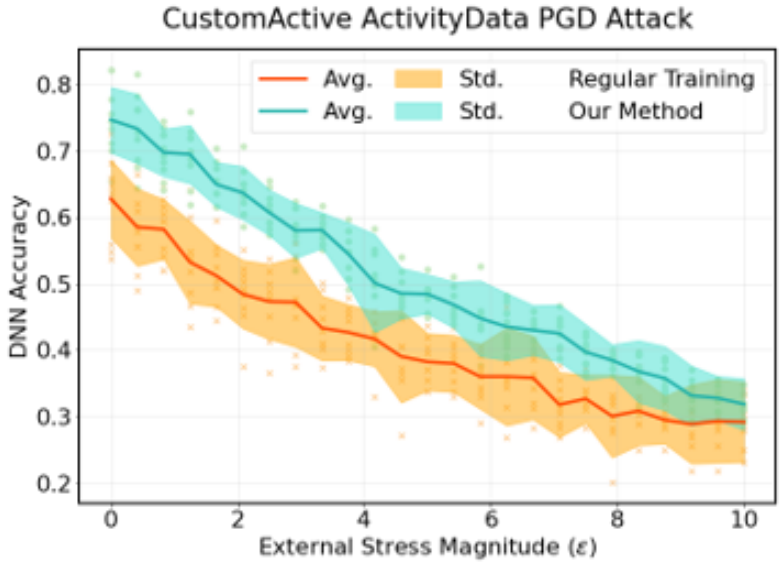


Fig. 5.19 Performances of the activity classification DNN, trained using a regular training procedure and the selective backpropagation method. Both the regularly trained network (shown in orange) and the network trained using selective backpropagation (shown in teal) were tested on the PGD attack with varying perturbation magnitudes. Showing the average (labelled 'Avg.' and shown as a solid line) and standard deviation (labelled 'Std.' and shown by the shaded coloured regions) of the results for all networks evaluated.

Intensity Classification The local layer-wise parameter scores for the custom-designed activity classification DNN are shown in Figure 5.17. It can be noticed that as the layer depth increases from the first convolutional layer to the last fully-connected layer, the parameter scores indicate that the layers situated centrally to the network, are identified as more fragile compared to the first and last layers of the network, when considering only the clean dataset (r_x). This fragility in the middle section of the network can be observed throughout network training process, as shown for all epochs from epoch 10 to epoch 100. Conversely, the adversarial parameter scores (r_{x_ε}) do not show a defined concentration of fragile parameters in the central region of the network layers as the distribution of fragile parameters shows to vary over epochs and layer depth.

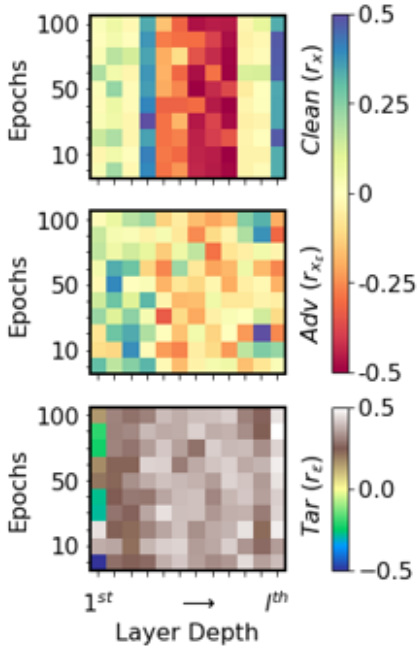


Fig. 5.20 Local layer-wise parameter scores for the intensity classification model. Results measured periodically every 10 epochs from epoch 10 to epoch 100. Presented are the parameter scores r_x for the clean dataset, r_{x_ε} for the adversarial dataset, and r_ε for the scaled difference in clean and adversarial performances. The layer depth represents the layers of the network ranging from the 1^{st} convolutional layer to the l^{th} fully-connected layer.

The adversarially targeted difference in parameter scores (r_ε) show that the most fragile parameters are situated at the beginning of the network, the most prominent amongst which, is the first convolutional layer (labelled 1^{st}). Using the parameter scores r_ε we categorise the layers are fragile ($r_\varepsilon < -\varepsilon$), or robust, and antifragile ($r_\varepsilon >= -\varepsilon$). The value of ε was chosen to to be equal to 0, defining a strict robustness where clean and adversarial performances must be exactly alike for the parameter to be characterised as robust.

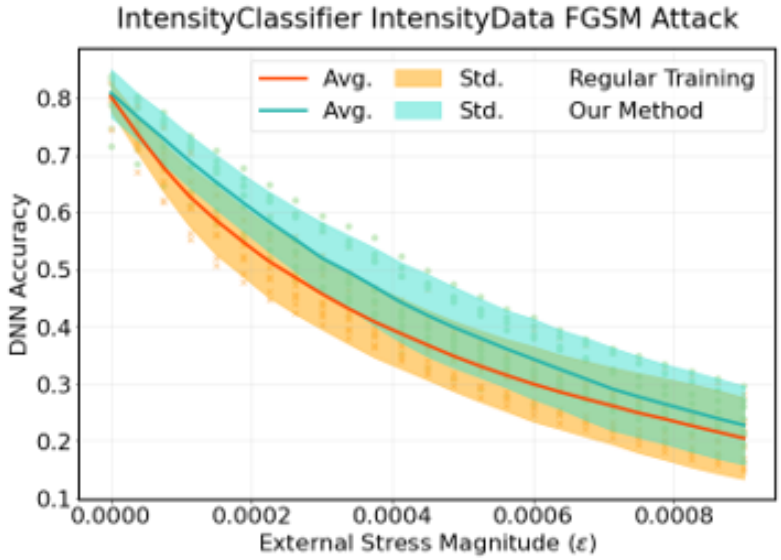


Fig. 5.21 Performances of the proposed intensity classification network, trained using a regular training procedure and the selective backpropagation method, as presented in Chapter 4. Both the regularly trained network (shown in orange) and the network trained using selective backpropagation (shown in teal) were tested on the FGSM attack with varying perturbation magnitudes to evaluate the effects of the proposed training. Showing the average (labelled 'Avg.' and shown as a solid line) and standard deviation (labelled 'Std.' and shown by the shaded coloured regions) of the results all trained networks evaluated.

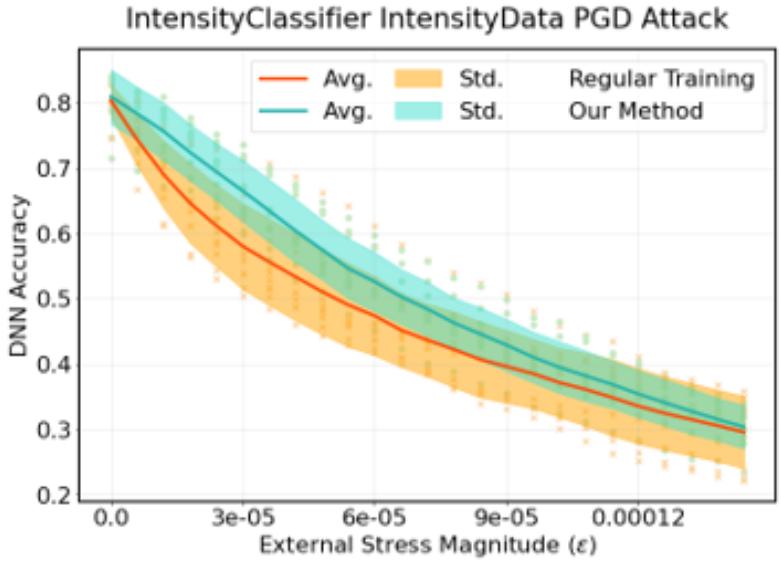


Fig. 5.22 Performances of the proposed intensity classification network, trained using a regular training procedure and the selective backpropagation method, as presented in Chapter 4. Both the regularly trained network (shown in orange) and the network trained using selective backpropagation (shown in teal) were tested on the PGD attack with varying perturbation magnitudes to evaluate the effects of the proposed training. Showing the average (labelled 'Avg.' and shown as a solid line) and standard deviation (labelled 'Std.' and shown by the shaded coloured regions) of the results all trained networks evaluated.

With the identification of fragile, robust, and antifrgaile network parameters we can subsequently carry out the selective backpropagation procedure to retrain only the robust and antifragile parameters. Similarly to the activity classification analysis shown previously, we test the selectively backpropagated, and regularly trained intensity classification networks to the FGSM attack and the PGD attack. The results for which are presented in Figure 5.21 for the FGSM attack and Figure 5.22 for the PGD attack. We observe that the average intensity classification network performance increases in performance to both the FGSM attack and PGD attack as the external stress magnitude (perturbation magnitude ε) is increased. This shows that the selective backpropagation methodology is successful in increasing adversarial robustness of the network to two different methods of attack, whilst only the FGSM attack is considered during the identification process of fragile, robust, and antifragile parameters.

5.2.7 Activity Classification Summary

The work in this chapter investigates the use of 24 GHz radar system for the measurement and classification of home exercises using signal processing and deep learning. We record a custom dataset for this study consisting of 500 total signals, 25 repetitions of each exercise considered and collected from 2 individuals for activity classification and intensity classification. The three exercises considered in this study are: squats, star jumps, walking and standing upright. We also conduct a study to show how motion intensity (high intensity, medium intensity and low intensity) can be classified using the proposed CNN model.

In this work, we offer a method of processing the raw radar signals, such that important exercise features are identified and extracted efficiently using a sliding notch filter over an identified frequency range. The networks developed in this section and results acquired utilised the selective backpropagation methodology, as detailed in Chapter 3 and Chapter 4. We process the signal using a signal denoising filter selection method, as outlined in Section 5.1, that achieves an optimum filter classification with an accuracy of 82.4% for noisy ECG signals. We propose a 1D CNN model to classify the 3 different exercises that achieves a classification accuracy of 92% for the 3 different exercises (squat, star jump and standing still). Using the developed network, we classify 6 different activities (squat, star jump, wall push ups, sitting, standing and walking) with an accuracy of 72.2%. We also develop a 2D CNN network designed to classify the intensity of the activities recorded using a UWB radar system that is able to classify activity intensity between low, medium, and high. The 2D CNN network achieved a test accuracy of 80%. The developed networks were also test against the fast gradient sign method (FGSM) and projected gradient descent

(PGD) adversarial attacks to evaluate the robustness of the classification tasks and using the selective backpropagation methodology, we observed an increase in both adversarial robustness and the clean, unperturbed, datasets. We successfully improved the ability of networks to classify the objective functions and remain robust to adversarial attacks using the methodology developed throughout this thesis.

Chapter 6

Conclusions

In this thesis we have considered the robustness analysis of deep neural networks (DNNs) to different forms of input distortions and network parameter perturbations. We adopt the task of understanding how the architecture and the adversarial robustness of DNNs are related. Our investigations of network robustness have led us to approach the analysis from the perspective of various forms of stress applied to the overall system, input and network included. The types of stress we investigate can be broadly classified as internal and external to the network. With internal stress we refer particularly to perturbations of the trainable network parameters, in the convolutional and linear layer filter weights of DNNs. With external stress we refer particularly to distortions to network inputs, primarily of an adversarial nature. The two approaches of stress on DNNs allow us to evaluate the network with a greater level of scrutiny and also to highlight the specific internal components of networks that are most susceptible to external distortions to inputs. The proposed analysis goes into understanding the decision boundaries of DNNs and calling to attention the importance of network architecture to the robustness of networks.

The definition of robustness is considered to be a relative invariance when confronted with stress. We utilise the characterisations of fragility, robustness, and antifragility to define the conditions where the performance of a network is either impacted negatively, invariantly, or positively by the applied stress. We use the notions of fragility, robustness, and antifrgaility as central concepts when analysing the behaviour of deep neural networks, to the developed internal and external stress methods. We observe network fragility when the performance of the network degrades as stress is applied in increasing magnitudes. We can observe network robustness when the performance of a network remains constant, or bounded within a defined range, as stress of increasing magnitudes is applied to the network. Naturally, we must consider the condition where an increase in network performance is

observed, as increasing magnitudes of stress is applied. For such a circumstance, we use the notion of antifragility, as defined by Taleb and Douady [38], to characterise the condition where applied stress improves performance. In the thesis, we provide formal definitions of fragility, robustness, and antifragility within the context of DNNs and specific to the definitions of internal and external stress that we develop. The defined characterisations enable the grouping of particular network components dependent on the response of the network to the applied stress, be internal, external, or a combination of the two.

We have also discussed, in the context of classification networks, how adversarial robustness is closely related to network architectures. The work presented in this thesis offers novel insights into this relationship between adversarial robustness and network architectures. Through using adversarial attacks, we offer information on the importance of the composite components of architectures, and equally, through an analysis of network architectures we draw insights into the adversarial robustness of DNNs. The central methodology that we develop throughout this thesis is a general analysis of DNNs, which in practice, vary in shape and size. The method of applied stress is systematic and agnostic to different network architectures applied to tasks. This allows us to evaluate different networks on a common basis, and thus, compare the robustness of different network architectures working on differing datasets. The ability to compare the robustness of different networks also offers the opportunity to propagate only the most robust variations of networks when developing newer, stronger models.

Comparing network robustness is possible due to the systematic manner in which the defined internal and external stress is applied to the evaluated networks. Furthermore, the characterisations of fragility, robustness, and antifragility encapsulate the effects of different stress methods applied, thus allowing us to differentiate between networks with varying architecture arrangements. We bound the effects of an adversarial attack applied on a DNN to be equal for all networks, and achieve this by setting constraints on the adversarial perturbation magnitude. This consequently allows us to analyse the external robustness of networks within a confined range. The defined internal perturbation methods, in the synaptic filters, are also systematic in function and operate with a process that is the same for different network architectures. The synaptic filters target and remove the connective network weights of convolutional and linear parameters. The combination of using a relative adversarial attack constraint, along with a systematic filtering methodology, results in a comparable space upon which we can evaluate networks.

Applications of the Analysis The primary goal in analysing the robustness of DNNs is to further our understanding of how the networks respond to sub-optimal conditions and in the case of adversarial robustness, the worst-case conditions. We employ network parameter filtering as a way of enforcing sub-optimal conditions with respect to network architecture. The combination of internal and external stress, in network parameter filtering and adversarial attacks, allows us to observe the response of networks to sub-optimal conditions where both input and network are considered. We carry out such an evaluation to ascertain whether a trained network is able to generalise on unforeseen examples, as is case in real-world applications. We direct our work on identifying and reinforcing networks developed for real-world applications, where data exhibit forms of randomness that may be difficult to include in network training. Particularly, in this thesis we focused our analysis on various applications of DNNs for radar signal classification, as the task is highly sensitive to randomness from the environmental and measurement system alike. Such applications require networks to function under sub-optimal conditions and thus, the proposed analysis is used to drive the development of mode robust networks.

In further experiments, we apply the developed analysis methods on two applications using novel DNNs designed for custom radar signal datasets. The first task, we address, is a denoising filter selection algorithm based on a novel DNN architecture. We apply the analysis to identify the fragility, robustness, and antifragility of DNN components, and subsequently apply selective backpropagation to omit specifically the fragile components from the training process. In employing this method of network reinforcement, we observe an increase in both the test accuracy, as well as the adversarial robustness of the network. We further develop an activity identification DNN and an intensity classification DNN for a custom radar signal dataset, which is preprocessed using the aforementioned denoising filter selection algorithm. We again observe an increase in both the test accuracy and adversarial robustness for the network using the methodologies developed in this thesis. The results show that it is possible to selectively omit large parts of a network during the training process, leading a network that exhibits an increase in test performance. The implemented novel selective backpropagation methodology is successful in making DNNs perform better against both adversarial attacks and, in some instances, also the regular training dataset. The implications of the selective backpropagation is two fold; increase in performance, both adversarially and regularly, and requiring less computations compared to regular network training.

6.1 Future Work

The work presented in this thesis offers a novel methodology, to the best of our knowledge, of analysing DNNs under both adversarial settings and network parameter perturbations in conjunction. The analytical methods discussed in the preceding chapters have been extended to application on real-world implementations of DNNs, specifically for radar signal classification. The novel DNNs developed for the real-world implementation have been developed in collaboration with an industrial partner, for whom the custom recorded dataset and network architecture have been the subject of a patenting application. Furthermore, there has been work carried out in collaboration with undergraduate final year projects [68, 73], where the preliminary findings of this thesis have aided in the development of robust DNN applications.

A possible research direction of the work detailed in this thesis is to evaluate the proposed methodologies in greater extent to further benchmarks and real-world applications, such that the robustness of the procedures is evaluated in practice. Referring specifically to the work presented in Chapter 3, there are opportunities to evaluate the proposed nodal dropout method against various other forms of noise, such as different adversarial attacks and other forms of input distortions. This would constitute as a future direction of research from that presented in this thesis, as with the introduction of different forms of input distortions, we are confronted with selecting other relevant distortion methods, of which there are many. It is possible to use the work presented in Chapter 4 as the basis for studying the similarities and differences of different networks on a variety of benchmark tasks. The objective for such a study would be to highlight the fragility, robustness, and antifragilitiy of a variety of network architecture designs, thus, offering a systematic approach on designing new networks. The work developed in this thesis sets out a method of achieving these very objectives.

Additionally, we direct future work to address the robustness of more real-world applications of DNNs, so as to bridge the space between implementations of deep learning in research environments and commercial use. There are several other directions of research that arise from the work carried out in this thesis. We highlight a number of directions in the following paragraphs.

Robustness in Training Deep neural networks have proven to be intrinsically vulnerable to specific manifestations of noise and the most effective method of protecting networks to such randomness is adversarial training. Adversarial training is a scheme in which adversarial examples, the product of an adversarial attack on input examples, are used as part of the

training procedure [114, 34, 33]. In practice, this is a computationally expensive process, particularly when dealing with large networks and datasets. From the findings in Chapter 3 and Chapter 4, we observe comparable network performances, and in some cases improved performance, when we omit large portions of the network periodically during the training process. There have also been numerous studies into few-shot learning methods, where partial examples of the complete dataset are used to train the network [219]. The notion that not all components of either the dataset or network are required to achieve satisfactory network performance is one of the key findings from our work. We outline a method of categorising different components of a network as fragile, robust, and antifragile relative to the influence of the categorised component, on the network performance.

Through the proposed identification method, we identify opportunities to investigate novel training mechanisms and schemes, such that the selective backpropagation of network weights is varied. Furthermore, future work into incorporating adversarial examples into the training process, as is the case in adversarial training, may also aid in identifying patterns of the fragility, robustness, and antifragility in network architectures. Such investigations of network characterisation may also draw parallels with research on regularisation techniques employed during network training [116, 117, 24]. There are directions of research into identifying overparameterised network architectures during the network training process using the study proposed in this thesis. The methodologies we detail in this thesis can be used in the synthesis of more robust networks, and this is shown in the results we present in the thesis.

Robust Model Designs Thus far we have discussed how the analytical methods we develop can be used during network training, in order to increase network performance and robustness. A possible direction of work leading on from these findings is also in the development of more robust network architectures. The robustness in question may be adversarial [19] and architectural [107], as both themes are investigated in conjunction in this work. An important aspect in developing more robust networks is the ability to compare different network architectures applied to different tasks. Using the defined relative adversarial magnitude constraints and systematic synaptic filtering techniques, we are able to compare the relative effects of internal and external perturbations on the network, thus identifying which are more fragile, robust, or antifragile. This finding is a prerequisite to an effective network selection procedure that experimentally identifies the critical strengths and weaknesses of different architectures.

Analysis to Application A significant aspect of this thesis is directed at applying the developed analysis methods on different applications of DNNs. In Chapter 5, we specifically investigate DNNs applied to two different tasks; firstly, predicting the optimum ECG signal denoising filter for noisy waveforms, and secondly, classifying human activities and intensities using an ultra-wideband (UWB) radar system. The task of radar signal classification poses a challenging task due to the nature of the noisy recordings and sensitivity of the radar system to environmental noise. We apply the developed analysis to identify fragile, robust, and antifragile network parameters and subsequently, apply selective backpropagation at periodic intervals in the training to improve the performance of the DNNs, both under adversarial settings and the regular test datasets. Future work may investigate the effectiveness of applying the analysis and selective backpropagation on the performance of DNNs for different tasks.

There exist various forms of perturbations for applications of DNNs. An investigation into the proposed analysis applied to other forms of dataset corruption, such as miss-labeling errors and dataset trend variations, is another direction of future work that has the potential to yield interesting results. The ultimate goal of such an undertaking is to replicate the potential performances of DNNs developed in and for research environments, to the real-world where the operating conditions are seldom ideal.

6.2 Outlook

We, in this thesis, have investigated the robustness analysis of deep neural networks both under adversarial settings (external stress) and network parameter perturbations (internal stress). We have developed the notions of fragility, robustness, and introduced the concept of antifragile within the context of DNNs, to aid in improving network performances to various forms of disturbances. This has allowed us to experimentally observe several interesting phenomena of DNNs, whose behaviour when subjected external and internal stress exhibits various commonalities between different network architectures for the same task, and equally, the stress response of a particular network architecture exhibits commonalities when applied to different tasks. We show that selectively updating specific network parameters, and omitting others, results in networks that perform better on adversarial examples. Indeed, we also observe that, in certain instances, the network performance on a regular dataset is also improved using the proposed selective training method.

In finding patterns of behaviour within DNNs, we show that the task of robustness analysis, and in particular adversarial robustness, offers the opportunity to understand the

what explicitly renders certain networks robust and other not. Beyond the research of robustness analysis, we also hope that the work detailed in this thesis may assist in the development of stronger networks designed for application in the real-world. We hope that the methodologies discussed in this thesis, and the preliminary results presented, act as motivation for further investigations into the fascinating phenomena of deep neural networks and adversarial properties.

References

[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553): 436–444, 2015.

[2] H. Brink, J. Richards, and M. Fetherolf. *Real-World Machine Learning*. Manning, 2016. ISBN 9781638357001. URL https://books.google.co.uk/books?id=zTczEAAAQBAJ.

[3] Wojciech Samek, Grégoire Montavon, et al. Explaining deep neural networks and beyond: A review of methods and applications. *Proc IEEE*, 109(3):247–278, 2021.

[4] Jerome H Friedman. Recent advances in predictive (machine) learning. *Journal of classification*, 23(2):175–197, 2006.

[5] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[7] Riccardo Miotto, Fei Wang, Shuang Wang, Xiaoqian Jiang, and Joel T Dudley. Deep learning for healthcare: review, opportunities and challenges. *Briefings in bioinformatics*, 19(6):1236–1246, 2018.

[8] Dong Yu and Li Deng. *Automatic speech recognition*, volume 1. Springer, 2016.

[9] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, and Eftychios Protopapadakis. Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, 2018, 2018.

[10] Dan Guest, Kyle Cranmer, and Daniel Whiteson. Deep learning and its application to lhc physics. *arXiv preprint arXiv:1806.11484*, 2018.

[11] June-Goo Lee, Sanghoon Jun, Young-Won Cho, Hyunna Lee, Guk Bae Kim, Joon Beom Seo, and Namkug Kim. Deep learning in medical imaging: general overview. *Korean journal of radiology*, 18(4):570–584, 2017.

[12] Christof Angermueller, Tanel Pärnamaa, Leopold Parts, and Oliver Stegle. Deep learning for computational biology. *Molecular systems biology*, 12(7):878, 2016.

[13] Mohamed Amine Ferrag, Leandros Maglaras, Sotiris Moschoyiannis, and Helge Janicke. Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. *Journal of Information Security and Applications*, 50:102419, 2020.

[14] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[15] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8821–8831. PMLR, 18–24 Jul 2021.

[16] Ben Goertzel. Artificial general intelligence: concept, state of the art, and future prospects. *Journal of Artificial General Intelligence*, 5(1):1, 2014.

[17] Anil K Jain, Jianchang Mao, and K Moidin Mohiuddin. Artificial neural networks: A tutorial. *Computer*, 29(3):31–44, 1996.

[18] Timo Freiesleben. The intriguing relation between counterfactual explanations and adversarial examples. *Minds and Machines*, 32(1):77–109, 2022.

[19] Christian Szegedy, Wojciech Zaremba, et al. Intriguing properties of neural networks. In *ICLR*, 2014.

[20] Battista Biggio, Igino Corona, et al. Evasion attacks against machine learning at test time. In *ECML PKDD*, 2013.

[21] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proc 10th ACM Workshop Artif Intell Secur*, pages 3–14, 2017.

[22] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *CoRR*, abs/1607.02533, 2016.

[23] Jiakai Wang. Adversarial examples in physical world. In *IJCAI*, 2021.

[24] Chongzhi Zhang, Aishan Liu, Xianglong Liu, Yitao Xu, Hang Yu, Yuqing Ma, and Tianlin Li. Interpreting and improving adversarial robustness of deep neural networks with neuron sensitivity. *IEEE Transactions on Image Processing*, 30:1291–1304, 2021.

[25] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. On evaluating adversarial robustness, 2019.

[26] Naveed Akhtar and Ajmal Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 6:14410–14430, 2018.

[27] Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. Adversarial attacks and defences: A survey. *CoRR*, abs/1810.00069, 2018.

[28] Renato Umeton, Giovanni Stracquadanio, Anilkumar Sorathiya, Pietro Liò, Alessio Papini, and Giuseppe Nicosia. Design of robust metabolic pathways. In Leon Stok and et al, editors, *Proceedings of the 48th Design Automation Conference, DAC 2011, San Diego, California, USA, June 5-10*, pages 747–752. ACM, 2011.

[29] Samuel Henrique Silva and Peyman Najafirad. Opportunities and challenges in deep learning adversarial robustness: A survey. *arXiv:2007.00753*, 2020.

[30] Kui Ren, Tianhang Zheng, Zhan Qin, and Xue Liu. Adversarial attacks and defenses in deep learning. *Eng*, 6(3):346–360, 2020.

[31] Xiaoyong Yuan, Pan He, et al. Adversarial examples: Attacks and defenses for deep learning. *IEEE Trans Neural Netw Learn Syst*, 30(9):2805–2824, 2019.

[32] Valentina Zantedeschi, Maria-Irina Nicolae, and Ambrish Rawat. Efficient defenses against adversarial attacks. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, AISec '17, page 39–49, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450352024.

[33] Klim Kireev, Maksym Andriushchenko, and Nicolas Flammarion. On the effectiveness of adversarial training against common corruptions. In James Cussens and Kun Zhang, editors, *Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence*, volume 180 of *Proceedings of Machine Learning Research*, pages 1012–1021. PMLR, 01–05 Aug 2022.

[34] Eric Wong, Leslie Rice, and J Zico Kolter. Fast is better than free: Revisiting adversarial training. In *ICLR*, 2020.

[35] Nitish Srivastava, Geoffrey Hinton, et al. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958, 2014.

[36] H Wang, Z Lei, X Zhang, B Zhou, and J Peng. Machine learning basics. *Deep learning*, pages 98–164, 2016.

[37] Leslie Rice, Eric Wong, and Zico Kolter. Overfitting in adversarially robust deep learning. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 8093–8104. PMLR, 13–18 Jul 2020.

[38] Nassim Nicholas Taleb and Raphael Douady. Mathematical definition, mapping, and detection of (anti) fragility. *Quantitative Finance*, 13(11):1677–1689, 2013.

[39] Christine N. Paulson, John T. Chang, Carlos E. Romero, Joseph Watson M.D., Fred J. Pearce, and Nathan Levin M.D. Ultra-wideband radar methods and techniques of medical sensing and imaging. In Brian M. Cullum and J. Chance Carter, editors, *Smart Medical and Biomedical Sensor Technology III*, volume 6007, pages 96 – 107. International Society for Optics and Photonics, SPIE, 2005.

[40] Jing Li, Lanbo Liu, Zhaofa Zeng, and Fengshan Liu. Advanced signal processing for vital sign extraction with applications in UWB radar detection of trapped victims in complex environments. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 7(3):783–791, 2013.

[41] Md Ashfanoor Kabir and Celia Shahnaz. Denoising of ECG signals based on noise reduction algorithms in EMD and wavelet domains. *Biomedical Signal Processing and Control*, 7(5):481–489, 2012.

[42] Chandresh Pravin and Varun Ojha. A novel ecg signal denoising filter selection algorithm based on conventional neural networks. In *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1094–1100, 2020.

[43] Mikolaj Czerkawski, Carmine Clemente, Craig Michie, Ivan Andonovic, and Christos Tachtatzis. Robustness of deep neural networks for micro-doppler radar classification. In *2022 23rd International Radar Symposium (IRS)*, pages 480–485, 2022.

[44] Rahul K. Vigneswaran, R. Vinayakumar, K.P. Soman, and Prabaharan Poornachandran. Evaluating shallow and deep neural networks for network intrusion detection systems in cyber security. In *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pages 1–6, 2018. doi: 10.1109/ICCCNT.2018.8494096.

[45] Changhui Jiang, Jichun Shen, Shuai Chen, Yuwei Chen, Di Liu, and Yuming Bo. Uwb nlos/los classification using deep learning method. *IEEE Communications Letters*, 24 (10):2226–2230, 2020.

[46] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

[47] Chandresh Pravin, Ivan Martino, et al. Adversarial robustness in deep learning: attacks on fragile neurons. In *ICANN*, 2021.

[48] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d
textquotesingle Alch
'e-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[49] Sébastien Marcel and Yann Rodriguez. Torchvision the machine-vision package of torch. In *Proceedings of the 18th ACM International Conference on Multimedia*, MM '10, page 1485–1488, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781605589336. doi: 10.1145/1873951.1874254.

[50] al. Charles R. Harris et. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2.

[51] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

[52] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. In *ICLR*, 2016.

[53] Sandy Huang, Nicolas Papernot, et al. Adversarial attacks on neural network policies. In *ICLR*, 2017.

[54] Han Xu, Yao Ma, et al. Adversarial attacks and defenses in images, graphs and text: A review. *Int. J. of Autom. and Comput.*, 17(2):151–178, 2020.

[55] Shixiang Gu and Luca Rigazio. Towards deep neural network architectures robust to adversarial examples, 2014.

[56] Michael Weisberg. Robustness analysis. *Philosophy of Science*, 73(5):730–742, 2006. doi: 10.1086/518628.

[57] Huan Zhang, Hongge Chen, Chaowei Xiao, Sven Gowal, Robert Stanforth, Bo Li, Duane Boning, and Cho-Jui Hsieh. Towards stable and efficient training of verifiably robust neural networks, 2019.

[58] Daniel Hernández-lobato, Jose Hernández-lobato, and Pierre Dupont. Robust multi-class gaussian process classification. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011. URL https://proceedings.neurips.cc/paper/2011/file/7eabe3a1649ffa2b3ff8c02ebfd5659f-Paper.pdf.

[59] Di Tang, XiaoFeng Wang, Haixu Tang, and Kehuan Zhang. Demon in the variant: Statistical analysis of DNNs for robust backdoor contamination detection. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 1541–1558. USENIX Association, August 2021. ISBN 978-1-939133-24-3.

[60] Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky. Spectrally-normalized margin bounds for neural networks. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[61] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[62] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. On the (statistical) detection of adversarial examples. arXiv:1702.06280, 2017.

[63] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1310–1320. PMLR, 09–15 Jun 2019.

[64] Shao-Bo Lin. Generalization and expressivity for deep nets. *IEEE Transactions on Neural Networks and Learning Systems*, 30(5):1392–1406, 2019. doi: 10.1109/TNNLS.2018.2868980.

[65] Ignacio Lopez-Moreno, Javier Gonzalez-Dominguez, Oldrich Plchot, David Martinez, Joaquin Gonzalez-Rodriguez, and Pedro Moreno. Automatic language identification using deep neural networks. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5337–5341, 2014.

[66] Hugo Larochelle, Yoshua Bengio, Jérôme Louradour, and Pascal Lamblin. Exploring strategies for training deep neural networks. *Journal of machine learning research*, 10 (1), 2009.

[67] Li Deng, Geoffrey Hinton, and Brian Kingsbury. New types of deep neural network learning for speech recognition and related applications: an overview. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8599–8603, 2013. doi: 10.1109/ICASSP.2013.6639344.

[68] Jagjeevan Singh Shergill, Chandresh Pravin, and Varun Ojha. Accent and gender recognition from english language speech and audio using signal processing and deep learning. In *International Conference on Hybrid Intelligent Systems*, pages 62–72. Springer, 2020.

[69] U Rajendra Acharya, Hamido Fujita, Shu Lih Oh, Yuki Hagiwara, Jen Hong Tan, and Muhammad Adam. Application of deep convolutional neural network for automated detection of myocardial infarction using ECG signals. *Information Sciences*, 415: 190–198, 2017.

[70] Roshan Joy Martis, U. Rajendra Acharya, K.M. Mandana, A.K. Ray, and Chandan Chakraborty. Application of principal component analysis to ECG signals for automated diagnosis of cardiac health. *Expert Systems with Applications*, 39(14): 11792–11800, 2012. ISSN 0957-4174.

[71] Nikolaus Kriegeskorte. Deep neural networks: a new framework for modelling biological vision and brain information processing. *bioRxiv*, 2015. doi: 10.1101/029876.

[72] Xinyu Li, Yuan He, and Xiaojun Jing. A survey of deep learning-based human activity recognition in radar. *Remote Sensing*, 11(9):1068, 2019.

[73] Billy Ward, Chandresh Pravin, Alec Chetcuti, Yoshikatsu Hayashi, and Varun Ojha. Classification of musical preference in generation z through eeg signal processing and machine learning. In *International Conference on Intelligent Systems Design and Applications*, pages 117–127. Springer, 2020.

[74] Plamen Angelov and Alessandro Sperduti. Challenges in deep learning. 2016.

[75] Li Deng and Dong Yu. Deep learning: methods and applications. *Foundations and trends in signal processing*, 7(3–4):197–387, 2014.

[76] Alhussein Fawzi, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Robustness of classifiers: from adversarial to random noise. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[77] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu. Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(10):1533–1545, 2014.

[78] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[79] Guneet S. Dhillon, Kamyar Azizzadenesheli, Zachary C. Lipton, Jeremy Bernstein, Jean Kossaifi, Aran Khanna, and Anima Anandkumar. Stochastic activation pruning for robust adversarial defense, 2018.

[80] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456. PMLR, 2015.

[81] Hossein Gholamalinezhad and Hossein Khosravi. Pooling methods in deep neural networks, a review, 2020.

[82] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. Deep learning for generic object detection: A survey. *International journal of computer vision*, 128(2):261–318, 2020.

[83] Weibo Liu, Zidong Wang, Xiaohui Liu, Nianyin Zeng, Yurong Liu, and Fuad E. Alsaadi. A survey of deep neural network architectures and their applications. *Neurocomputing*, 234:11–26, 2017. ISSN 0925-2312.

[84] Chiyuan Zhang, Samy Bengio, and Yoram Singer. Are all layers created equal? *Journal of Machine Learning Research*, 2019.

[85] Seul-Ki Yeom, Philipp Seegerer, Sebastian Lapuschkin, Alexander Binder, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. Pruning by explaining: A novel criterion for deep neural network pruning. *Pattern Recognition*, 115:107899, 2021. ISSN 0031-3203.

[86] Oliver Willers, Sebastian Sudholt, Shervin Raafatnia, and Stephanie Abrecht. Safety concerns and mitigation approaches regarding the use of deep learning in safety-critical perception tasks. In António Casimiro, Frank Ortmeier, Erwin Schoitsch, Friedemann Bitsch, and Pedro Ferreira, editors, *Computer Safety, Reliability, and Security. SAFECOMP 2020 Workshops*, pages 336–350, Cham, 2020. Springer International Publishing.

[87] Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models, 2017.

[88] Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Analysis of classifiers' robustness to adversarial perturbations. *Machine learning*, 107(3):481–508, 2018.

[89] Gregor Urban, Krzysztof J. Geras, Samira Ebrahimi Kahou, Ozlem Aslan, Shengjie Wang, Rich Caruana, Abdelrahman Mohamed, Matthai Philipose, and Matt Richardson. Do deep convolutional nets really need to be deep and convolutional?, 2016.

[90] Frank Seide, Gang Li, and Dong Yu. Conversational speech transcription using context-dependent deep neural networks. In *Twelfth annual conference of the international speech communication association*, 2011.

[91] David Eigen, Jason Rolfe, Rob Fergus, and Yann LeCun. Understanding deep architectures using a recursive convolutional network, 2013.

[92] Siddhant Garg, Adarsh Kumar, Vibhor Goel, and Yingyu Liang. Can adversarial weight perturbations inject neural backdoors. In *Proceedings of the 29th ACM International Conference on Information & amp; Knowledge Management*, CIKM '20, page 2029–2032, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450368599.

[93] Monique F Crane, Ben J Searle, Maria Kangas, and Y Nwiran. How resilience is strengthened by exposure to stressors: The systematic self-reflection model of resilience strengthening. *Anxiety, Stress, & Coping*, 32(1):1–17, 2019.

[94] Marco Del Giudice, C Loren Buck, Lauren E Chaby, Brenna M Gormally, Conor C Taff, Christopher J Thawley, Maren N Vitousek, and Haruka Wada. What Is Stress? A Systems Perspective. *Integrative and Comparative Biology*, 58(6):1019–1032, 09 2018. ISSN 1540-7063.

[95] Barry S Oken, Irina Chamine, and Wayne Wakeland. A systems approach to stress, stressors and resilience in humans. *Behavioural brain research*, 282:144–154, 2015.

[96] Vivek Ramanujan, Mitchell Wortsman, Aniruddha Kembhavi, Ali Farhadi, and Mohammad Rastegari. What's hidden in a randomly weighted neural network? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11893–11902, 2020.

[97] Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[98] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S. Emer. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12): 2295–2329, 2017. doi: 10.1109/JPROC.2017.2761740.

[99] Ali Alqahtani, Xianghua Xie, and Mark W. Jones. Literature review of deep network compression. *Informatics*, 8(4), 2021. ISSN 2227-9709.

[100] Neil C. Thompson, Kristjan Greenewald, Keeheon Lee, and Gabriel F. Manso. The computational limits of deep learning, 2020.

[101] Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. What is the state of neural network pruning? In I. Dhillon, D. Papailiopoulos, and V. Sze, editors, *Proceedings of Machine Learning and Systems*, volume 2, pages 129–146, 2020.

[102] Song Han, Jeff Pool, et al. Learning both weights and connections for efficient neural networks. In *NIPS*, 2015.

[103] Philippe Lauret, Eric Fock, and Thierry Alex Mara. A node pruning algorithm based on a fourier amplitude sensitivity test method. *IEEE transactions on neural networks*, 17(2):273–293, 2006.

[104] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I. Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S. Davis. Nisp: Pruning networks using neuron importance score propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[105] Yiren Zhao, Xitong Gao, Daniel Bates, Robert Mullins, and Cheng-Zhong Xu. Focused quantization for sparse cnns. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/58aaee7ae94b52697ad3b9275d46ec7f-Paper.pdf.

[106] Roberto Rigamonti, Amos Sironi, Vincent Lepetit, and Pascal Fua. Learning separable filters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013.

[107] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. Model compression and acceleration for deep neural networks: The principles, progress, and challenges. *IEEE Signal Processing Magazine*, 35(1):126–136, 2018. doi: 10.1109/MSP.2017.2765695.

[108] Changan Chen, Frederick Tung, Naveen Vedula, and Greg Mori. Constraint-aware deep neural network compression. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[109] Zhengli Zhao, Dheeru Dua, and Sameer Singh. Generating natural adversarial examples, 2017.

[110] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. Deep networks with stochastic depth. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 646–661, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46493-0.

[111] Grégoire Montavon, G Orr, and Klaus-Robert Müller. Neural networks-tricks of the trade second edition. *Springer, DOI*, 10:978–3, 2012.

[112] Shaokai Ye, Kaidi Xu, Sijia Liu, Hao Cheng, Jan-Henrik Lambrechts, Huan Zhang, Aojun Zhou, Kaisheng Ma, Yanzhi Wang, and Xue Lin. Adversarial robustness vs. model compression, or both? In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[113] Artur Jordão and Hélio Pedrini. On the effect of pruning on adversarial robustness. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pages 1–11, October 2021.

[114] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016.

[115] Jing Han, Zixing Zhang, Nicholas Cummins, and Björn Schuller. Adversarial training in affective computing and sentiment analysis: Recent advances and perspectives [review article]. *IEEE Computational Intelligence Magazine*, 14(2):68–81, 2019.

[116] Yang Bai, Yuyuan Zeng, Yong Jiang, Shu-Tao Xia, Xingjun Ma, and Yisen Wang. Improving adversarial robustness via channel-wise activation suppressing, 2021.

[117] Adnan Siraj Rakin, Zhezhi He, Li Yang, Yanzhi Wang, Liqiang Wang, and Deliang Fan. Robust sparse regularization: Simultaneously optimizing neural network robustness and compactness, 2019.

[118] Cihang Xie and Alan Yuille. Intriguing properties of adversarial training at scale, 2019.

[119] Francesco Croce and Matthias Hein. On the interplay of adversarial robustness and architecture components: patches, convolution and attention, 2022.

[120] Nikolaos Pitropakis, Emmanouil Panaousis, Thanassis Giannetsos, Eleftherios Anastasiadis, and George Loukas. A taxonomy and survey of attacks against machine learning. *Computer Science Review*, 34:100199, 2019. ISSN 1574-0137.

[121] Nicholas Carlini and David Wagner. Audio adversarial examples: Targeted attacks on speech-to-text. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 1–7, 2018. doi: 10.1109/SPW.2018.00009.

[122] Teng Huang, Qixiang Zhang, Jiabao Liu, Ruitao Hou, Xianmin Wang, and Ya Li. Adversarial attacks on deep-learning-based sar image target recognition. *Journal of Network and Computer Applications*, 162:102632, 2020. ISSN 1084-8045.

[123] Wei Emma Zhang, Quan Z. Sheng, Ahoud Alhazmi, and Chenliang Li. Adversarial attacks on deep-learning models in natural language processing: A survey. *ACM Trans. Intell. Syst. Technol.*, 11(3), apr 2020. ISSN 2157-6904.

[124] Samuel G. Finlayson, John D. Bowers, Joichi Ito, Jonathan L. Zittrain, Andrew L. Beam, and Isaac S. Kohane. Adversarial attacks on medical machine learning. *Science*, 363(6433):1287–1289, 2019.

[125] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-GAN: Protecting classifiers against adversarial attacks using generative models. In *ICLR*, 2018.

[126] Andrew Ilyas, Shibani Santurkar, et al. Adversarial examples are not bugs, they are features. In *NIPS*, 2019.

[127] Pengzhan Jin, Lu Lu, Yifa Tang, and George Em Karniadakis. Quantifying the generalization error in deep learning in terms of data distribution and neural network smoothness. *Neural Networks*, 130:85–99, 2020. ISSN 0893-6080.

[128] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. On the (statistical) detection of adversarial examples, 2017.

[129] Matthias Hein and Maksym Andriushchenko. Formal guarantees on the robustness of a classifier against adversarial manipulation. *arXiv preprint arXiv:1705.08475*, 2017.

[130] Jacob Dumford and Walter Scheirer. Backdooring convolutional neural networks via targeted weight perturbations. In *2020 IEEE International Joint Conference on Biometrics (IJCB)*, pages 1–9, 2020.

[131] Gil Fidel, Ron Bitton, and Asaf Shabtai. When explainability meets adversarial learning: Detecting adversarial examples using shap signatures. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2020.

[132] Alexey Ignatiev, Nina Narodytska, and Joao Marques-Silva. On relating explanations and adversarial examples. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d. Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[133] Yinpeng Dong, Hang Su, Jun Zhu, and Fan Bao. Towards interpretable deep neural networks by leveraging adversarial examples, 2017.

[134] Beomsu Kim, Junghoon Seo, and Taegyun Jeon. Bridging adversarial robustness and gradient interpretability. *CoRR*, abs/1903.11626, 2019.

[135] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks, 2017.

[136] Ali Shafahi, Mahyar Najibi, Mohammad Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[137] Alfred Laugros, Alice Caplier, and Matthieu Ospici. Are adversarial robustness and common perturbation robustness independant attributes ? In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, Oct 2019.

[138] Justin Gilmer, Nicolas Ford, Nicholas Carlini, and Ekin Cubuk. Adversarial examples are a natural consequence of test error in noise. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2280–2289. PMLR, 09–15 Jun 2019.

[139] Dimitris Tsipras, Shibani Santurkar, et al. Robustness may be at odds with accuracy. In *ICLR*, 2019.

[140] Bai Li, Changyou Chen, Wenlin Wang, and Lawrence Carin. Certified adversarial robustness with additive noise. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[141] Sevgi Zubeyde Gurbuz and Moeness G. Amin. Radar-based human-motion recognition with deep learning: Promising applications for indoor monitoring. *IEEE Signal Processing Magazine*, 36(4):16–28, 2019.

[142] William Taylor, Kia Dashtipour, Syed Aziz Shah, Amir Hussain, Qammer H Abbasi, and Muhammad A Imran. Radar sensing for activity classification in elderly people exploiting micro-doppler signatures using machine learning. *Sensors*, 21(11):3881, 2021.

[143] Ping Lang, Xiongjun Fu, Marco Martorella, Jian Dong, Rui Qin, Xianpeng Meng, and Min Xie. A comprehensive survey of machine learning applied to radar signal processing, 2020.

[144] Youngwook Kim and Hao Ling. Human activity classification based on micro-doppler signatures using a support vector machine. *IEEE Transactions on Geoscience and Remote Sensing*, 47(5):1328–1337, 2009.

[145] Branka Jokanovic, Moeness Amin, and Fauzia Ahmad. Radar fall motion detection using deep learning. In *2016 IEEE radar conference (RadarConf)*, pages 1–6. IEEE, 2016.

[146] Haipeng Wang, Sizhe Chen, Feng Xu, and Ya-Qiu Jin. Application of deep-learning algorithms to mstar data. In *2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 3743–3745, 2015.

[147] Maoguo Gong, Hailun Yang, and Puzhao Zhang. Feature learning and change feature classification based on deep learning for ternary change detection in sar images. *ISPRS Journal of Photogrammetry and Remote Sensing*, 129:212–225, 2017. ISSN 0924-2716.

[148] Antonio Lazaro, David Girbau, and Ramon Villarino. Analysis of vital signs monitoring using an IR-UWB radar. *Progress In Electromagnetics Research*, 100:265–284, 2010.

[149] Farnaz Shikhsarmast, Tingting Lyu, Xiaolin Liang, Hao Zhang, and Thomas Gulliver. Random-noise denoising and clutter elimination of human respiration movements based on an improved time window selection algorithm using wavelet transform. *Sensors*, 19(1):95, Dec 2018.

[150] Karol Antczak. Deep recurrent neural networks for ECG signal denoising. *arXiv preprint arXiv:1807.11551*, 2018.

[151] Yann LeCun and Corinna Cortes. MNIST handwritten digit database, 2010. http://yann.lecun.com/exdb/mnist/.

[152] Nicolas Papernot, Patrick McDaniel, et al. The limitations of deep learning in adversarial settings. In *EuroS&P*, pages 372–387, 2016.

[153] Nicholas Carlini and David Wagner. MagNet and "efficient defenses against adversarial attacks" are not robust to adversarial examples. *arXiv:1711.08478*, 2017.

[154] Giovanni Stracquadanio and Giuseppe Nicosia. Computational energy-based redesign of robust proteins. *Comput. Chem. Eng.*, 35(3):464–473, 2011.

[155] Bo Li and Cheng Chen. First-order sensitivity analysis for hidden neuron selection in layer-wise training of networks. *Neural Process Lett*, 48(2):1105–1121, 2018.

[156] Gabriel Goh et al. Multimodal neurons in artificial neural networks. *Distill*, 6(3), 2021.

[157] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *IEEE CVPR*, pages 2574–2582, 2016.

[158] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symp Secur Priv*, 2017.

[159] Huan Zhang et al. The limitations of adversarial training and the blind-spot attack. In *ICLR*, 2019.

[160] Lily Weng et al. Towards fast computation of certified robustness for relu networks. In *ICML*, pages 5276–5285, 2018.

[161] Tsui-Wei Weng, Huan Zhang, et al. Evaluating the robustness of neural networks: An extreme value theory approach. In *ICLR*, 2018.

[162] Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan L. Yuille, and Kaiming He. Feature denoising for improving adversarial robustness. In *IEEE CVPR*, 2019.

[163] Bolei Zhou, David Bau, Aude Oliva, and Antonio Torralba. Interpreting deep visual representations via network dissection. *IEEE Trans Pattern Anal Mach Intell*, 2018.

[164] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *IEEE CVPR*, pages 9304–9312, 2020.

[165] Nicholas Cheney, Martin Schrimpf, and Gabriel Kreiman. On the robustness of convolutional neural networks to internal architecture and weight perturbations. *arXiv:1703.08245*, 2017.

[166] Frederic Branchaud-Charron, Andrew Achkar, and Pierre-Marc Jodoin. Spectral metric for dataset complexity assessment. In *IEEE CVPR*, 2019.

[167] Matan Gavish and David L Donoho. The optimal hard threshold for singular values is $4/\sqrt{3}$. *IEEE Trans Inf Theory*, 60(8):5040–5053, 2014.

[168] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *Proc IEEE Symp Secur Priv (SP)*, 2017.

[169] Zelda Mariet and Suvrit Sra. Diversity networks: Neural network compression using determinantal point processes. *arXiv preprint arXiv:1511.05077*, 2015.

[170] Kaiming He, Xiangyu Zhang, et al. Deep residual learning for image recognition. In *CVPR*, 2016.

[171] Forrest N Iandola, Song Han, et al. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5mb model size. *arXiv:1602.07360v4*, 2016.

[172] Ningning Ma, Xiangyu Zhang, et al. ShuffleNet V2: Practical guidelines for efficient CNN architecture design. In *ECCV*, 2018.

[173] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge, 2015. Stanford CS 231N.

[174] Ilia N Karatsoreos and Bruce S McEwen. Psychobiological allostasis: resistance, resilience and vulnerability. *Trends in cognitive sciences*, 15(12):576–584, 2011.

[175] Xiang Gao, Ripon K Saha, Mukul R Prasad, and Abhik Roychoudhury. Fuzz testing based data augmentation to improve robustness of deep neural networks. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, pages 1147–1158. IEEE, 2020.

[176] Yajie Wang, Shangbo Wu, et al. Demiguise attack: Crafting invisible semantic adversarial perturbations with perceptual similarity. In *IJCAI*, 2021.

[177] Karthik Abinav Sankararaman, Soham De, et al. The impact of neural network overparameterization on gradient confusion and stochastic gradient descent. In *ICML*, 2020.

[178] Simon Kornblith, Mohammad Norouzi, et al. Similarity of neural network representations revisited. In *ICML*, 2019.

[179] Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(12):124003, 2021.

[180] Varun Ojha and Giuseppe Nicosia. Backpropagation neural tree. *Neural Networks*, 149:66–83, 2022.

[181] Rhian Taylor, Varun Ojha, Ivan Martino, and Giuseppe Nicosia. Sensitivity analysis for deep learning: ranking hyper-parameter influence. In *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 512–516. IEEE, 2021.

[182] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. In *ICLR*, 2017.

[183] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

[184] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[185] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[186] Kaiming He, Xiangyu Zhang, et al. Delving deep into rectifiers: Surpassing human-levelperformance on ImageNet classification. In *ICCV*, 2015.

[187] Suhail Khokhar, Abdullah Asuhaimi Mohd Zin, Ahmad Safawi Mokhtar, and M. Hashem Pesaran. A comprehensive overview on signal processing and artificial intelligence techniques applications in classification of power quality disturbances. *Renewable & Sustainable Energy Reviews*, 51:1650–1663, 2015.

[188] A. B. Nassif, I. Shahin, I. Attili, M. Azzeh, and K. Shaalan. Speech recognition using deep neural networks: A systematic review. *IEEE Access*, 7:19143–19165, 2019.

[189] Kenneth R. Foster, Robert Koprowski, and Joseph D. Skufca. Machine learning, medical diagnosis, and biomedical engineering research - commentary. *BioMedical Engineering OnLine*, 13:94 – 94, 2014.

[190] George B Moody and Roger G Mark. The impact of the MIT-BIH arrhythmia database. *IEEE Engineering in Medicine and Biology Magazine*, 20(3):45–50, 2001.

[191] J.G. Proakis and D.G. Manolakis. *Digital Signal Processing*. Pearson Prentice Hall, 2007. ISBN 9780131873742.

[192] Simon Haykin. *Adaptive Filter Theory (3rd Ed.)*. Prentice-Hall, Inc., USA, 1996. ISBN 013322760X.

[193] B. Kovačević, Z. Banjac, and M. Milosavljević. *Adaptive Digital Filters*. Springer Berlin Heidelberg, 2013. ISBN 9783642335617.

[194] S. L. Joshi, R. A. Vatti, and R. V. Tornekar. A survey on ECG signal denoising techniques. In *2013 International Conference on Communication Systems and Network Technologies*, pages 60–64, 2013.

[195] Heikki V Huikuri, Agustin Castellanos, and Robert J Myerburg. Sudden death due to cardiac arrhythmia's. *New England Journal of Medicine*, 345(20):1473–1482, 2001.

[196] Xiaolin Liang, Jianqin Deng, Hao Zhang, and Thomas Aaron Gulliver. Ultra-wideband impulse radar through-wall detection of vital signs. *Scientific Reports*, 8(1):1–21, 2018.

[197] Suranai Poungponsri and Xiao-Hua Yu. An adaptive filtering approach for electrocardiogram (ECG) signal noise reduction using neural networks. *Neurocomputing*, 117: 206 – 213, 2013. ISSN 0925-2312.

[198] Ge Wang, Lin Yang, Ming Liu, Xin Yuan, Peng Xiong, Feng Lin, and Xiuling Liu. ECG signal denoising based on deep factor analysis. *Biomedical Signal Processing and Control*, 57:101824, 2020. ISSN 1746-8094.

[199] Giovanni Pepe, Leonardo Gabrielli, Stefano Squartini, and Luca Cattani. Designing audio equalization filters by deep neural networks. *Applied Sciences*, 10(7):2483, Apr 2020. ISSN 2076-3417.

[200] W. Jenkal, R. Latif, A. Toumanari, A. Dliou, O. El B'Charri, and F.M.R. Maoulainine. An efficient algorithm of ECG signal denoising using the adaptive dual threshold filter and the discrete wavelet transform. *Biocybernetics and Biomedical Engineering*, 36 (3):499–508, 2016.

[201] Fernando Ramirez-Mireles. On the performance of ultra-wideband signals in gaussian noise and dense multipath. *IEEE Transactions on Vehicular Technology*, 50(1):244–249, 2001.

[202] Dengyong Zhang, Shanshan Wang, Feng Li, Jin Wang, Arun Kumar Sangaiah, Victor S Sheng, and Xiangling Ding. An ECG signal de-noising approach based on wavelet energy and sub-band smoothing filter. *Applied Sciences*, 9(22):4968, 2019.

[203] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. ISBN 9780262337373.

[204] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[205] Fatin A. Elhaj, Naomie Salim, Arief R. Harris, Tan Tian Swee, and Taqwa Ahmed. Arrhythmia recognition and classification using combined linear and nonlinear features of ECG signals. *Computer Methods and Programs in Biomedicine*, 127:52 – 63, 2016. ISSN 0169-2607.

[206] Gustavo López, Luis Quesada, and Luis A Guerrero. Alexa vs. siri vs. cortana vs. google assistant: a comparison of speech-based natural user interfaces. In *International Conference on Applied Human Factors and Ergonomics*, pages 241–250. Springer, 2017.

[207] Veton Kepuska and Gamal Bohouta. Next-generation of virtual personal assistants (microsoft cortana, apple siri, amazon alexa and google home). In *2018 IEEE 8th annual computing and communication workshop and conference (CCWC)*, pages 99–103. IEEE, 2018.

[208] Alan Messer, Anugeetha Kunjithapatham, Phuong Nguyen, Priyang Rathod, Mithun Sheshagiri, Doreen Cheng, and Simon Gibbs. Seensearch: A context directed search facilitator for home entertainment devices. *Pervasive and Mobile Computing*, 4(6): 871–888, 2008.

[209] Li Jiang, Da-You Liu, and Bo Yang. Smart home research. In *Proceedings of 2004 international conference on machine learning and cybernetics (IEEE Cat. No. 04EX826)*, volume 2, pages 659–663. IEEE, 2004.

[210] al. Bull, Fiona C et. World health organization 2020 guidelines on physical activity and sedentary behaviour. *British Journal of Sports Medicine*, 54(24):1451–1462, 2020. doi: 10.1136/bjsports-2020-102955.

[211] Annette Prüss-Üstün, J. Wolf, Carlos F. Corvalán, R. Bos, and Maria Purificación Neira. *Preventing disease through healthy environments: a global assessment of the burden of disease from environmental risks*. World Health Organization, 2016.

[212] Karen Milton, Adrian E Bauman, Guy Faulkner, Gerard Hastings, William Bellew, Chloë Williamson, and Paul Kelly. Maximising the impact of global and national physical activity guidelines: the critical role of communication strategies. *British Journal of Sports Medicine*, 54(24):1463–1467, 2020. doi: 10.1136/bjsports-2020-102324.

[213] Vini Vijayan, James P. Connolly, Joan Condell, Nigel McKelvey, and Philip Gardiner. Review of wearable devices and data collection considerations for connected health. *Sensors*, 21(16), 2021.

[214] Ke Wan Ching and Manmeet Mahinderjit Singh. Wearable technology devices security and privacy vulnerability analysis. *International Journal of Network Security & Its Applications*, 8(3):19–30, 2016.

[215] Jennifer A Bunn, James W Navalta, Charles J Fountaine, and Joel D Reece. Current state of commercial wearable technology in physical activity monitoring 2015–2017. *International journal of exercise science*, 11(7):503, 2018.

[216] Yang Yang, Chunping Hou, Yue Lang, Guanghui Yue, Yuan He, and Wei Xiang. Person identification using micro-doppler signatures of human motions and uwb radar. *IEEE Microwave and Wireless Components Letters*, 29(5):366–368, 2019.

[217] Daniele Ravi, Charence Wong, Benny Lo, and Guang-Zhong Yang. A deep learning approach to on-node sensor data analytics for mobile or wearable devices. *IEEE journal of biomedical and health informatics*, 21(1):56–64, 2016.

[218] Andrea Soro, Gino Brunner, Simon Tanner, and Roger Wattenhofer. Recognition and repetition counting for complex physical exercises with deep learning. *Sensors*, 19(3), 2019. ISSN 1424-8220. doi: 10.3390/s19030714.

[219] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip H.S. Torr, and Timothy M. Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

Appendix A

Appendix

We present the supplementary analysis on the learning landscape of different networks in the following sections. Sec. A.1, Sec. A.2, Sec. A.3, and Sec. A.4 for the ResNet-18, ResNet-50, SqueezeNet-V1.1, and ShuffleNet V2x1.0 results respectively. These results reflect network test accuracy for our synaptic filtering techniques presented in Secs 3.

A.1 ResNet-18

A.1.1 Clean Dataset Responses

The ResNet-18 network test accuracy responses to the proposed synaptic filters (See Sec. 3) on the clean, unperturbed datasets is given for all layers of the network. We present the ResNet-18 responses for synaptic filters h_1, h_2, h_3 and the combined system response (see Sec. 3.3) for the MNIST dataset in Figs. A.1 to A.4. The Pixel intensities in the presented figures show the measured network accuracy for all threshold values α ranging from α_0 to α_A, as per Sec.3, measured every 10 epochs from 10 to 100 epochs during network training. Similarly, the network responses for ResNet-18 on the CIFAR10 dataset, are presented in Figs. A.13 to A.16. The ResNet-18 network test accuracy responses for the ImageNet Tiny dataset are presented in Figs. A.25 to A.28. Using the results presented in Figs. A.1 to A.4, Figs. A.13 to A.16, and Figs. A.25 to A.28 we find layers, such as layers '*conv1*', '*layer3.0.conv1*' and '*layer4.1.conv1*' for example, where there are invariant response characteristics to different synaptic filters for the three evaluated datasets on the ResNet-18 network. Further analysis of these results is presented in Sec. 5.

A.1.2 Adversarial Dataset Responses

The ResNet-18 network test accuracy responses to the different synaptic filters (See Sec. 3) on the adversarial datasets is given for all layers of the network. We present the ResNet-18 responses for synaptic filters h_1, h_2, h_3 and the combined system response (see Sec. 3.3 Combined System Response) for the adversarially perturbed MNIST dataset in Figs. A.5 to A.8. The constraints for the perturbation magnitude of the attack is given in Sec. 3. The Pixel intensities in the presented figures show the measured network accuracy for all threshold values α ranging from α_0 to α_A, as per Sec. 3, measured every 10 epochs from 10 to 100 epochs during network training. Similarly, the network test accuracy responses for ResNet-18 on the adversarially perturbed CIFAR10 dataset, are presented in Figs. A.17 to A.20. The ResNet-18 network test accuracy responses for the adversarially perturbed ImageNet tiny dataset are presented in Figs. A.29 to A.32. Using the results presented in Figs. A.5 to A.8, Figs. A.17 to A.20, and Figs. A.29 to A.32 we find layers, similar to those highlighted for the clean dataset responses, where there are invariant response characteristics to different synaptic filters for the three evaluated adversarial datasets on the ResNet-18 network. Further analysis of results is given in Sec. 5.

A.1.3 Scaled Response Difference

The ResNet-18 network scaled test accuracy response differences (see Sec. 3.2) between the clean and adversarial datasets, to the different synaptic filters is given for all layers of the network. We present the ResNet-18 responses difference for synaptic filters h_1, h_2, h_3 and the combined system response (see Sec. 3.3) for the MNIST dataset in Figs. A.9 to A.12. The constraints for the perturbation magnitude of the attack is given in Sec. 3. The Pixel intensities in the presented figures show the measured network accuracy for all threshold values α ranging from α_0 to α_A, as per Sec. 3, measured every 10 epochs from 10 to 100 epochs during network training. The network test accuracy response differences for ResNet-18 on the CIFAR10 dataset, are presented in Figs. A.21 to A.24. The ResNet-18 network test accuracy response differences, for the ImageNet Tiny dataset are presented in Figs. A.33 to A.36. Using the results presented in Figs. A.9 to A.12, Figs. A.21 to A.24, and Figs. A.33 to A.36 we find links and layers where the adversary is targeting the ResNet-18 network. We also find layer response differences that show different characteristics for different datasets, further analysis of results is given in Sec. 5.

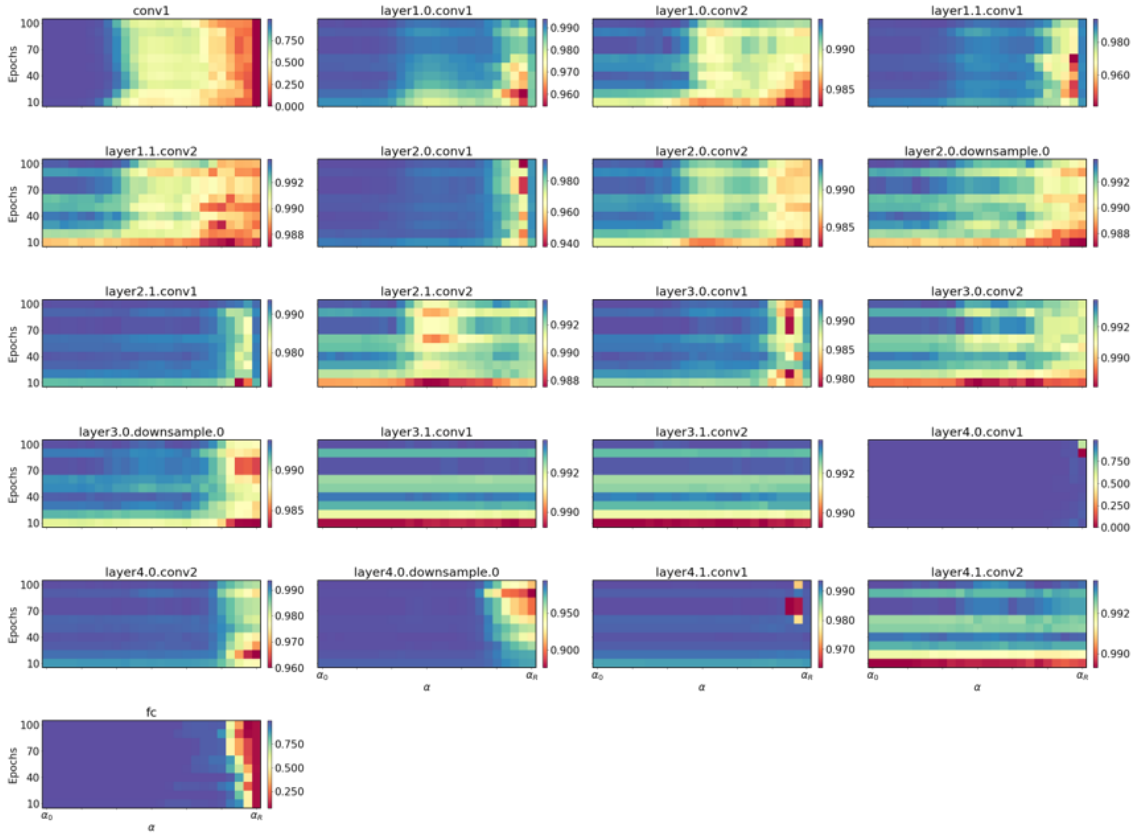


Fig. A.1 ResNet-18 response to clean MNIST dataset, for synaptic filter h_1.

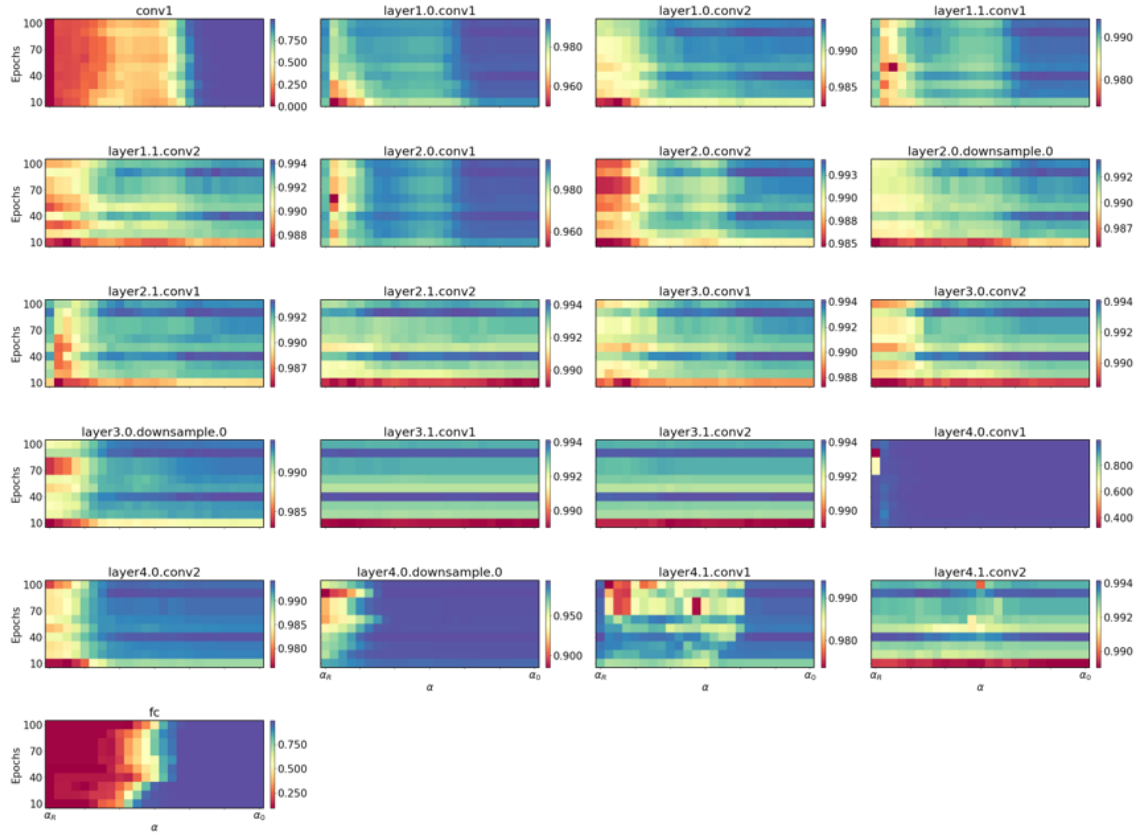


Fig. A.2 ResNet-18 response to clean MNIST dataset, for synaptic filter h_2.

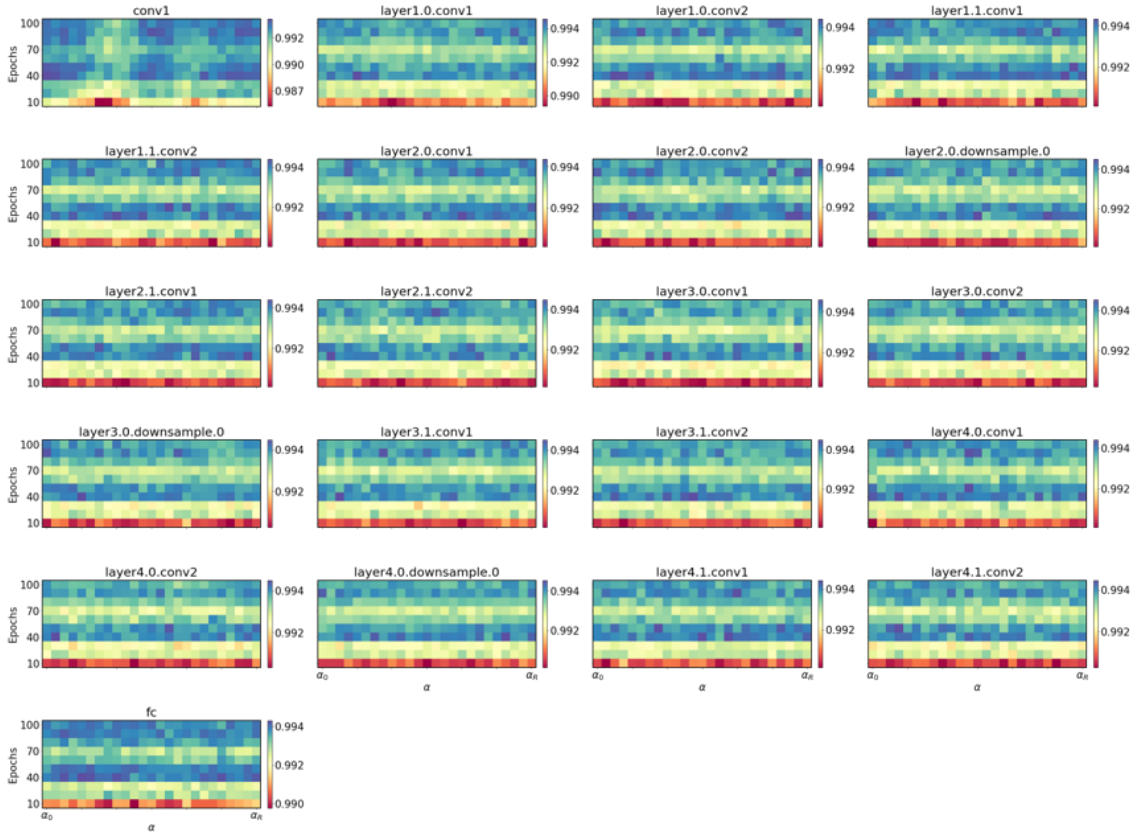


Fig. A.3 ResNet-18 response to clean MNIST dataset, for synaptic filter h_3.

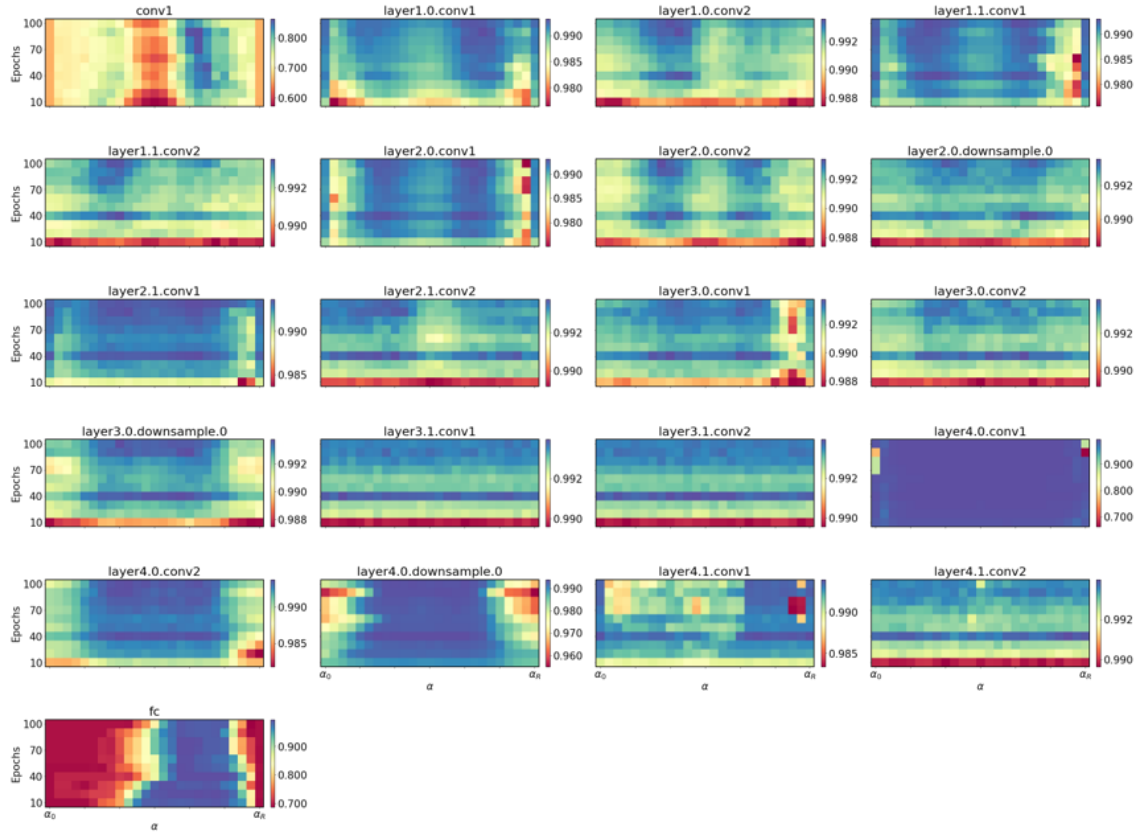


Fig. A.4 Combined ResNet-18 response to clean MNIST dataset, for all synaptic filters in h.

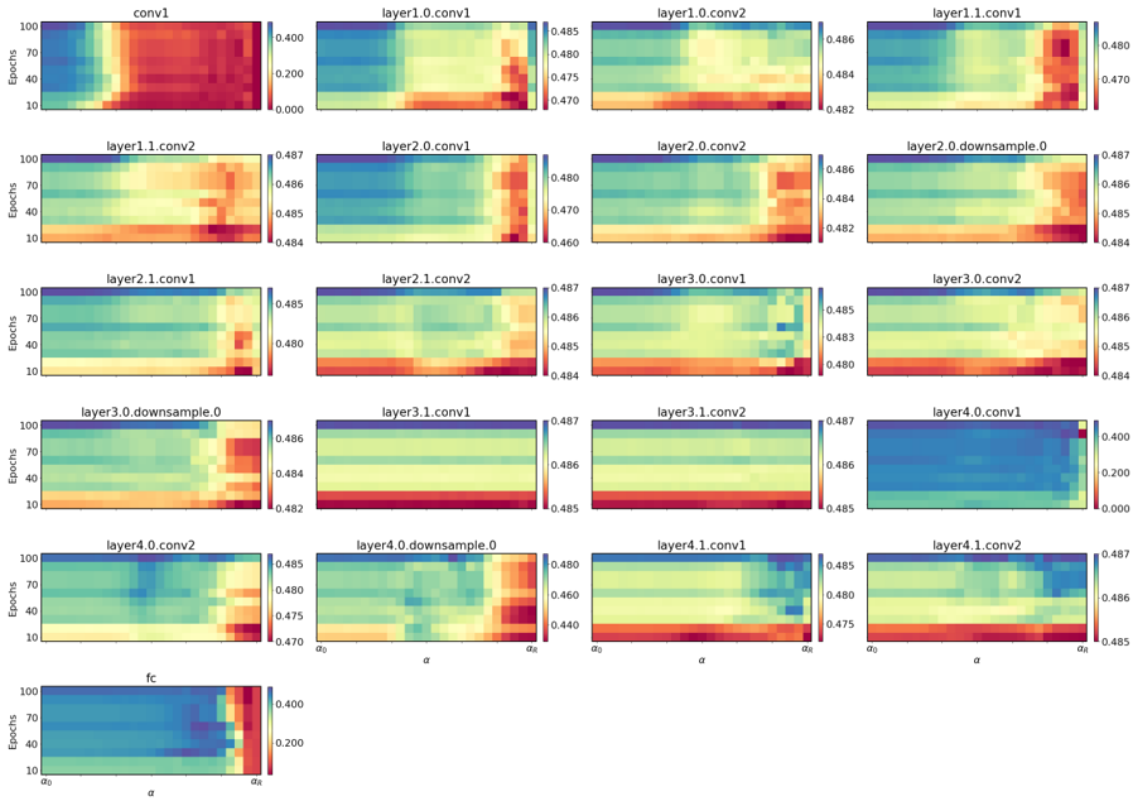


Fig. A.5 ResNet-18 response to adversarial MNIST dataset, for synaptic filter h_1.

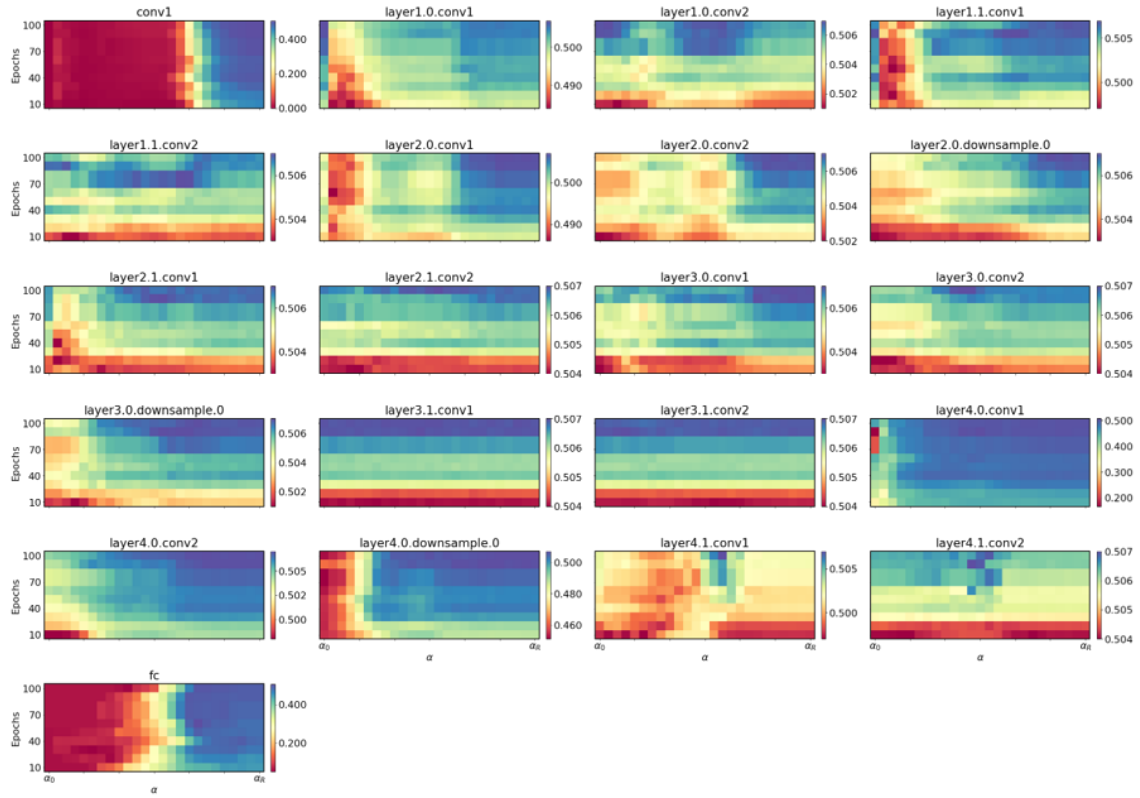


Fig. A.6 ResNet-18 response to adversarial MNIST dataset, for synaptic filter h_2.

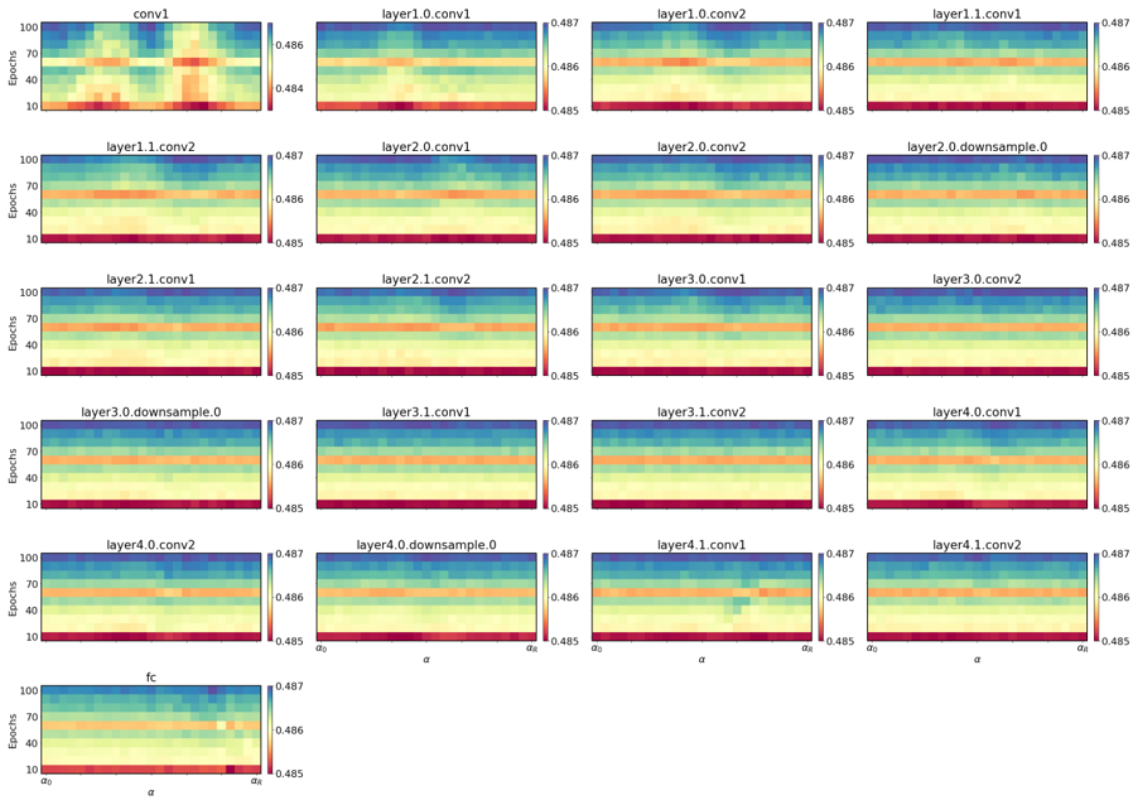


Fig. A.7 ResNet-18 response to adversarial MNIST dataset, for synaptic filter h_3.

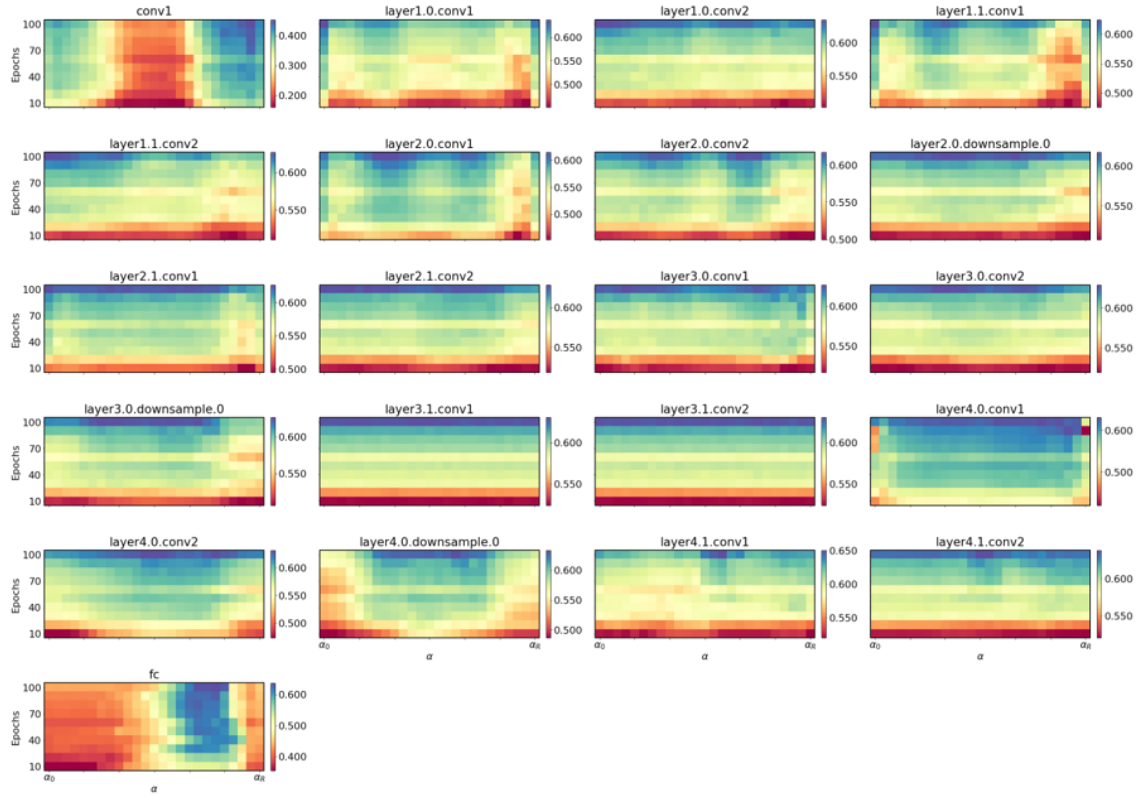


Fig. A.8 Combined ResNet-18 response to adversarial MNIST dataset, for all synaptic filters in h.

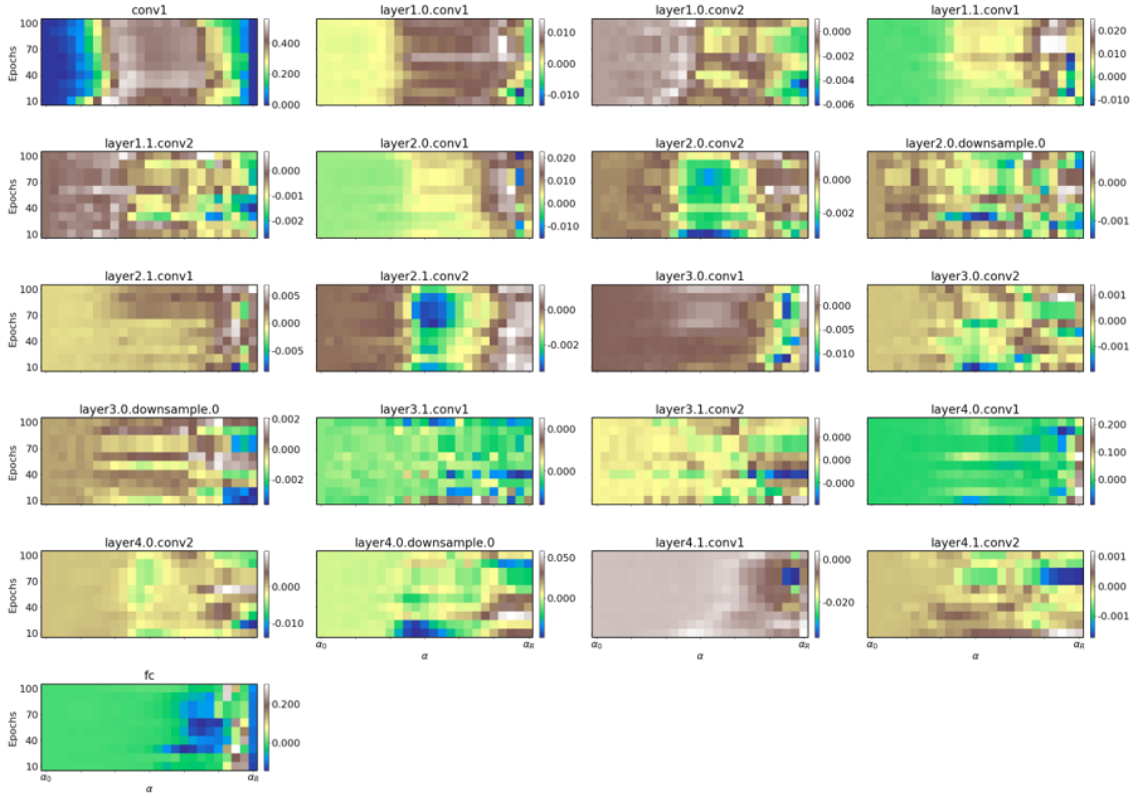


Fig. A.9 Difference in ResNet-18 responses to Clean and adversarial MNIST datasets, for synaptic filter h_1.

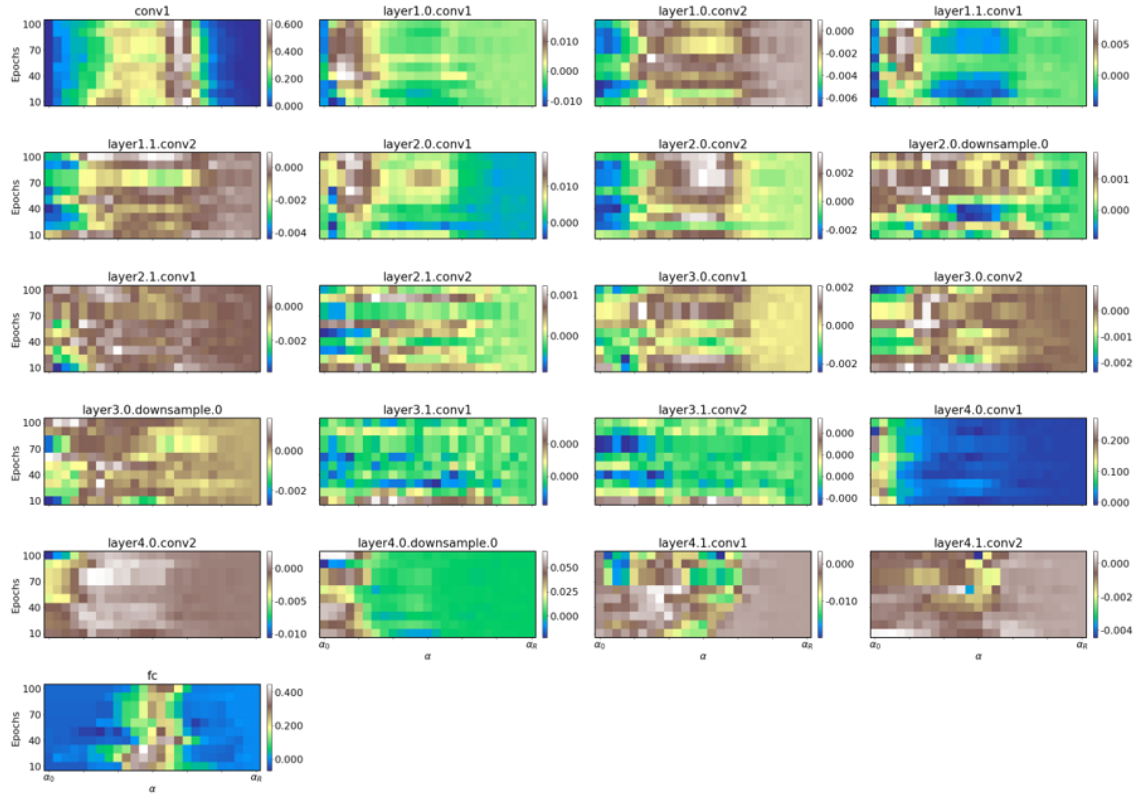


Fig. A.10 Difference in ResNet-18 responses to Clean and adversarial MNIST datasets, for synaptic filter h_2.

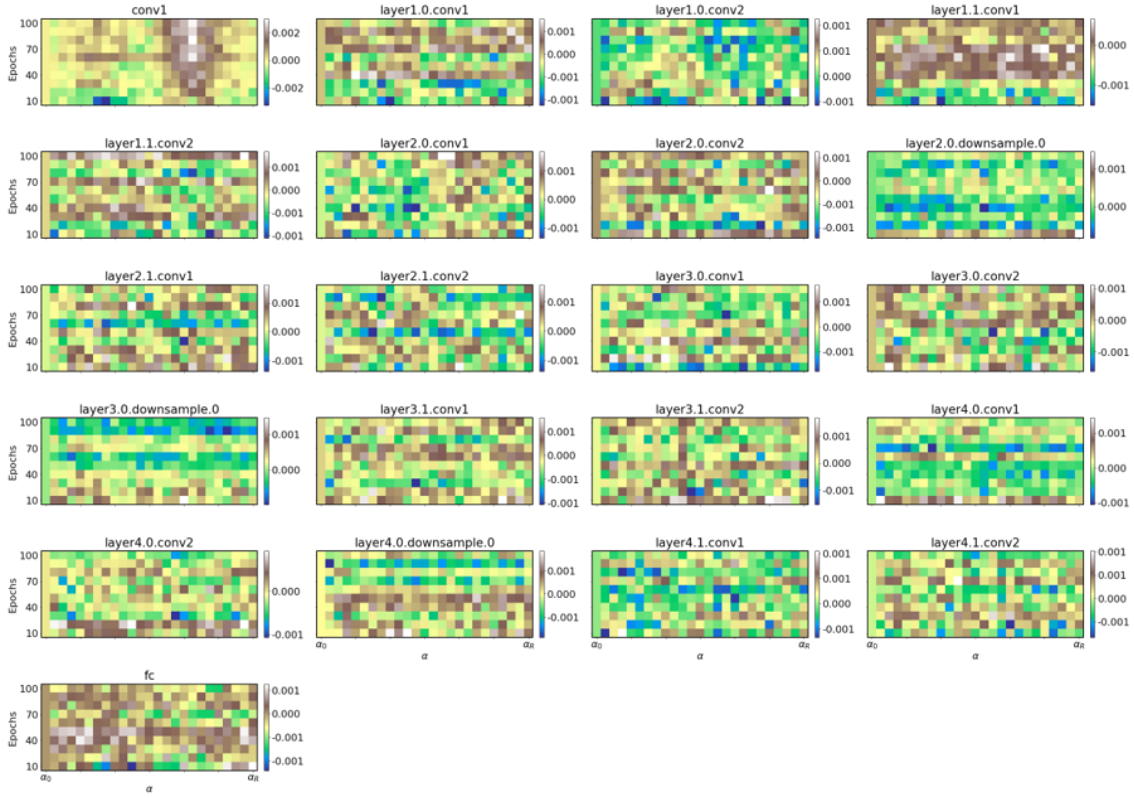


Fig. A.11 Difference in ResNet-18 responses to Clean and adversarial MNIST datasets, for synaptic filter h_3.

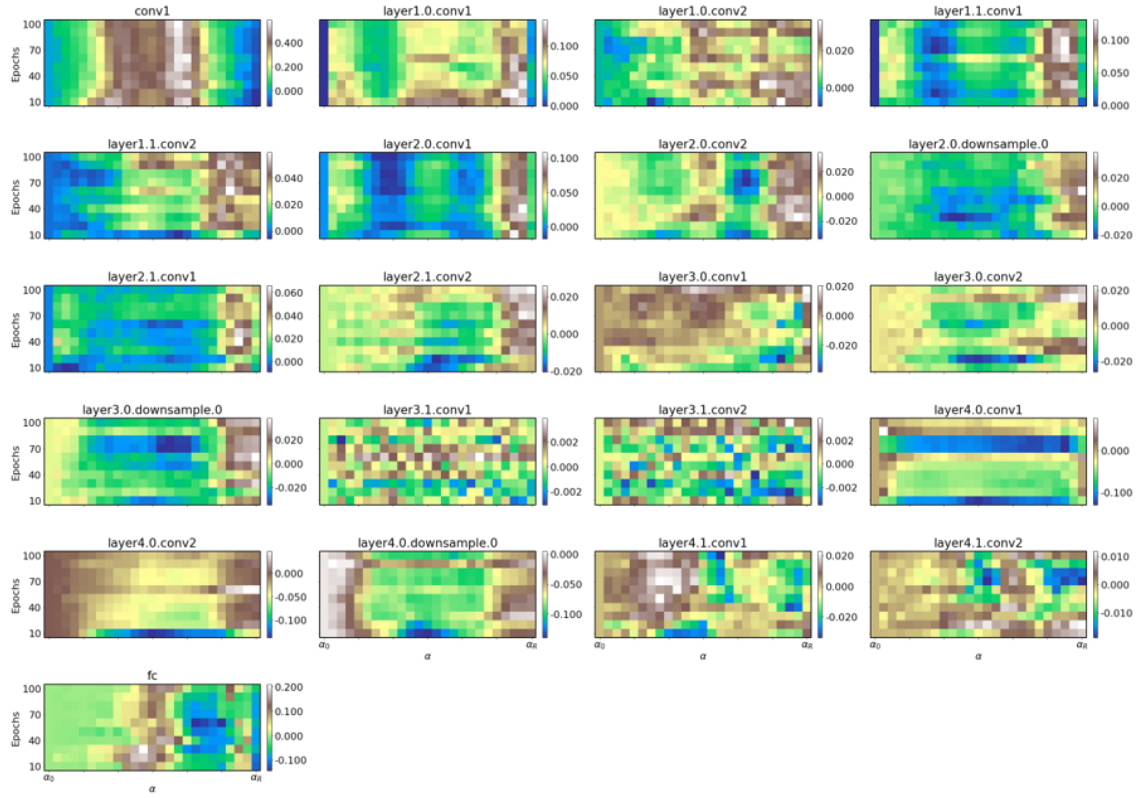


Fig. A.12 Combined difference in ResNet-18 responses to Clean and adversarial MNIST datasets, for all synaptic filters in h.

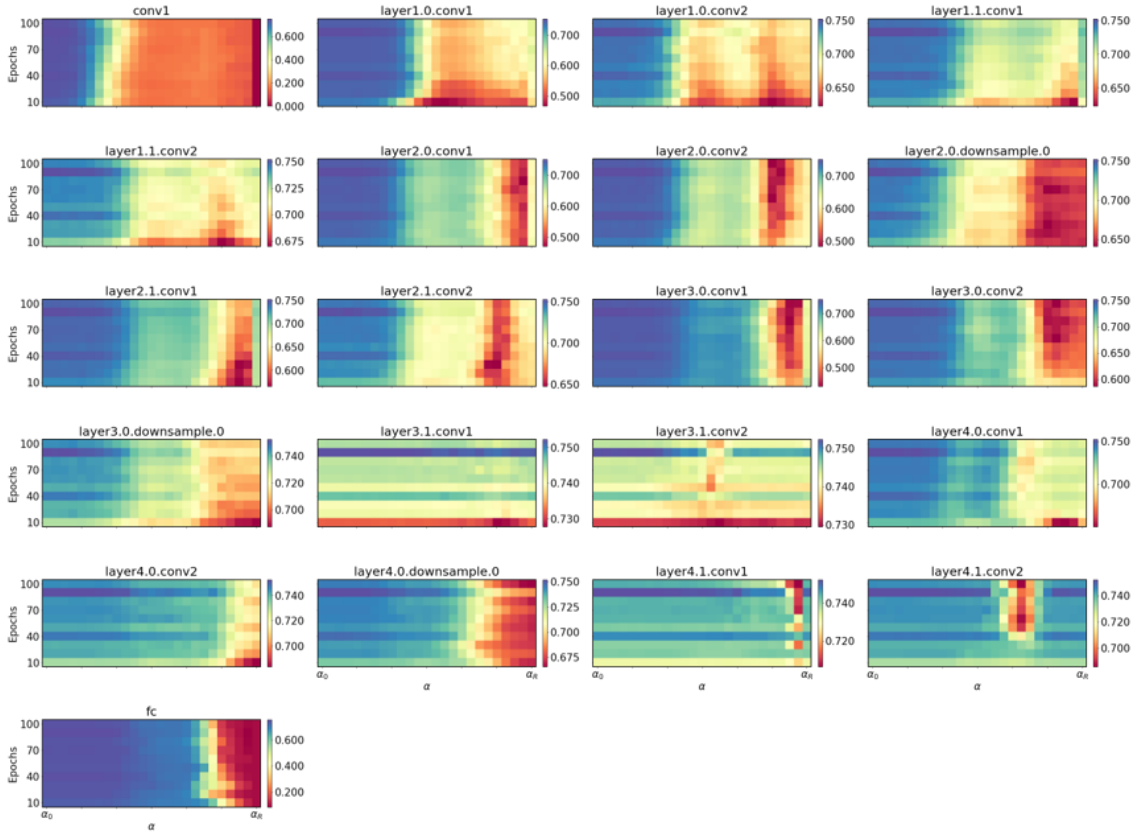


Fig. A.13 ResNet-18 response to clean CIFAR10 dataset, for synaptic filter h_1.

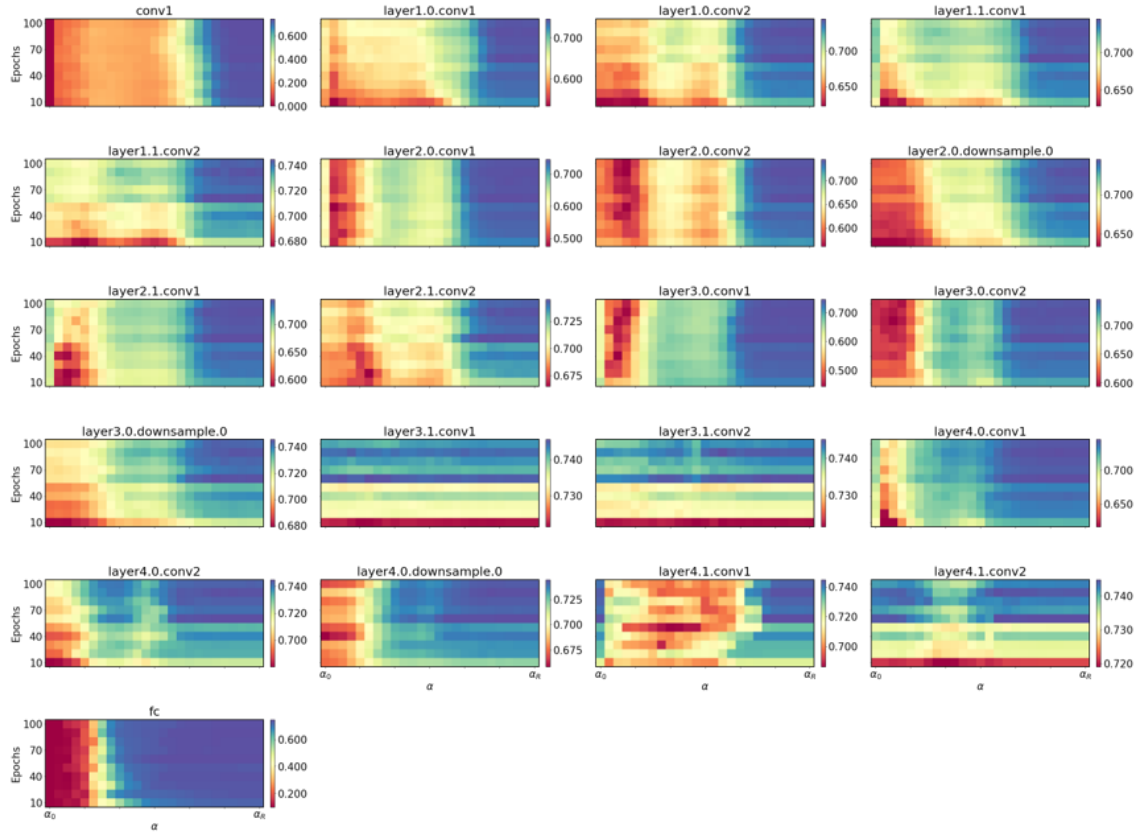


Fig. A.14 ResNet-18 response to clean CIFAR10 dataset, for synaptic filter h_2.

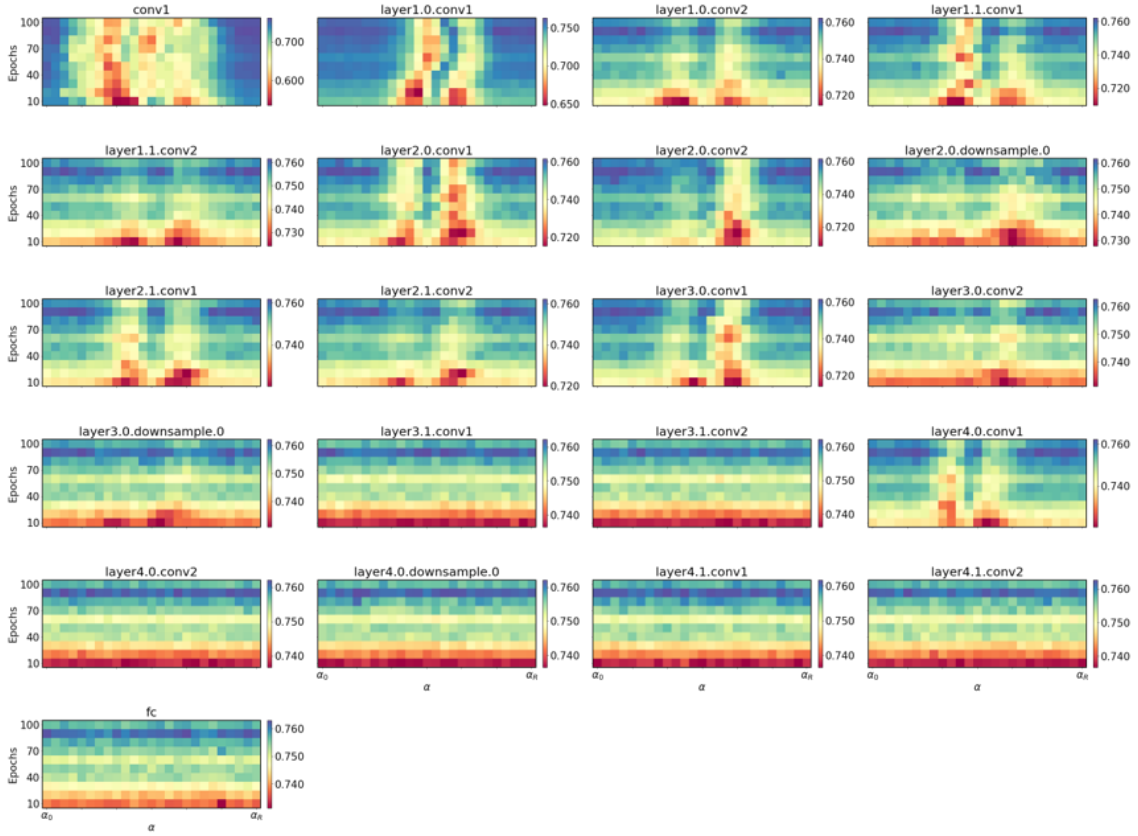


Fig. A.15 ResNet-18 response to clean CIFAR10 dataset, for synaptic filter h_3.

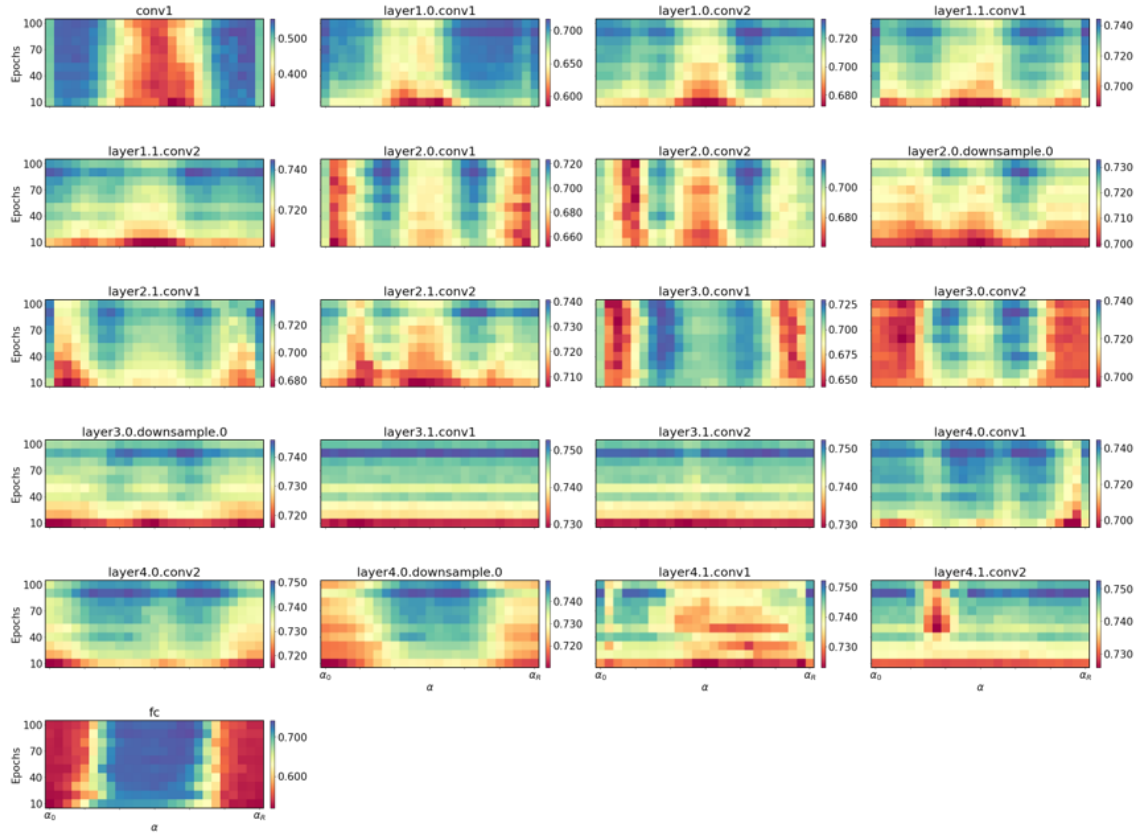


Fig. A.16 Combined ResNet-18 response to clean CIFAR10 dataset, for all synaptic filters in h.

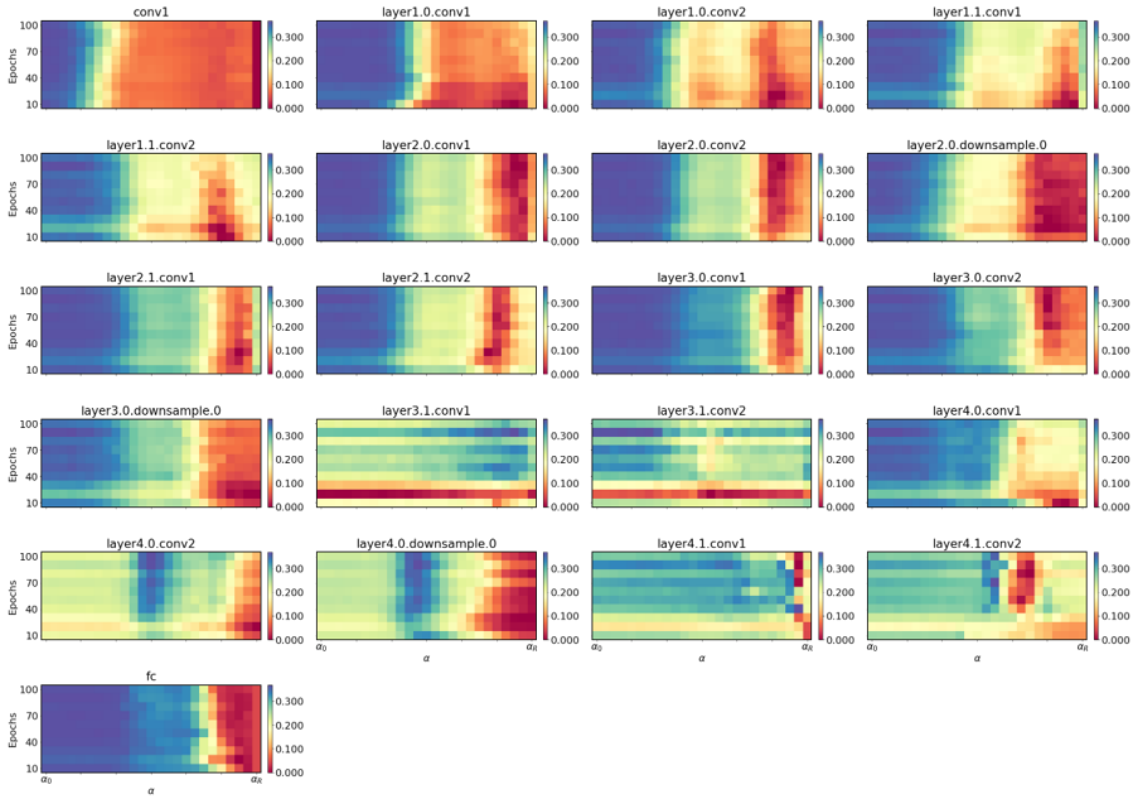


Fig. A.17 ResNet-18 response to adversarial CIFAR10 dataset, for synaptic filter h_1.

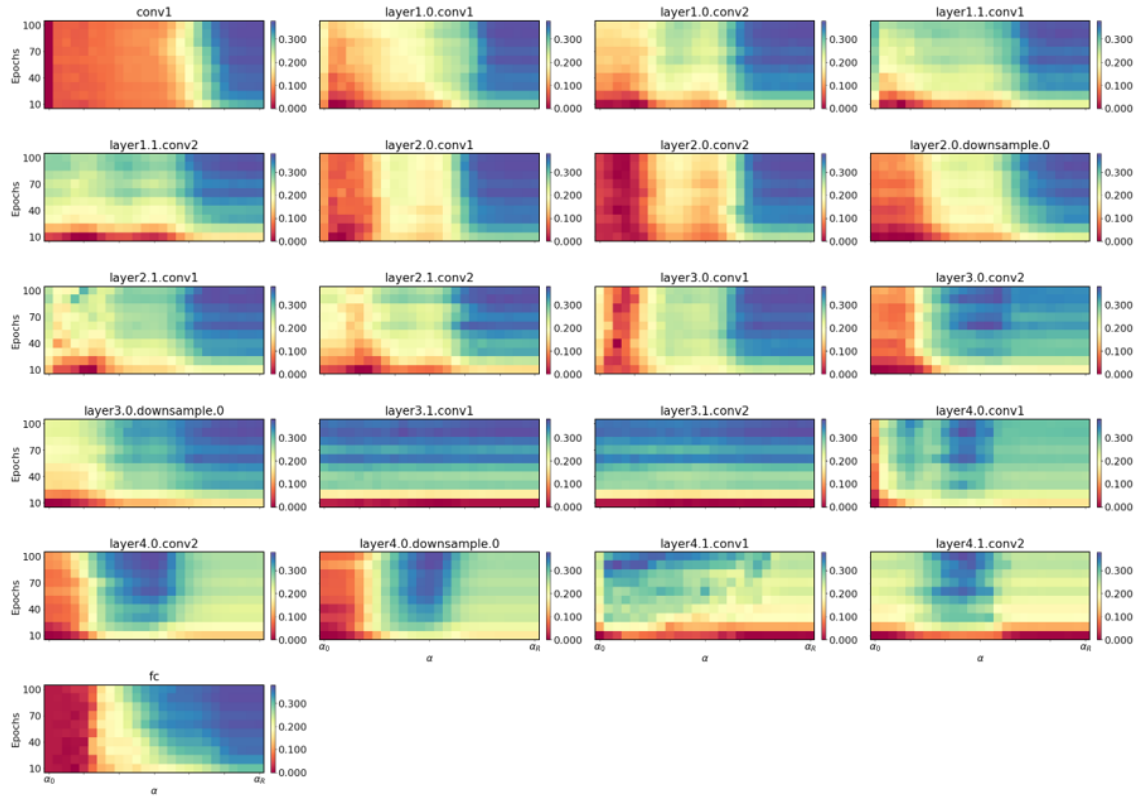


Fig. A.18 ResNet-18 response to adversarial CIFAR10 dataset, for synaptic filter h_2.

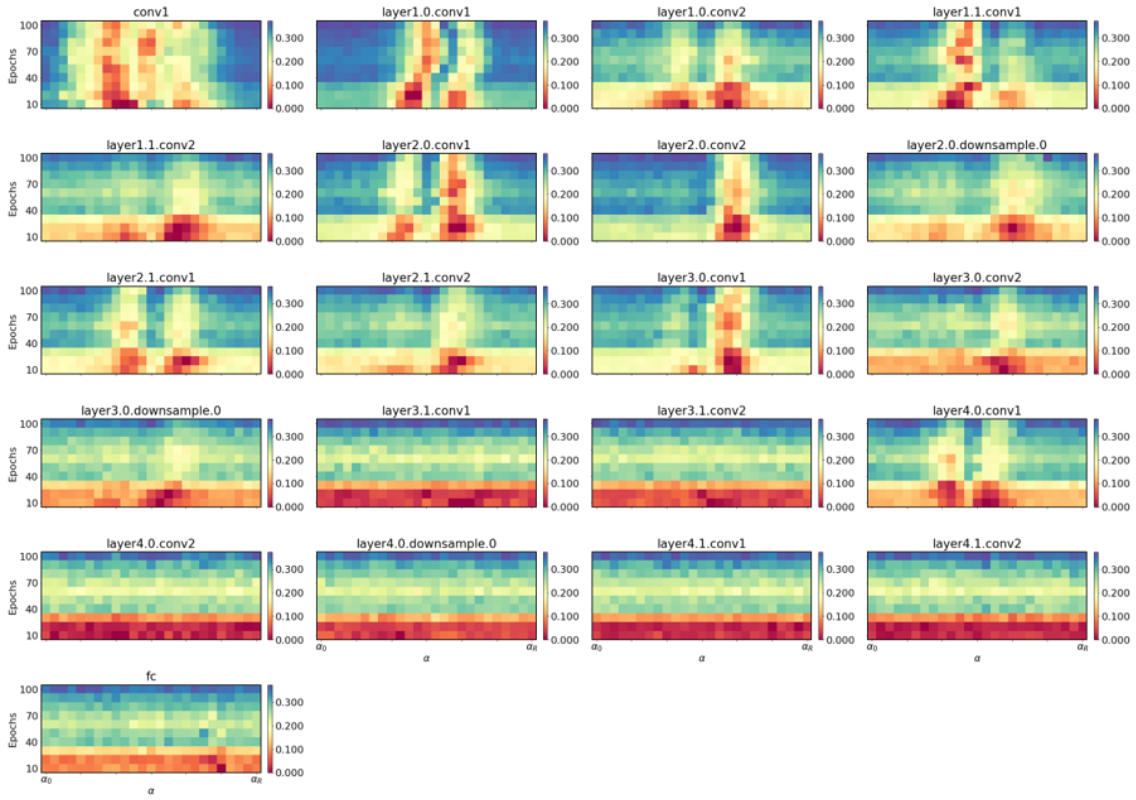


Fig. A.19 ResNet-18 response to adversarial CIFAR10 dataset, for synaptic filter h_3.

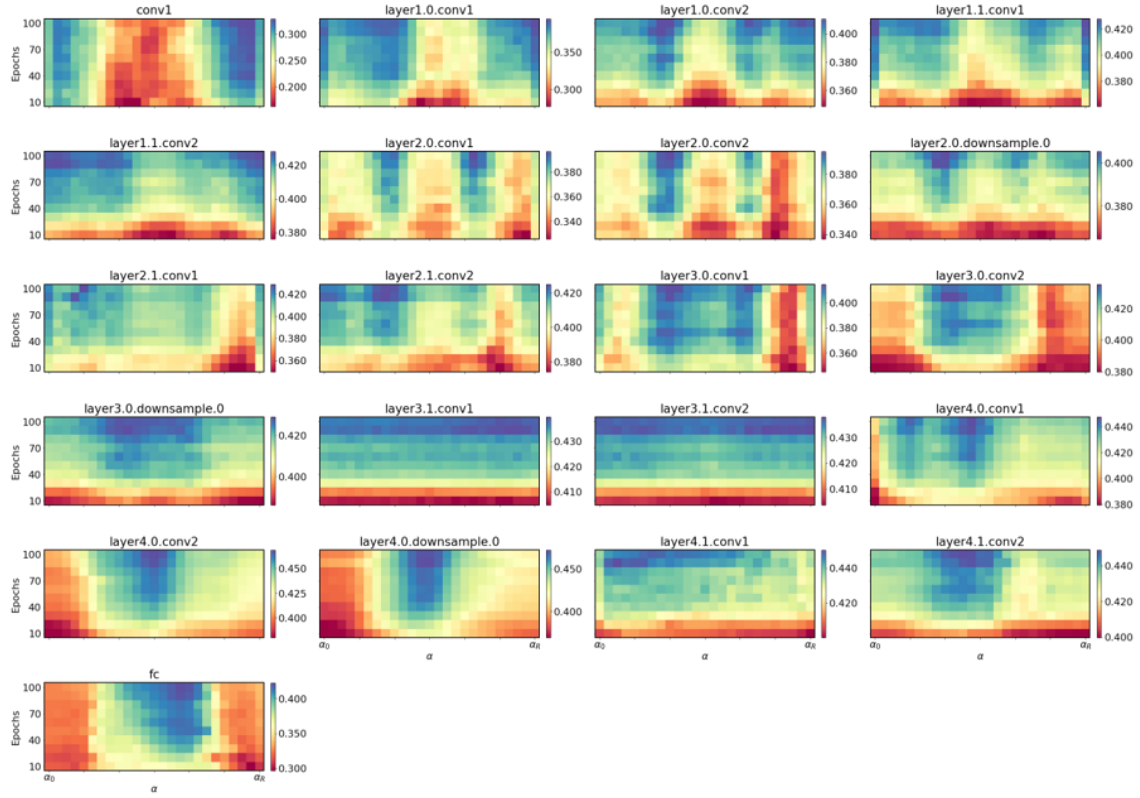


Fig. A.20 Combined ResNet-18 response to adversarial CIFAR10 dataset, for all synaptic filters in h.

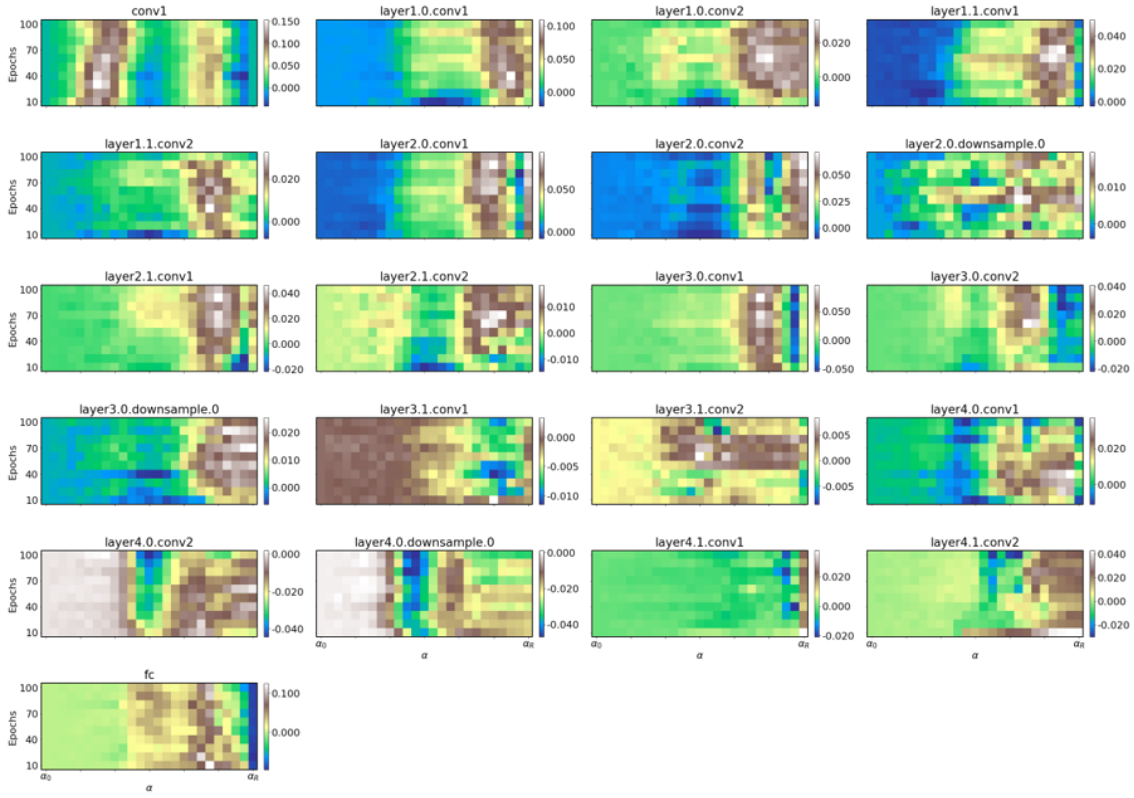


Fig. A.21 Difference in ResNet-18 responses to Clean and adversarial CIFAR10 datasets, for synaptic filter h_1.

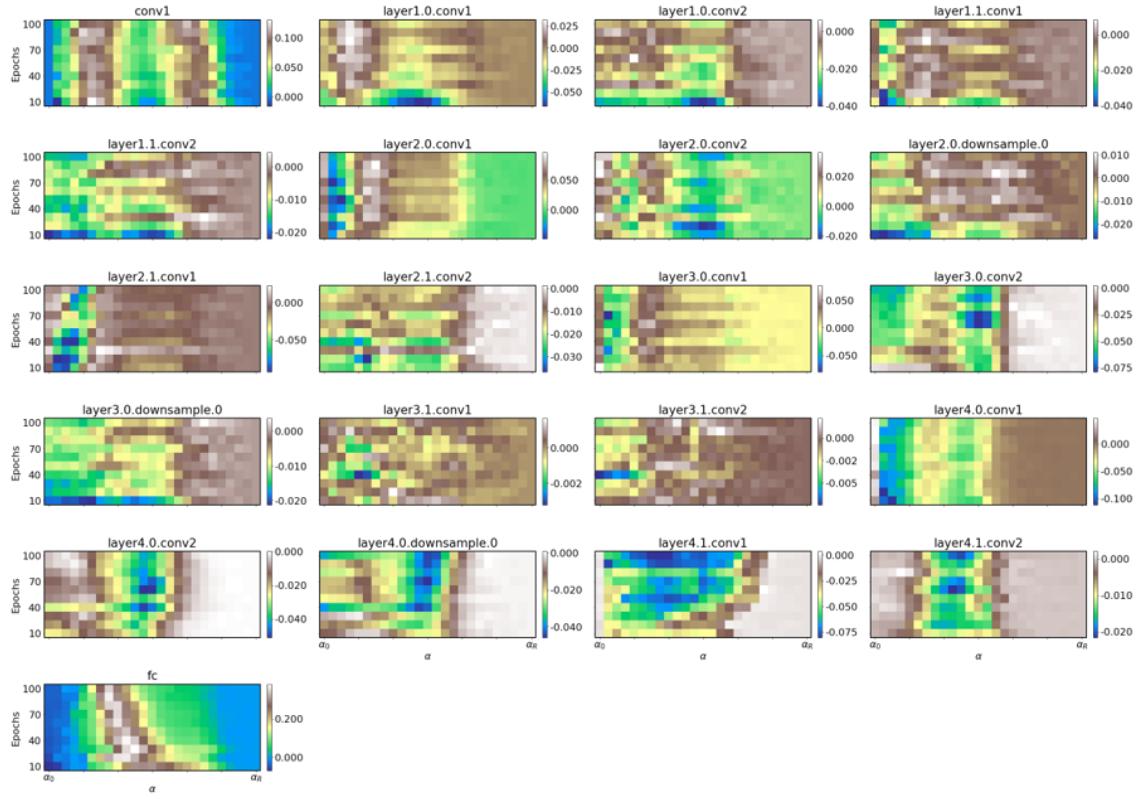


Fig. A.22 Difference in ResNet-18 responses to Clean and adversarial CIFAR10 datasets, for synaptic filter h_2.

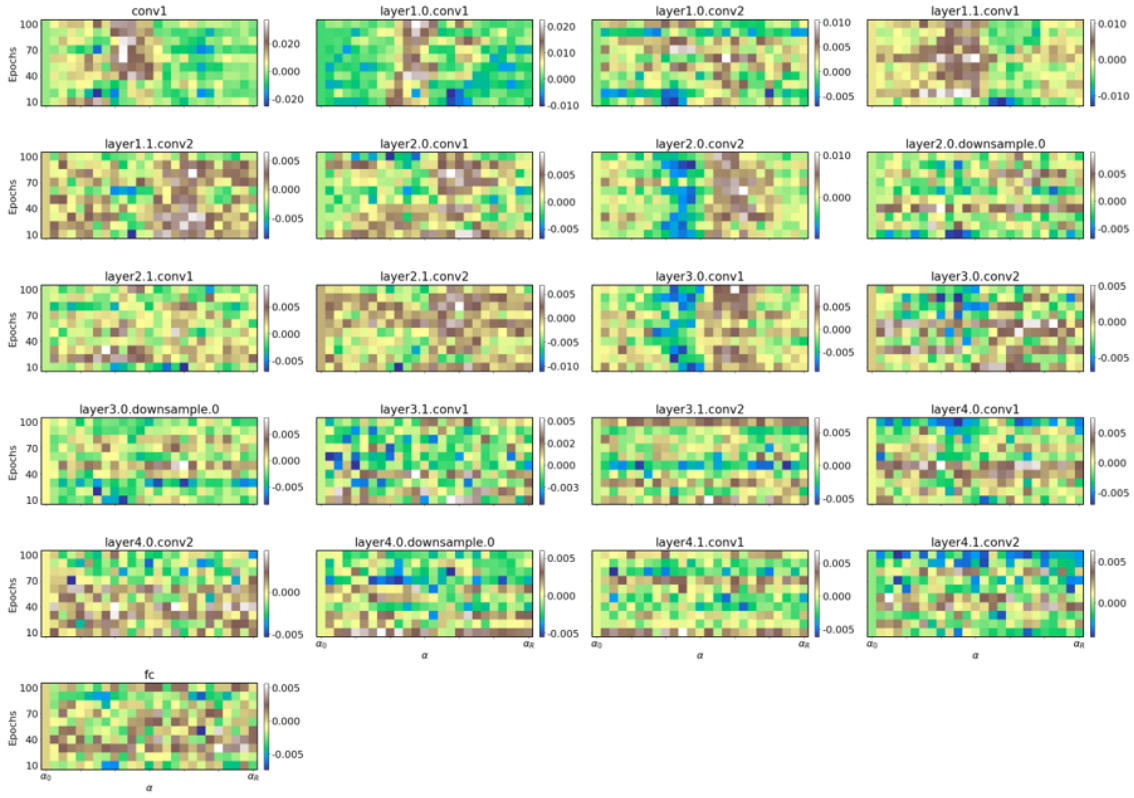


Fig. A.23 Difference in ResNet-18 responses to Clean and adversarial CIFAR10 datasets, for synaptic filter h_3.

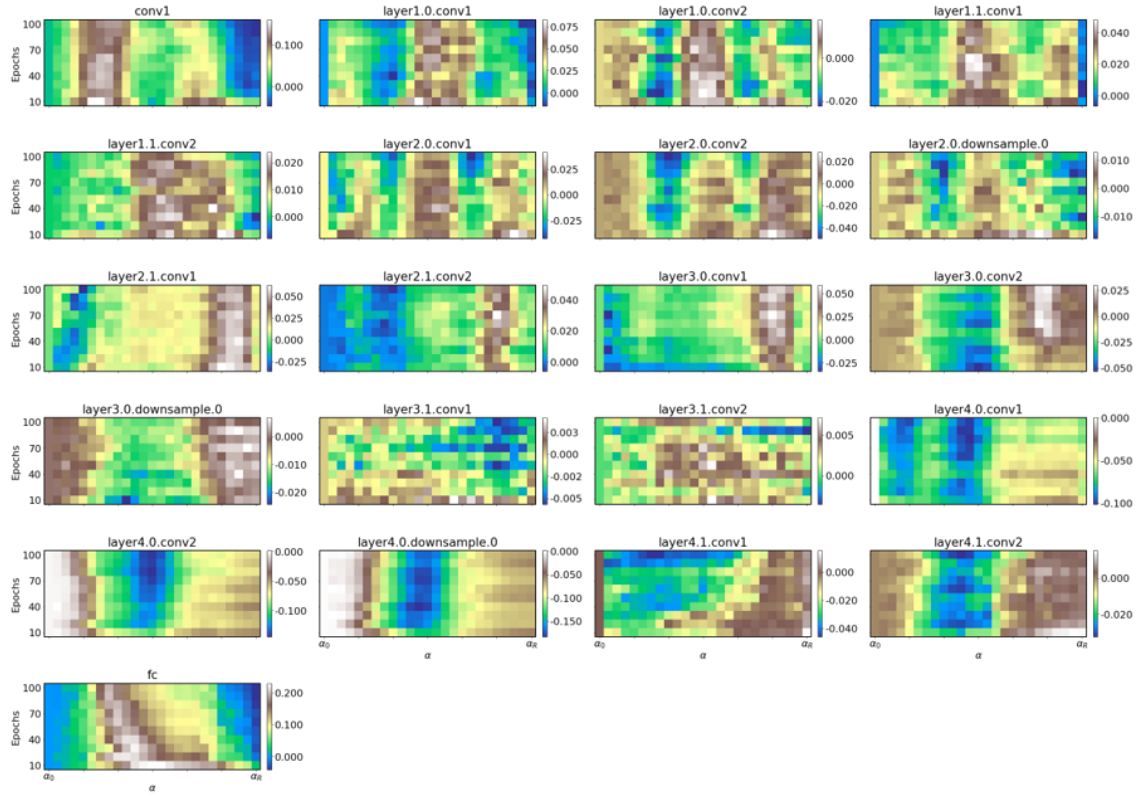


Fig. A.24 Combined difference in ResNet-18 responses to Clean and adversarial CIFAR10 datasets, for all synaptic filters in h.

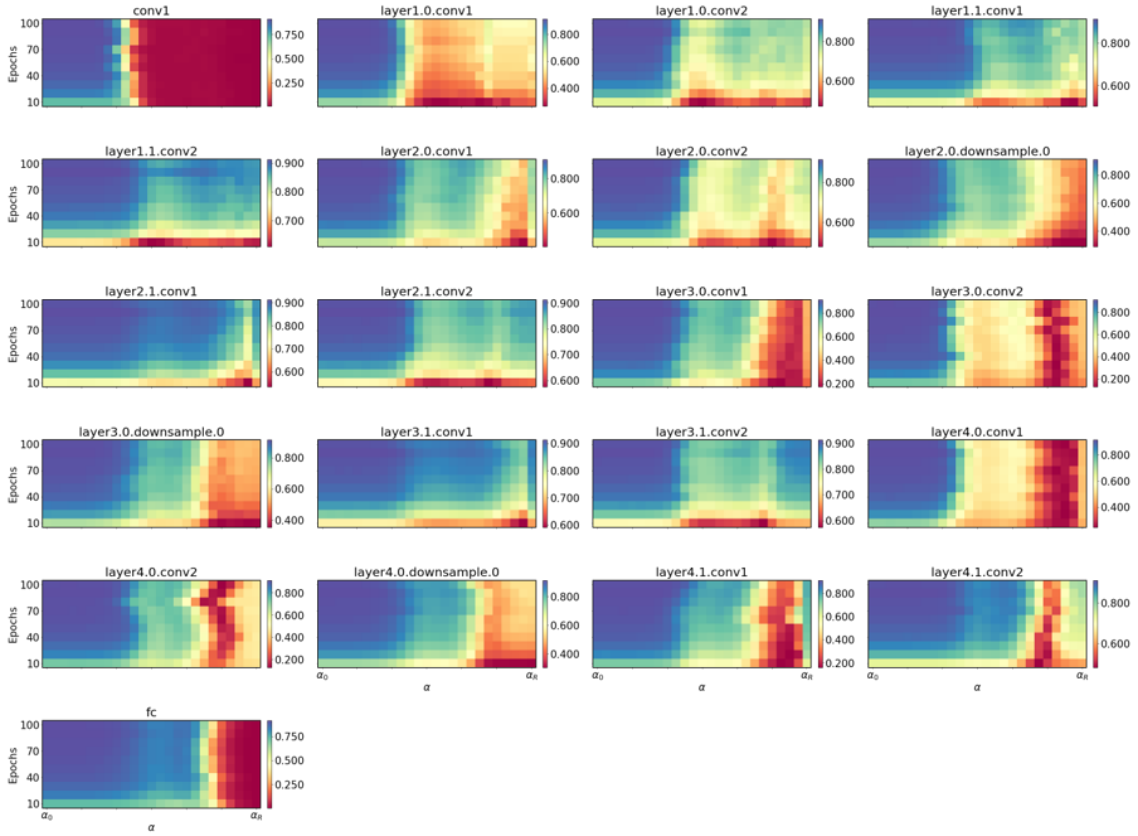


Fig. A.25 ResNet-18 response to clean ImageNet Tiny dataset, for synaptic filter h_1.

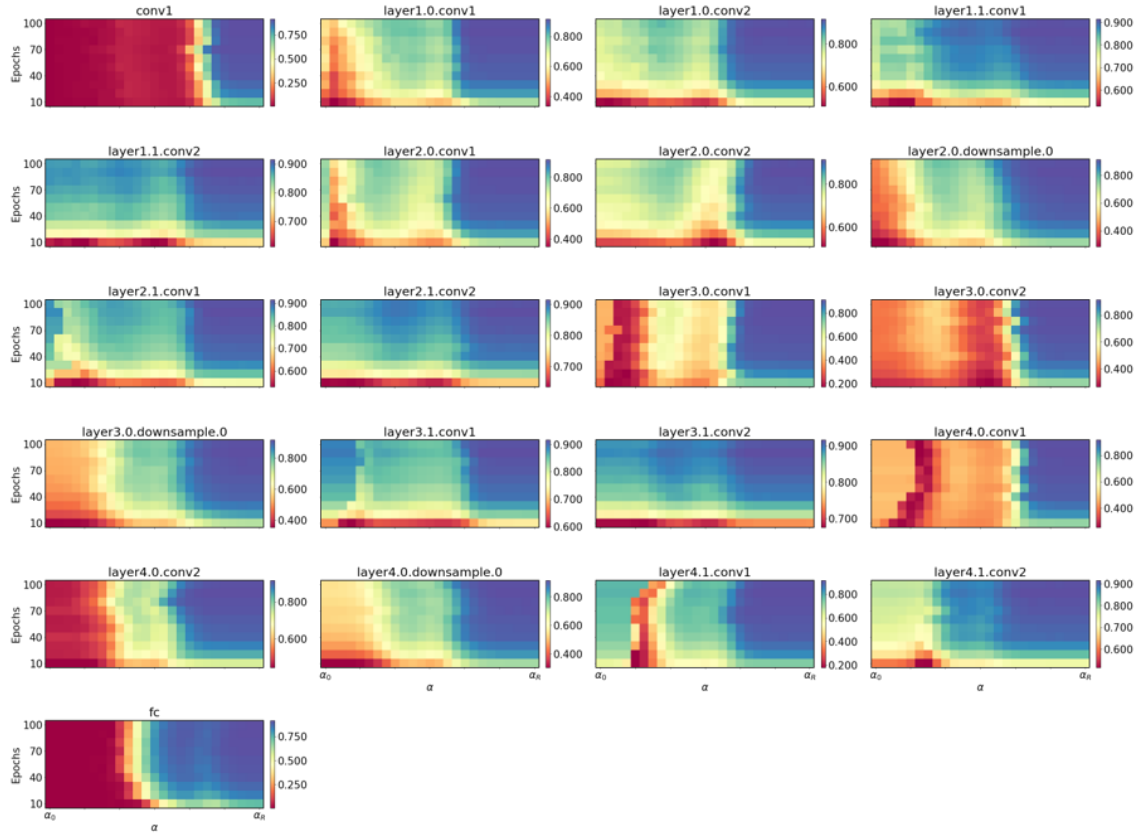


Fig. A.26 ResNet-18 response to clean ImageNet Tiny dataset, for synaptic filter h_2.

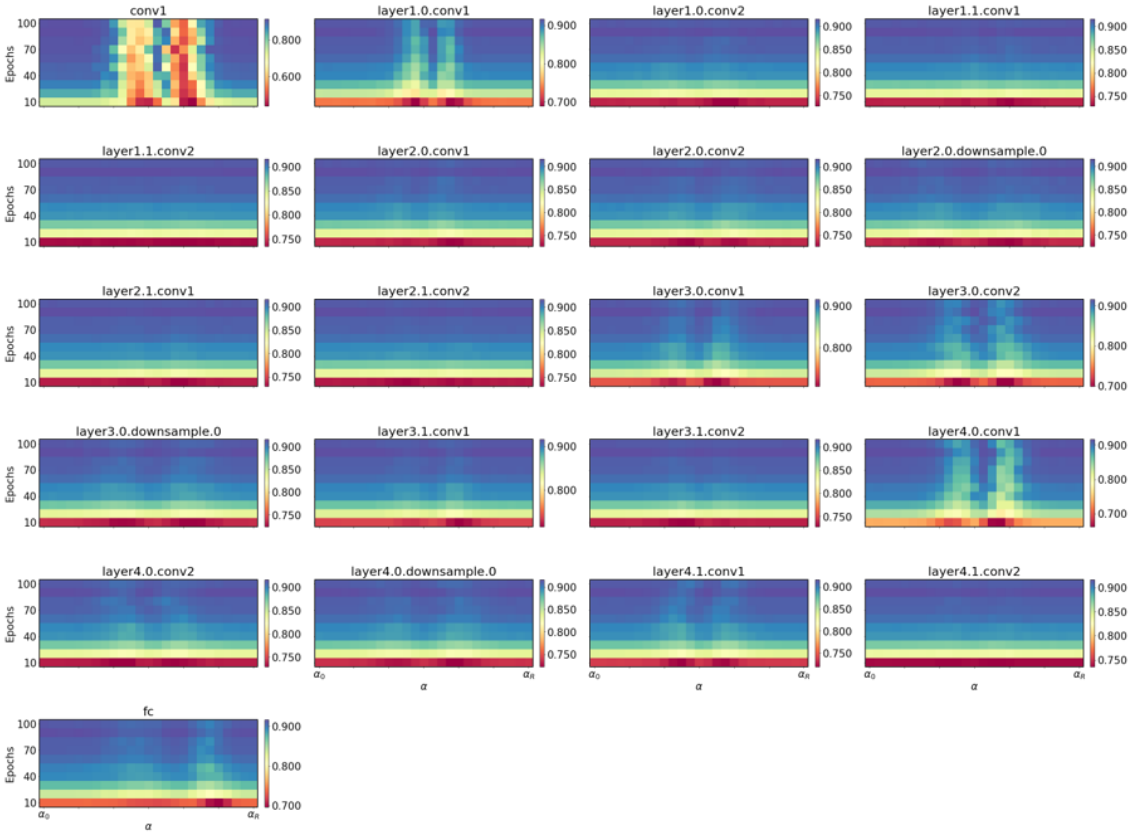


Fig. A.27 ResNet-18 response to clean ImageNet Tiny dataset, for synaptic filter h_3.

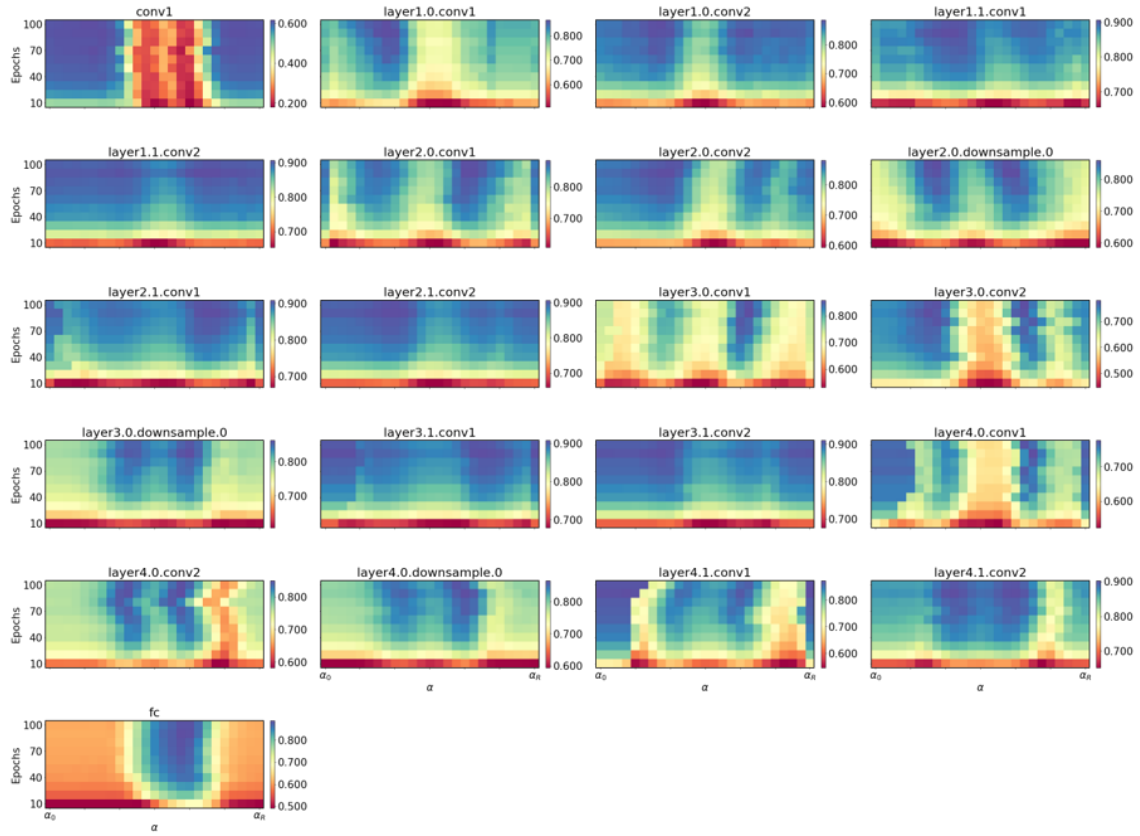


Fig. A.28 Combined ResNet-18 response to clean ImageNet Tiny dataset, for all synaptic filters in h.

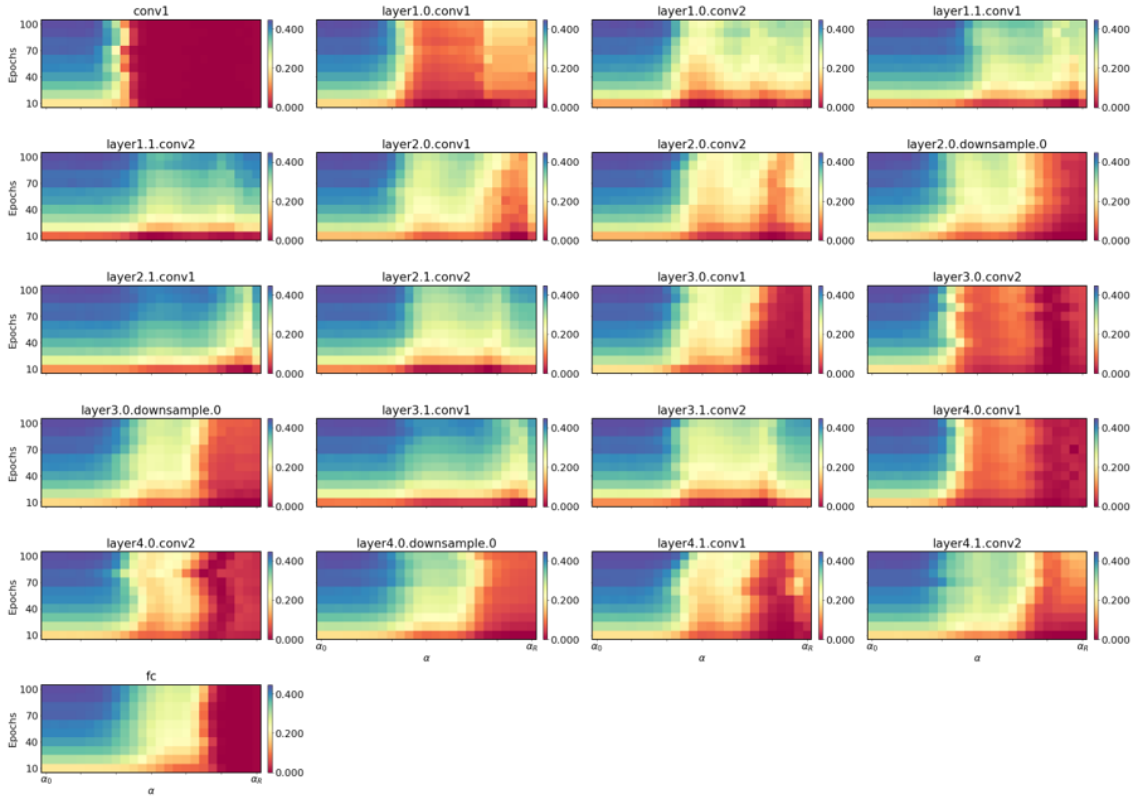


Fig. A.29 ResNet-18 response to adversarial ImageNet Tiny dataset, for synaptic filter h_1.

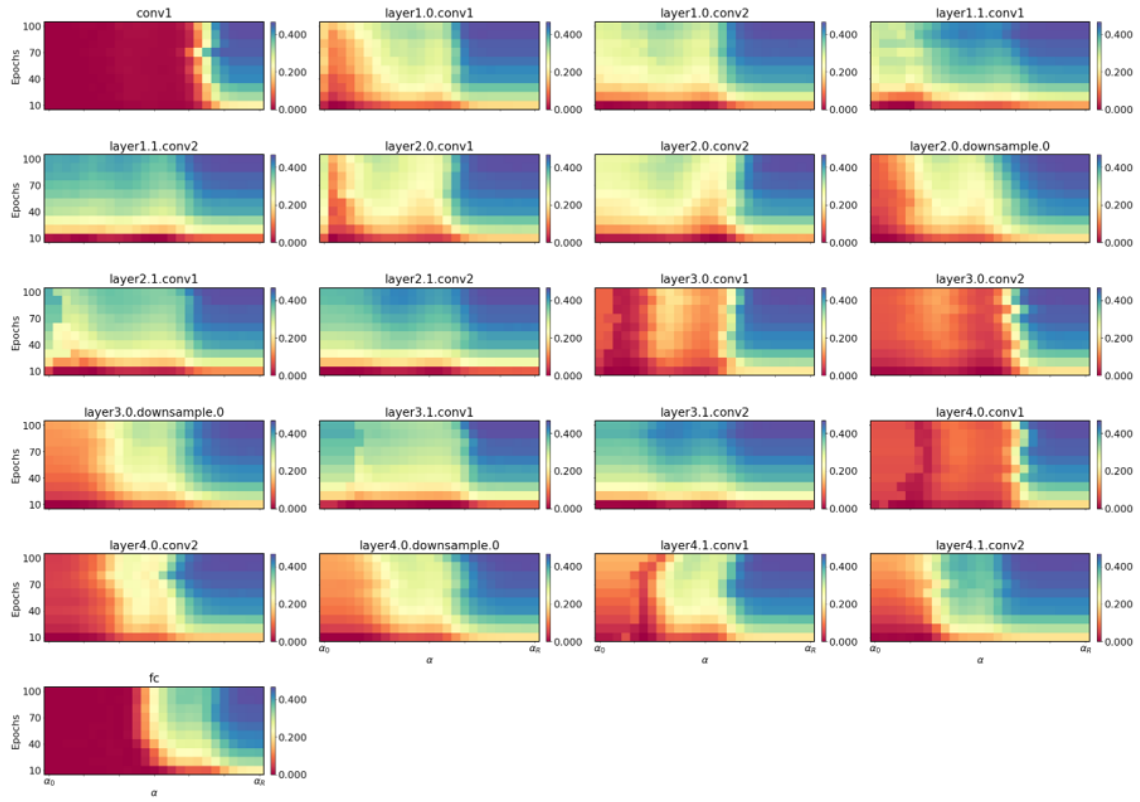


Fig. A.30 ResNet-18 response to adversarial ImageNet Tiny dataset, for synaptic filter h_2.

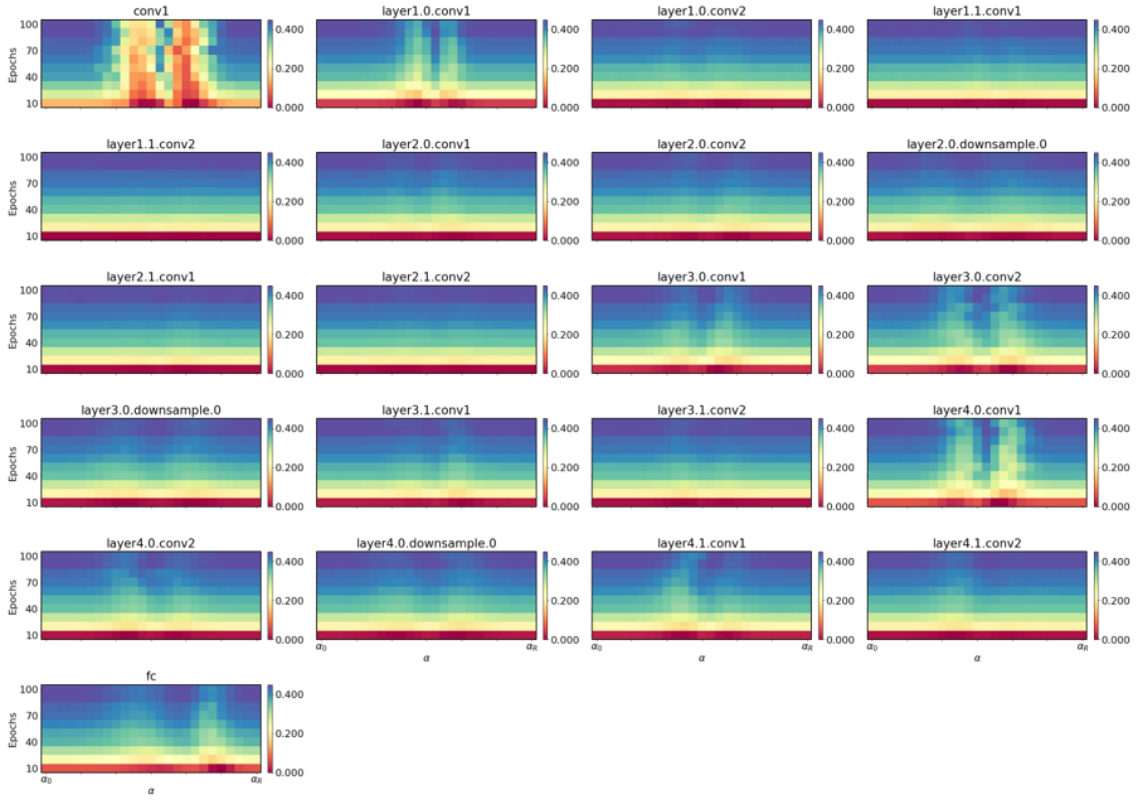


Fig. A.31 ResNet-18 response to adversarial ImageNet Tiny dataset, for synaptic filter h_3.

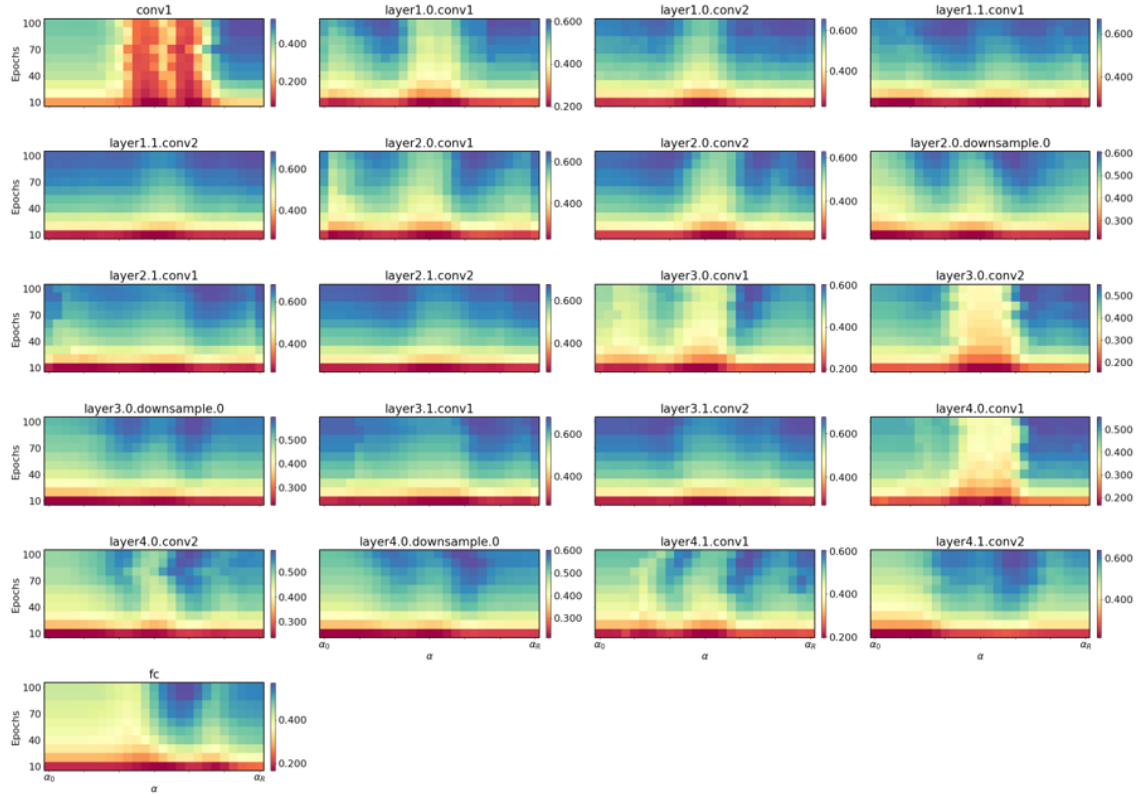


Fig. A.32 Combined ResNet-18 response to adversarial ImageNet Tiny dataset, for all synaptic filters in h.

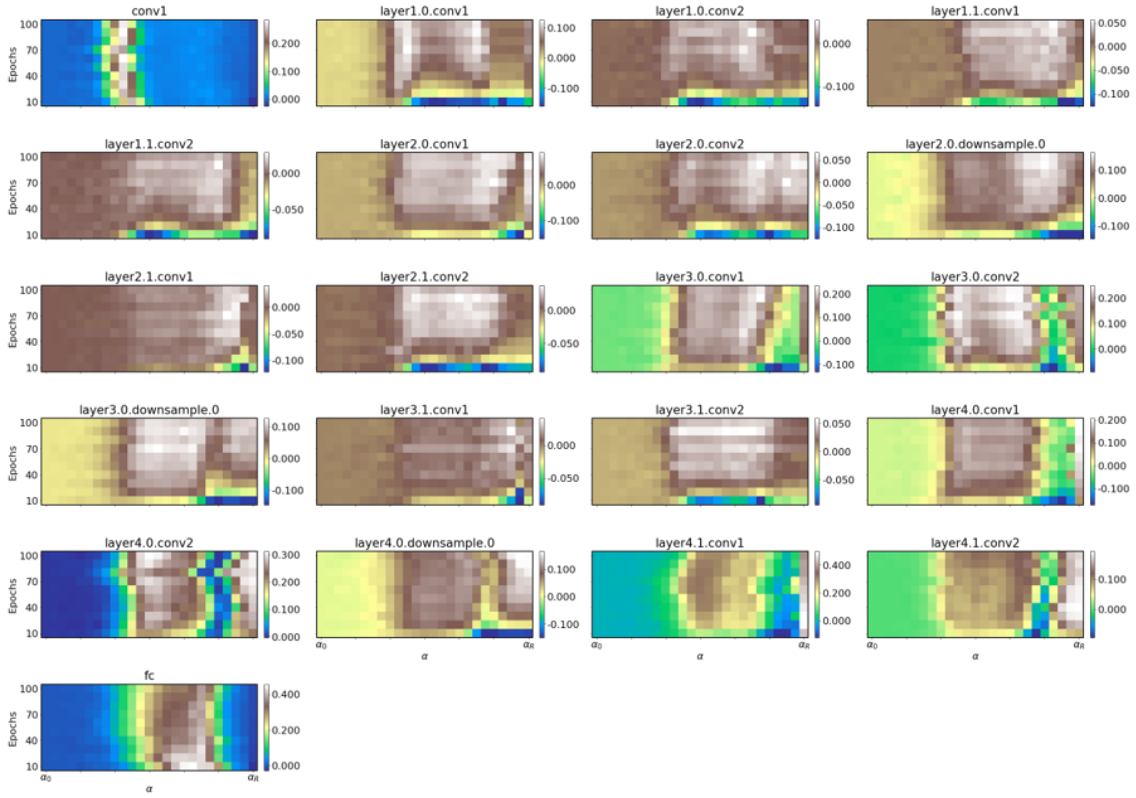


Fig. A.33 Difference in ResNet-18 responses to Clean and adversarial ImageNet Tiny datasets, for synaptic filter h_1.

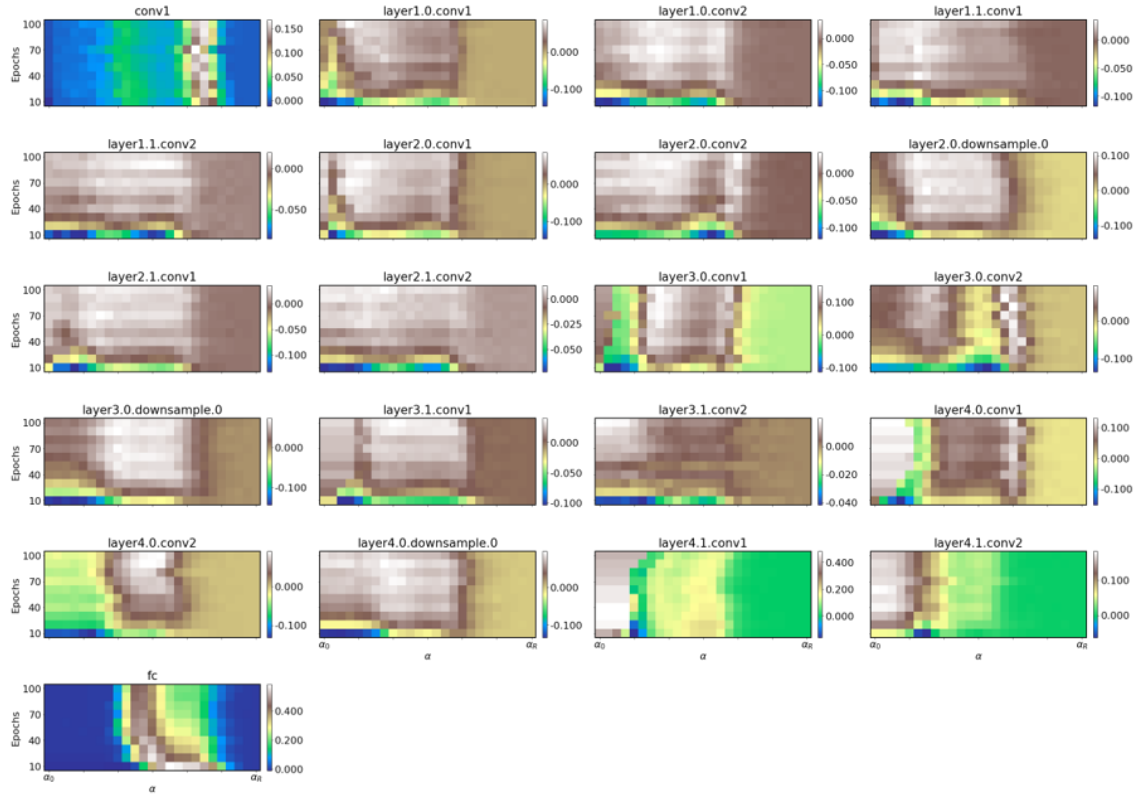


Fig. A.34 Difference in ResNet-18 responses to Clean and adversarial ImageNet Tiny datasets, for synaptic filter h_2.

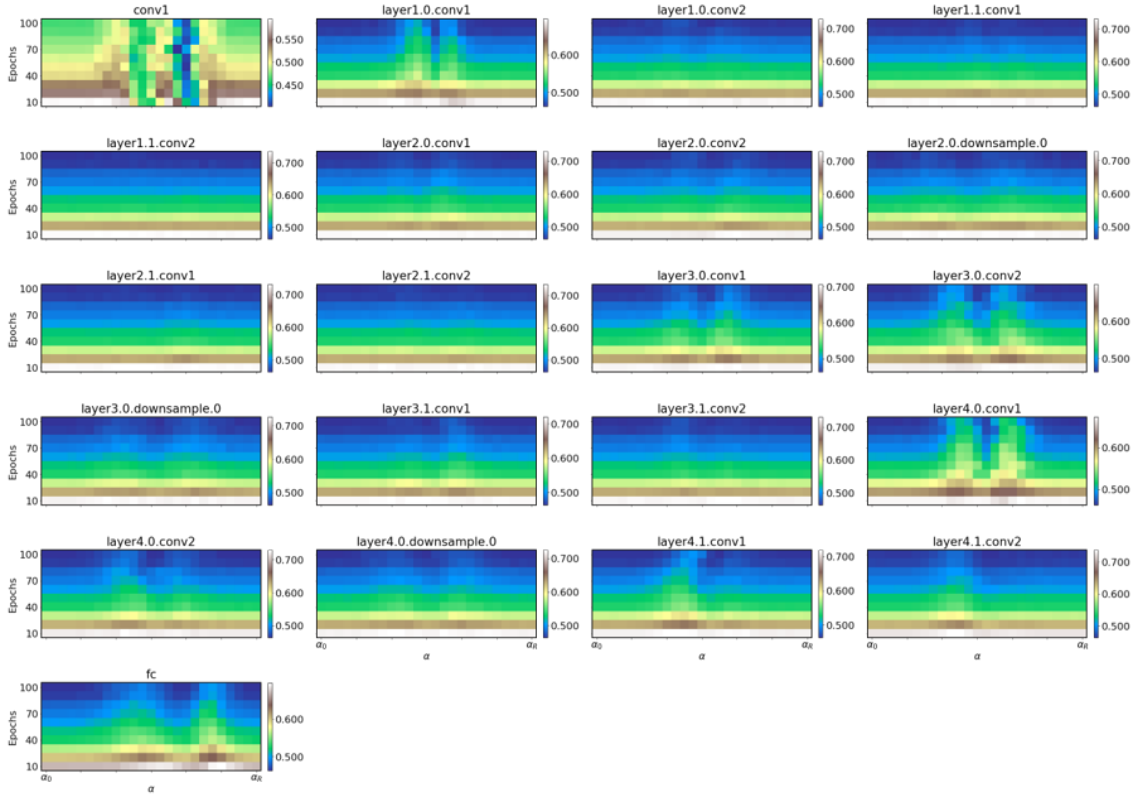


Fig. A.35 Difference in ResNet-18 responses to Clean and adversarial ImageNet Tiny datasets, for synaptic filter h_3.

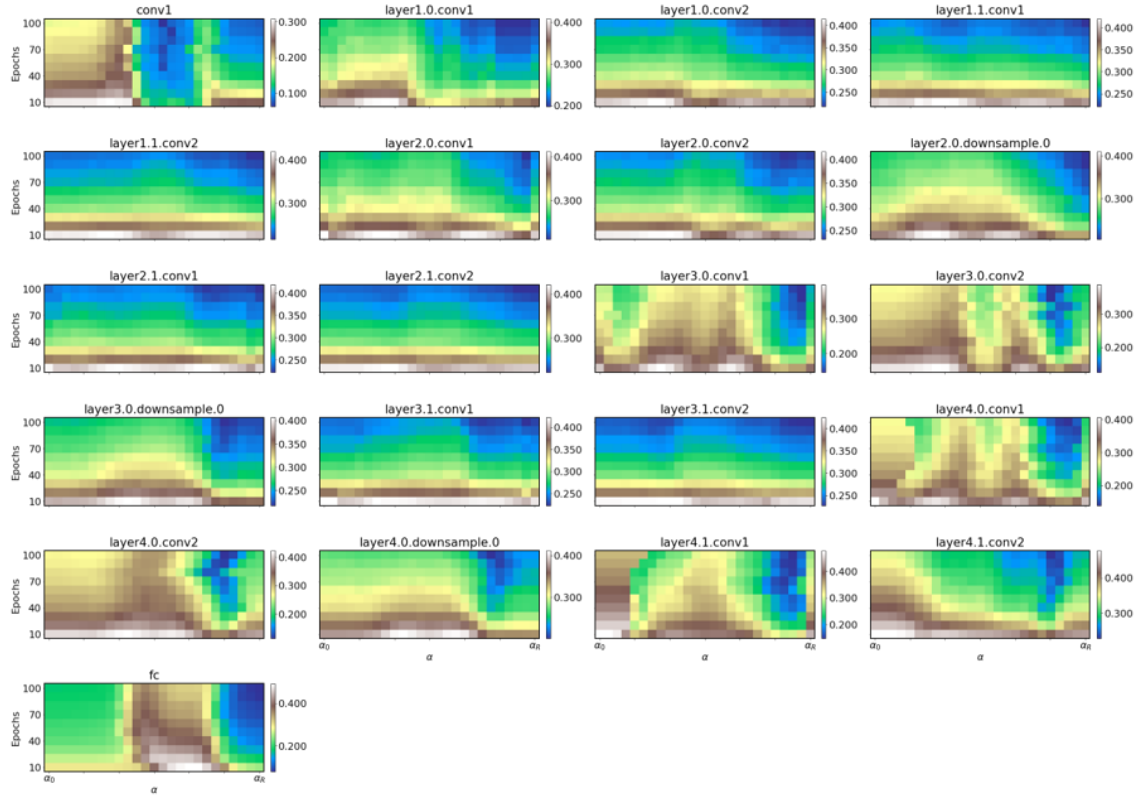


Fig. A.36 Combined difference in ResNet-18 responses to Clean and adversarial ImageNet Tiny datasets, for all synaptic filters in h.

A.2 ResNet-50

A.2.1 Clean Dataset Responses

The ResNet-50 network test accuracy responses to the proposed synaptic filters (See Sec. 3) on the clean, unperturbed, datasets is given for all layers of the network. We present the ResNet-50 responses for synaptic filters h_1, h_2, h_3 and the combined system response (see Sec. 3.3) for the MNIST dataset in Figs. A.37 to A.40. The Pixel intensities in the presented figures show the measured network accuracy for all threshold values α ranging from α_0 to α_A, as per Sec. 3, measured every 10 epochs from 10 to 100 epochs during network training. Similarly, the network test accuracy responses for ResNet-50 on the CIFAR10 dataset, are presented in Figs. A.49 to A.52. The ResNet-50 network test accuracy responses for the ImageNet tiny dataset are presented in Figs. A.61 to A.64. Using the results presented in Figs. A.37 to A.40, Figs. A.49 to A.52, and Figs. A.61 to A.64 we find layers, such as layers '*conv1*', '*layer2.0.conv1*' and '*layer4.0.downsample.0*' for example, where there are invariant response characteristics to different synaptic filters for the three evaluated datasets on the ResNet-50 network.

Adversarial Dataset Responses

The ResNet-50 network test accuracy responses to the different synaptic filters (See Sec. 3) on the adversarial datasets is given for all layers of the network. We present the ResNet-50 responses for synaptic filters h_1, h_2, h_3 and the combined system response (see Sec. 3.3) for the adversarially perturbed MNIST dataset in Figs. A.41 to A.44. The constraints for the perturbation magnitude of the attack is given in Sec. 3. The Pixel intensities in the presented figures show the measured network accuracy for all threshold values α ranging from α_0 to α_A, as per Sec. 3, measured every 10 epochs from 10 to 100 epochs during network training. Similarly, the network test accuracy responses for ResNet-50 on the adversarially perturbed CIFAR10 dataset, are presented in Figs. A.53 to A.56. The ResNet-50 network test accuracy responses, for the adversarially perturbed ImageNet Tiny dataset are presented in Figs. A.65 to A.68. Using the results presented in Figs. A.41 to A.44, Figs. A.53 to A.56, and Figs. A.65 to A.68 we find layers, similar to those highlighted for the clean dataset responses, where there are invariant response characteristics to different synaptic filters for the three evaluated adversarial datasets on the ResNet-50 network. Further analysis of results is given in Sec. 5.

Scaled Response Difference

The ResNet-50 network scaled response differences (see Sec. 3.2) between the clean and adversarial datastes, to the different synaptic filters is given for all layers of the network. We present the ResNet-50 responses difference for synaptic filters h_1, h_2, h_3 and the combined system response (see Sec. 3.3) for the MNIST dataset in Figs. A.45 to A.48. The constraints for the perturbation magnitude of the attack is given in Sec. 3. The Pixel intensities in the presented figures show the measured network accuracy for all threshold values α ranging from α_0 to α_A, as per Sec. 3, measured every 10 epochs from 10 to 100 epochs during network training. The network response differences for ResNet-50 on the CIFAR10 dataset, are presented in Figs. A.57 to A.60. The ResNet-50 network test accuracy response differences, for the ImageNet Tiny dataset are presented in Figs. A.69 to A.72. Using the results presented in Figs. A.45 to A.48, Figs. A.57 to A.60, and Figs. A.69 to A.72 we find links and layers where the adversary is targeting the ResNet-50 network links and layers. We also find layer response differences that show different characteristics for different datasets, further analysis of results is given in Sec. 5.

A.3 SqueezeNet-v1.1

A.3.1 Clean Dataset Responses

The SqueezeNet-v1.1 network test accuracy responses to the ideal high-pass synaptic filter (see Sec. 3) on the clean, unperturbed datasets is given for all layers of the network. We present the SqueezeNet-v1.1 responses to the synaptic filter h_1, for the MNIST dataset in Fig. A.73. The Pixel intensities in the presented figures show the measured network accuracy for all threshold values α ranging from α_0 to α_A, as per Sec. 3, measured every 10 epochs from 10 to 100 epochs during network training. Similarly, the network test accuracy responses for SqueezeNet-v1.1 on the CIFAR10 dataset, are presented in Fig. A.76. The SqueezeNet-v1.1 network test accuracy responses, for the ImageNet tiny dataset are presented in Fig. A.79. Using the results presented in Figs. A.73, Fig. A.73, and Fig. A.79 we find layers, such as layers '*features.3.squeeze*', '*features.9.squeeze*' and '*features.11.squeeze*' for example, where there are invariant response characteristics to different synaptic filters for the three evaluated datasets on the SqueezeNet-v1.1 network.

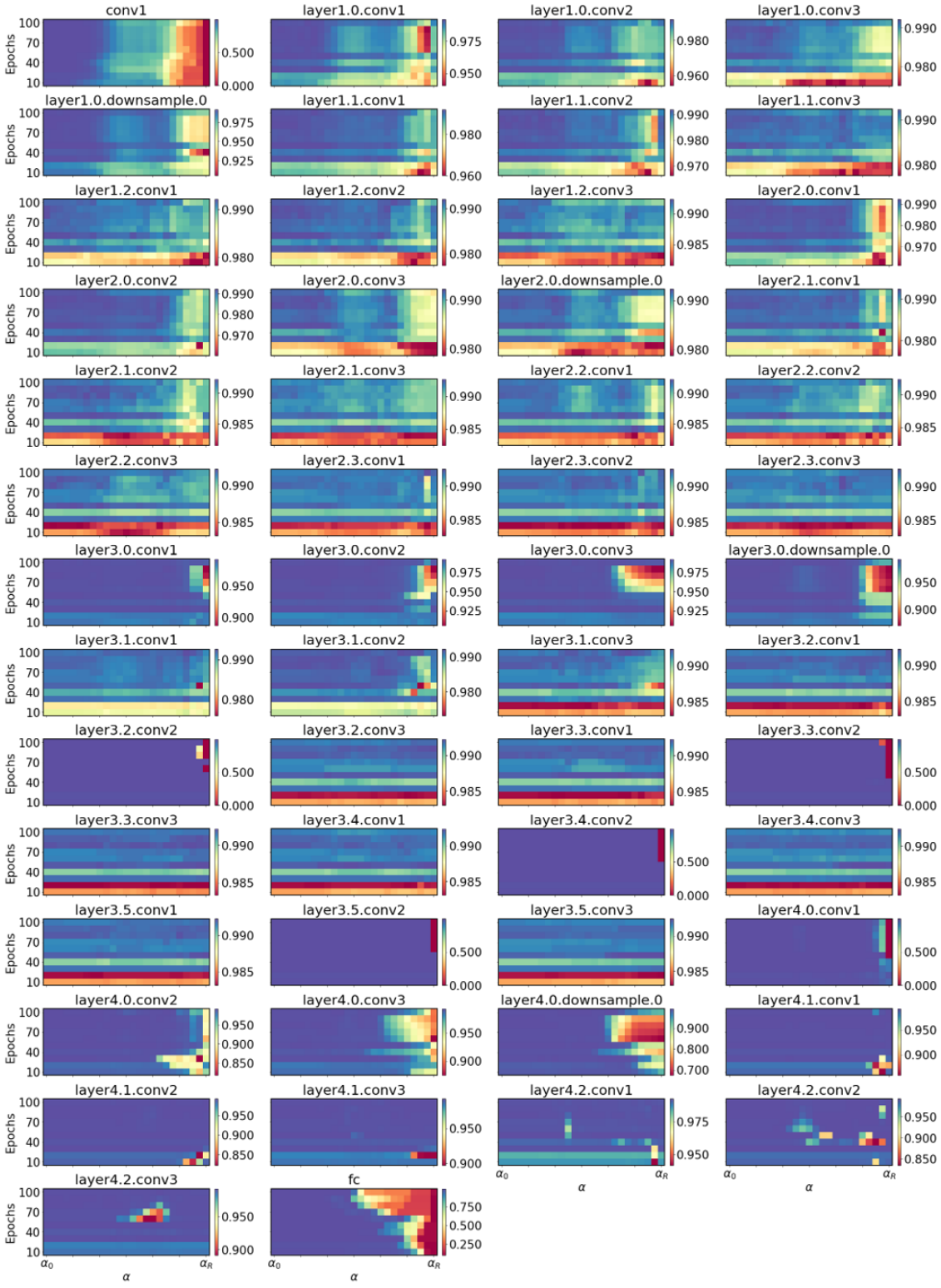


Fig. A.37 ResNet-50 response to clean MNIST dataset, for synaptic filter h_1.

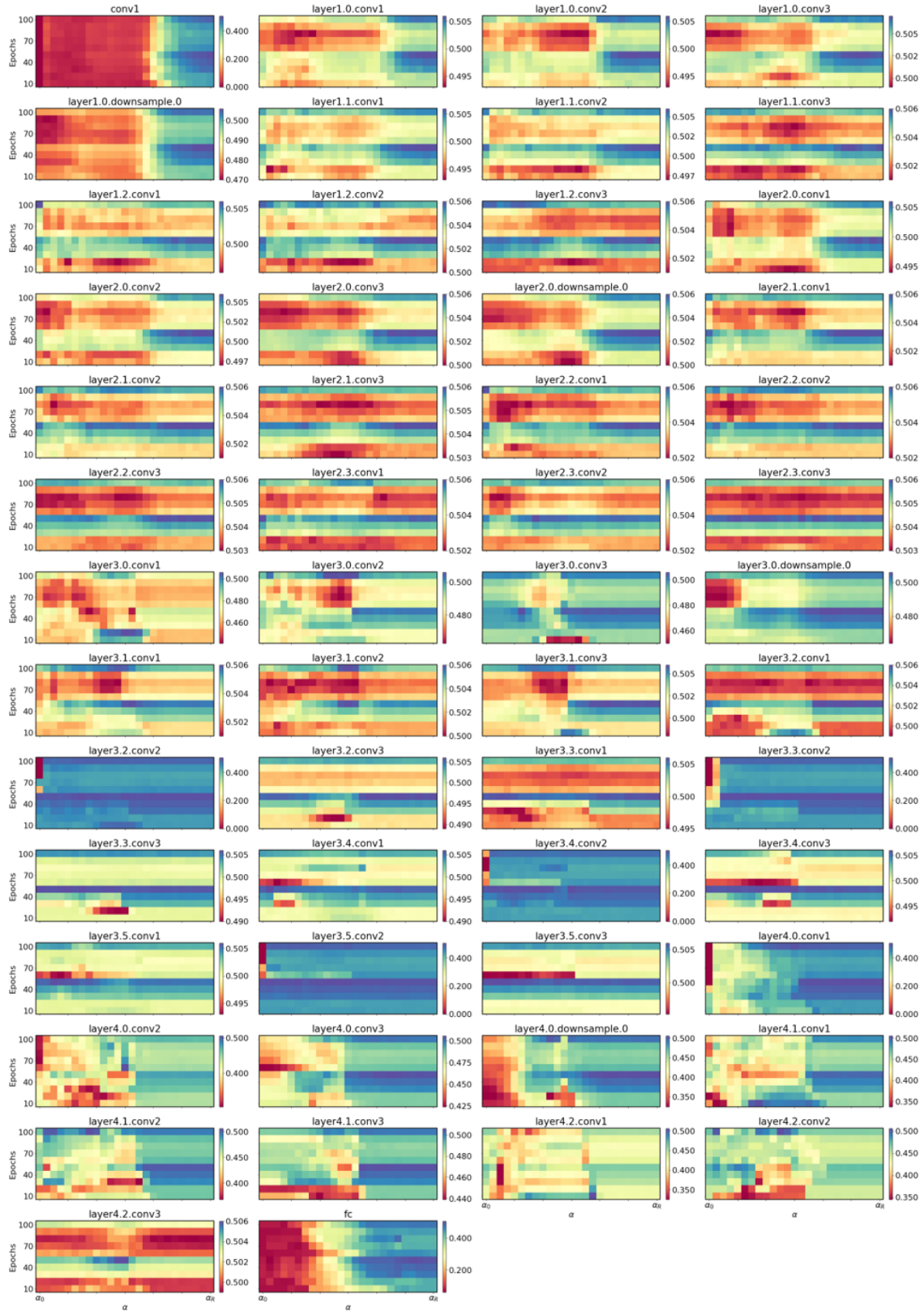


Fig. A.38 ResNet-50 response to clean MNIST dataset, for synaptic filter h_2.

Fig. A.39 ResNet-50 response to clean MNIST dataset, for synaptic filter h_3.

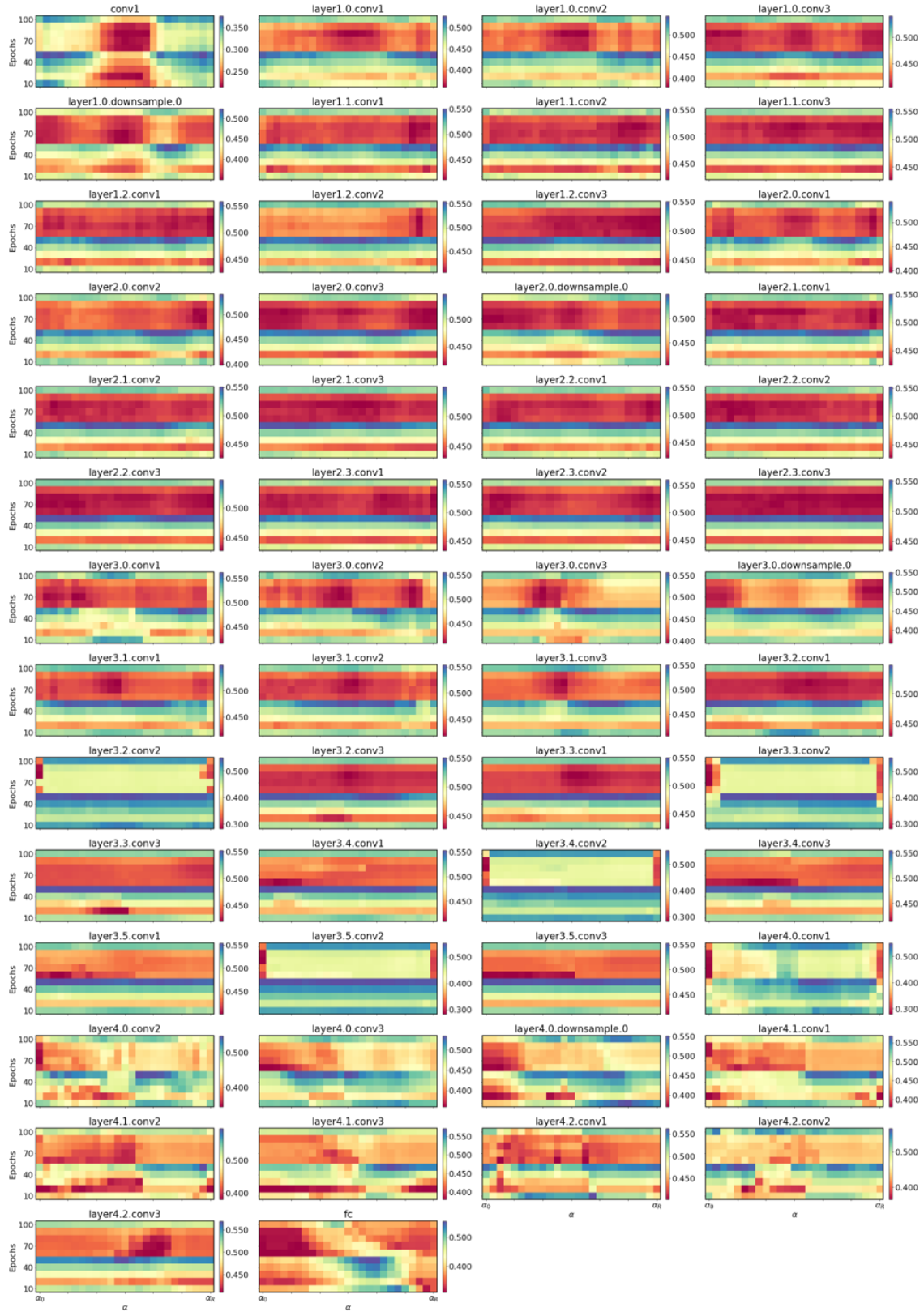


Fig. A.40 Combined ResNet-50 response to clean MNIST dataset, for all synaptic filters in h.

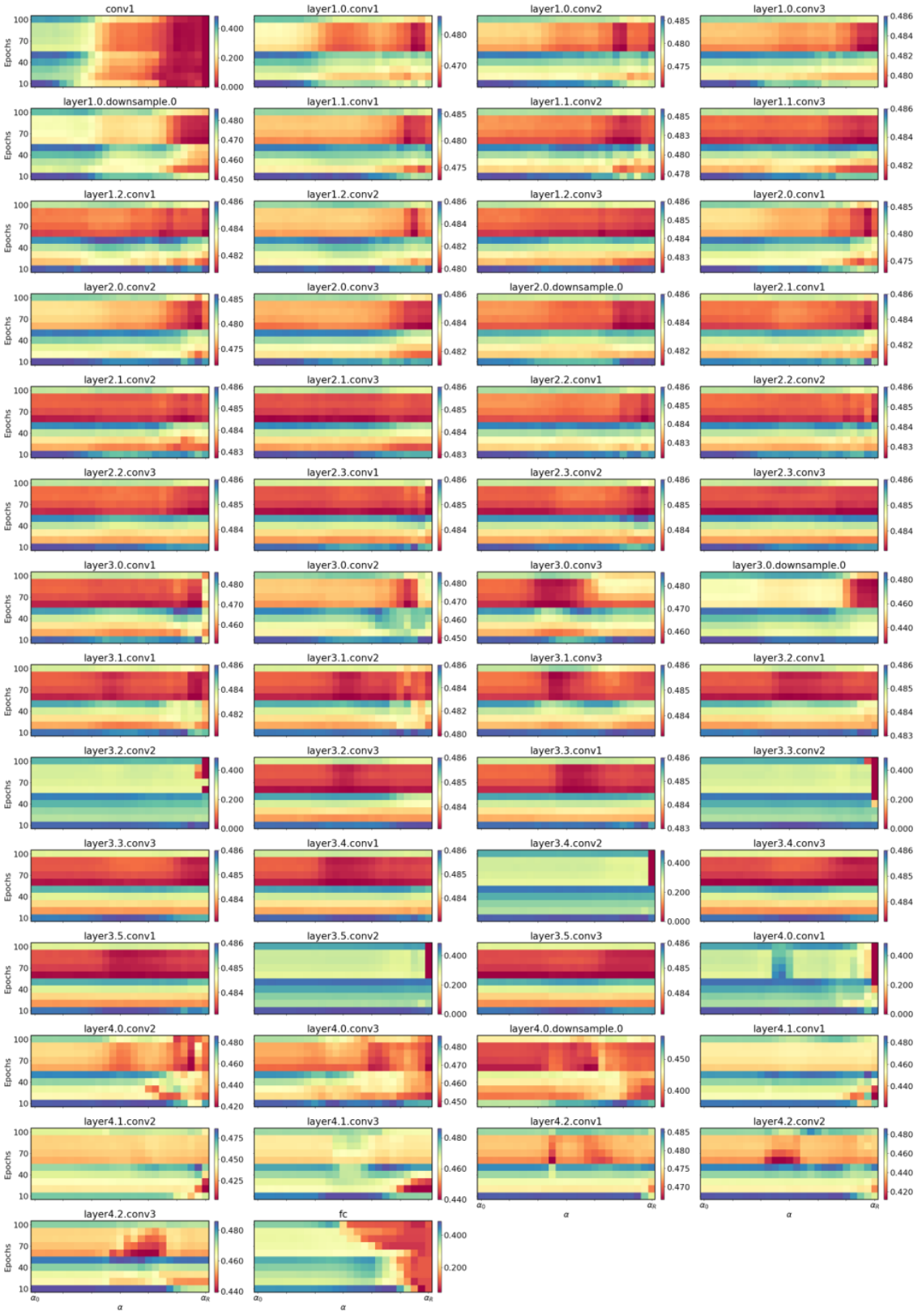


Fig. A.41 ResNet-50 response to adversarial MNIST dataset, for synaptic filter h_1.

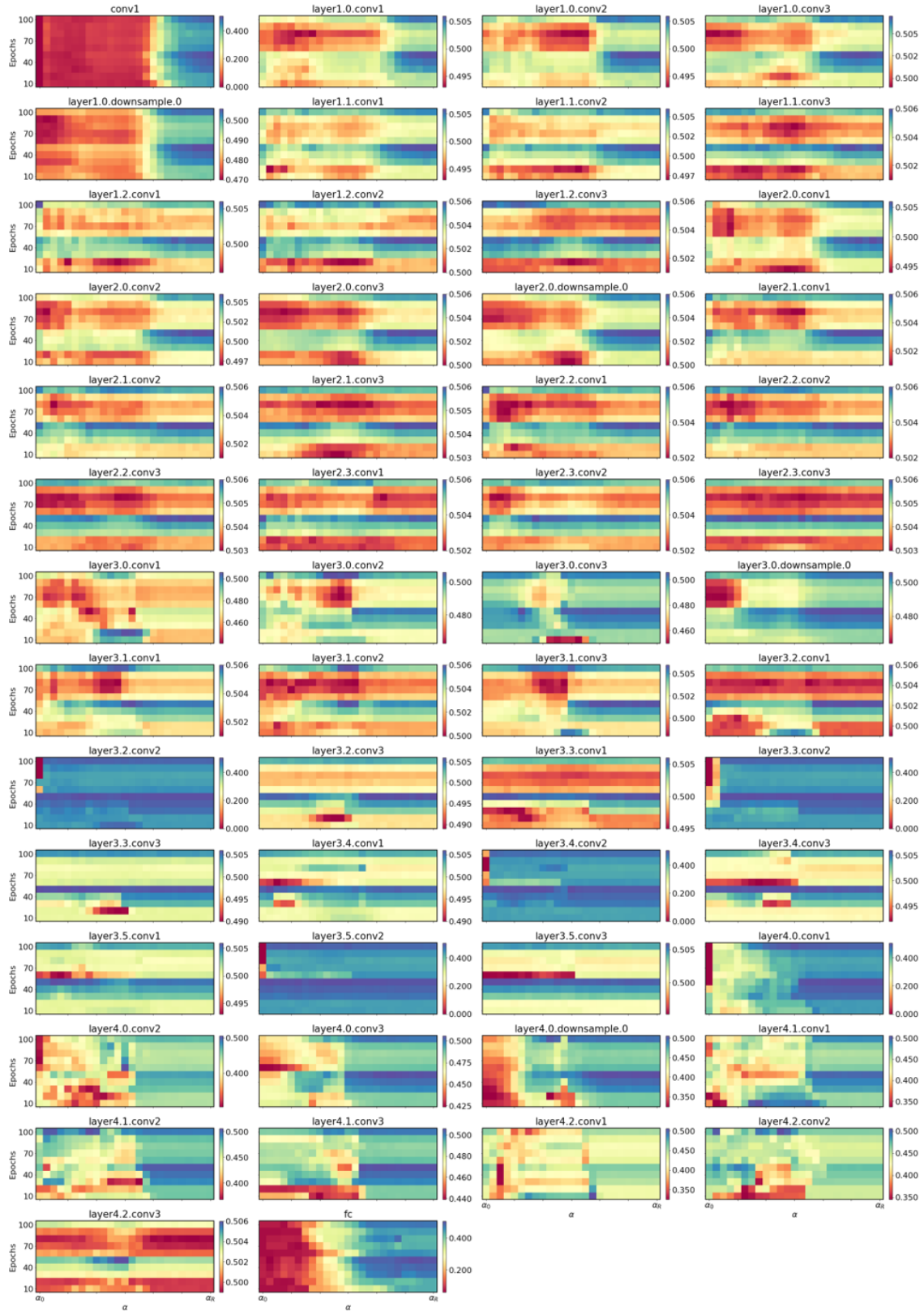


Fig. A.42 ResNet-50 response to adversarial MNIST dataset, for synaptic filter h_2.

Fig. A.43 ResNet-50 response to adversarial MNIST dataset, for synaptic filter h_3.

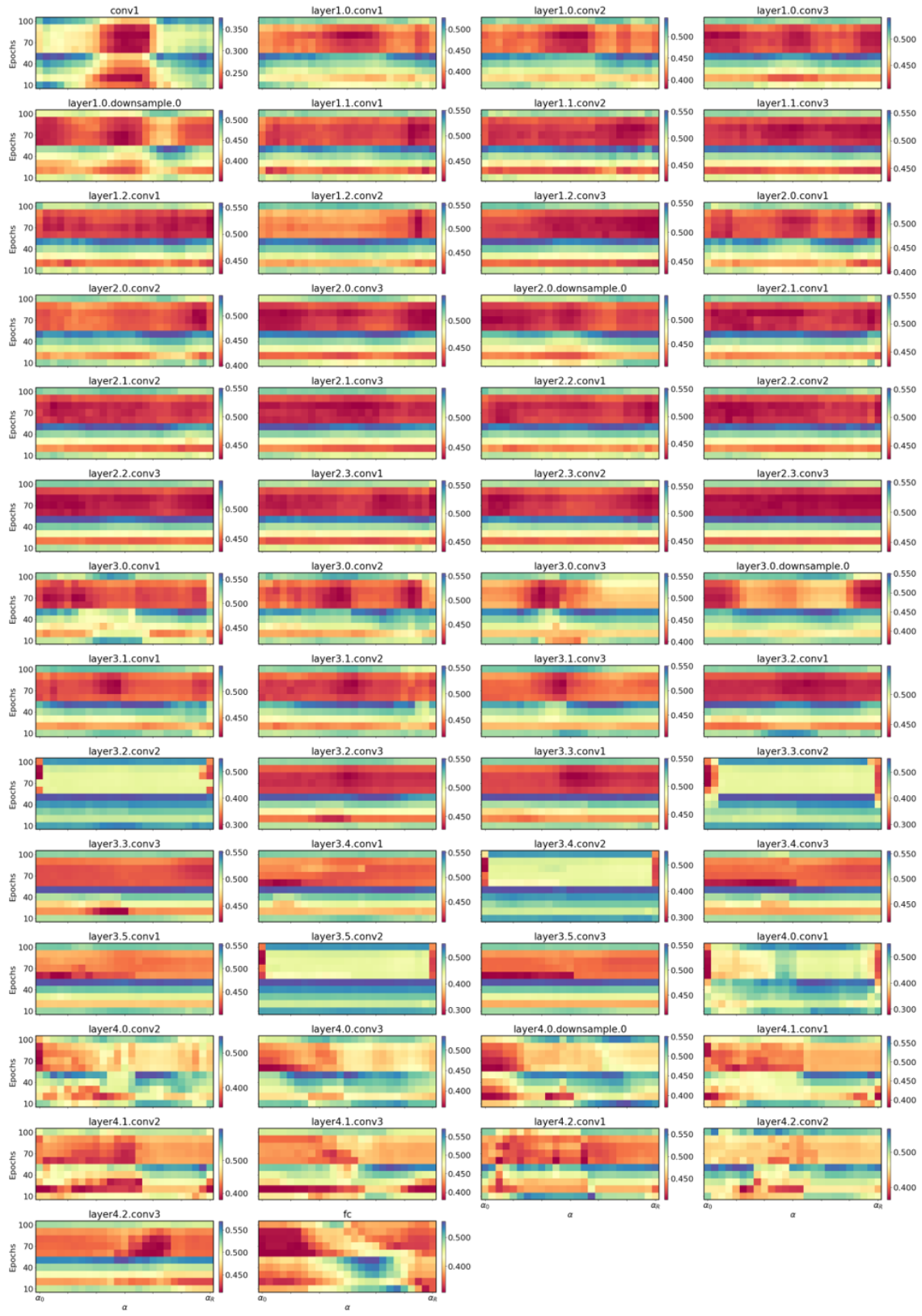


Fig. A.44 Combined ResNet-50 response to adversarial MNIST dataset, for all synaptic filters in h.

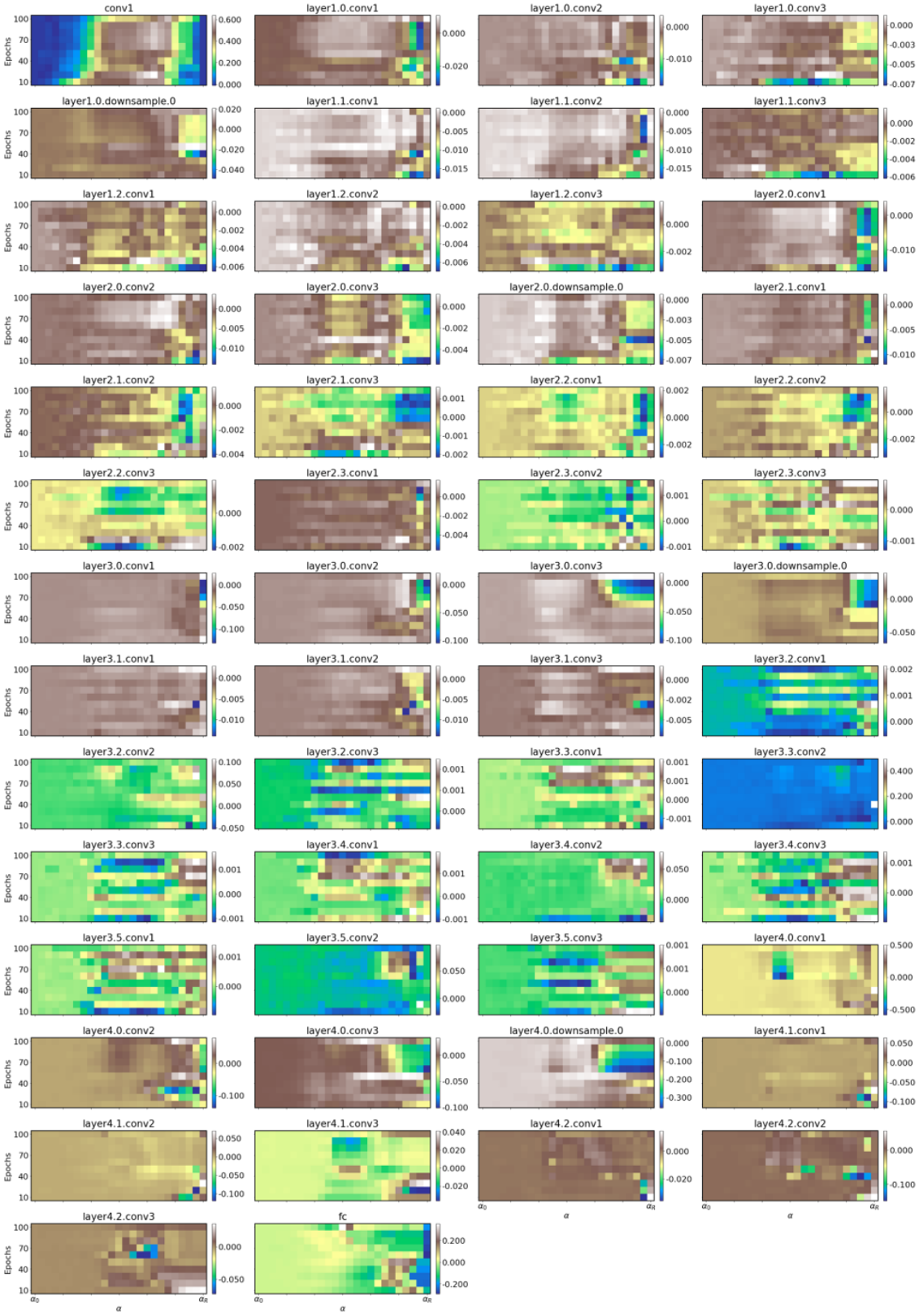


Fig. A.45 Difference in ResNet-50 responses to Clean and adversarial MNIST datasets, for synaptic filter h_1.

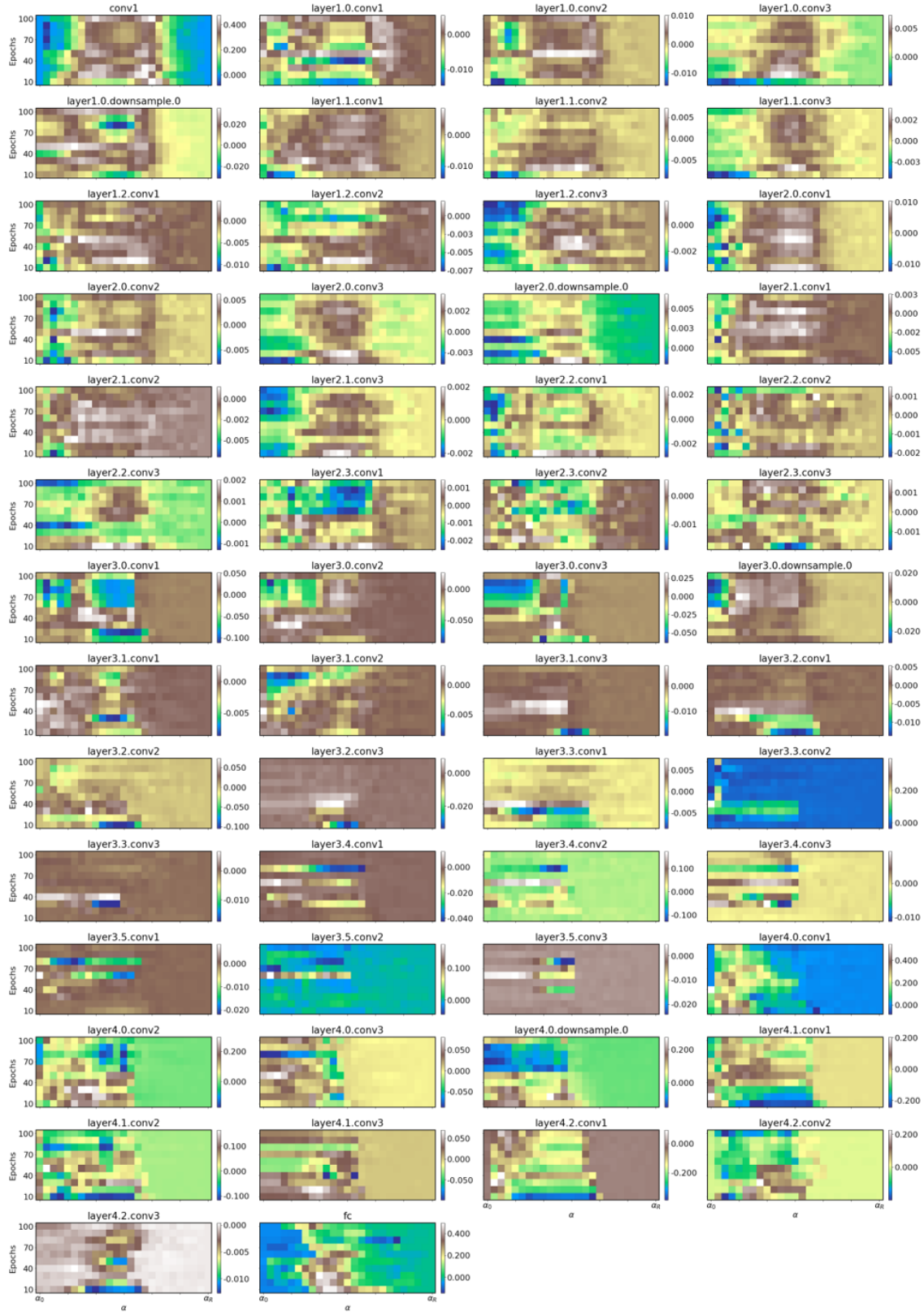


Fig. A.46 Difference in ResNet-50 responses to Clean and adversarial MNIST datasets, for synaptic filter h_2.

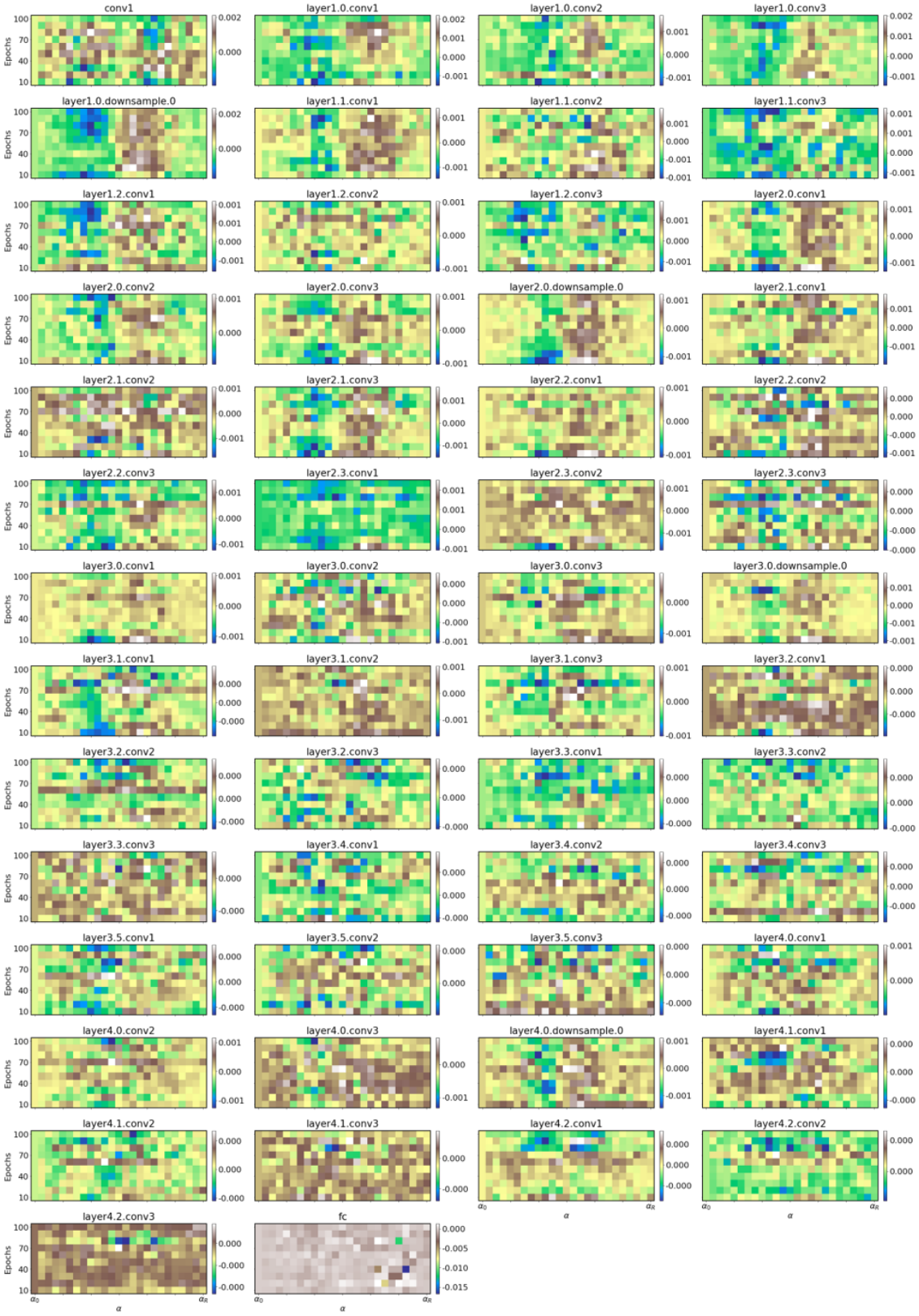


Fig. A.47 Difference in ResNet-50 responses to Clean and adversarial MNIST datasets, for synaptic filter h_3.

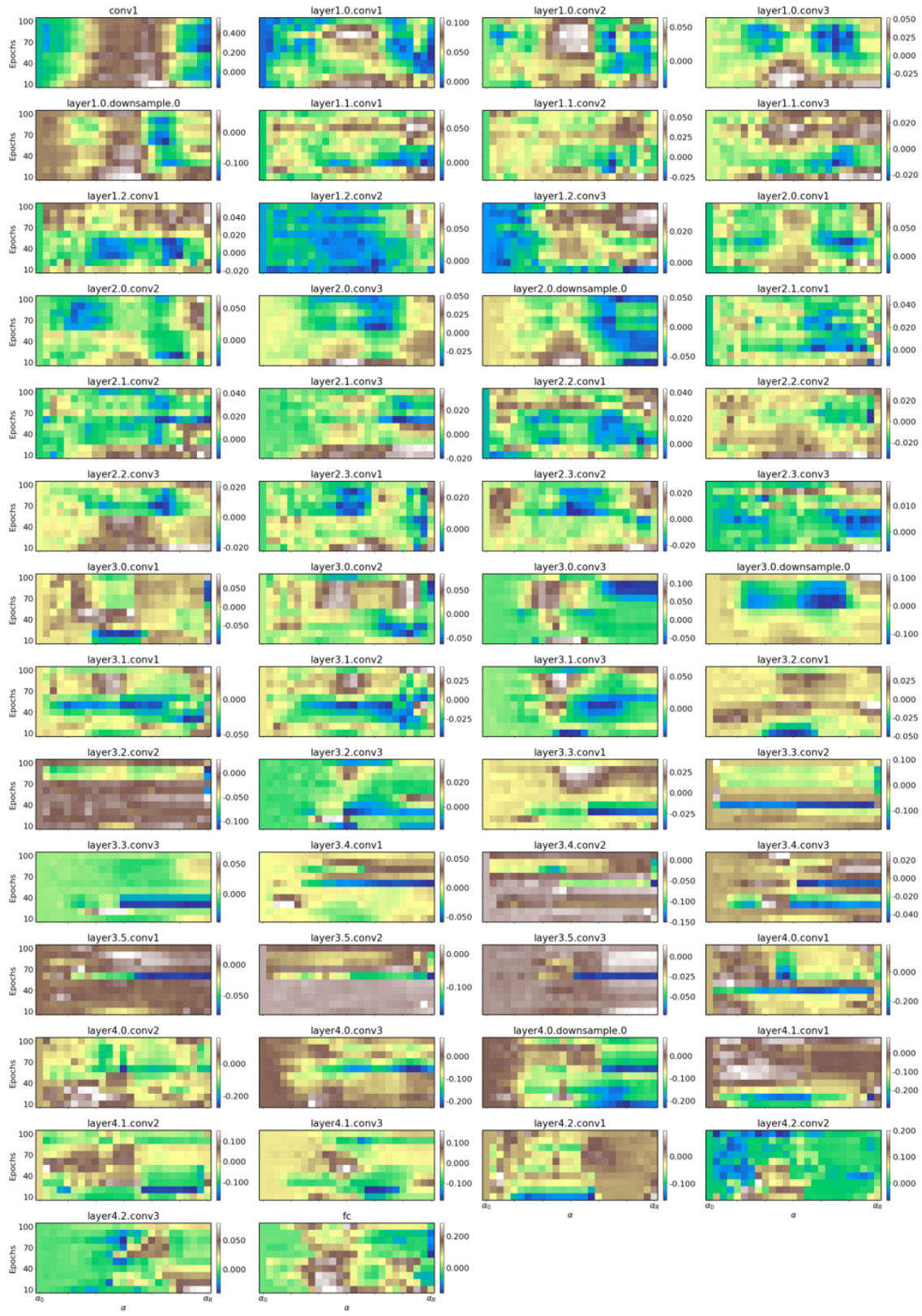


Fig. A.48 Combined difference in ResNet-50 responses to Clean and adversarial MNIST datasets, for all synaptic filters in h.

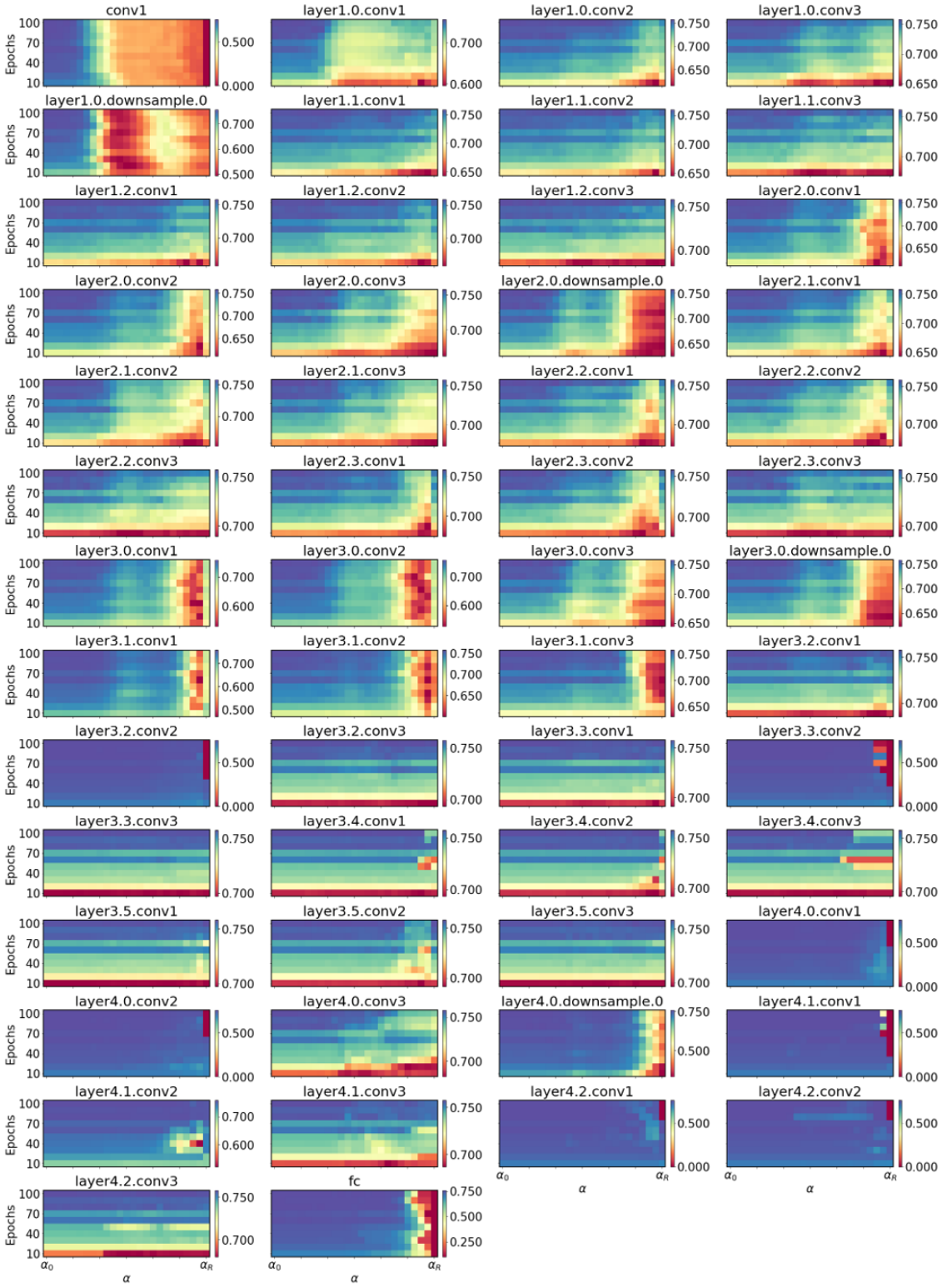


Fig. A.49 ResNet-50 response to clean CIFAR10 dataset, for synaptic filter h_1.

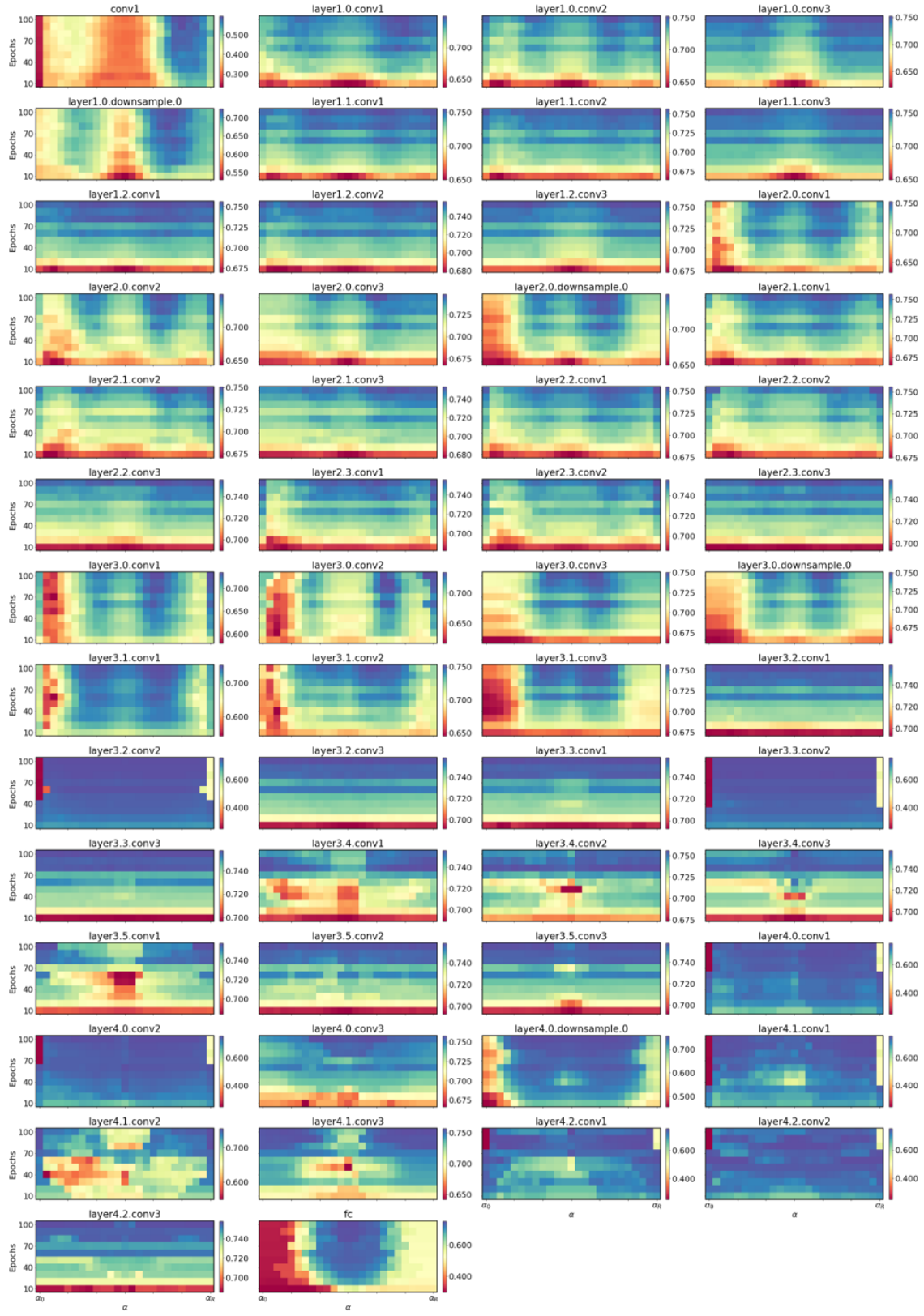


Fig. A.50 ResNet-50 response to clean CIFAR10 dataset, for synaptic filter h_2.

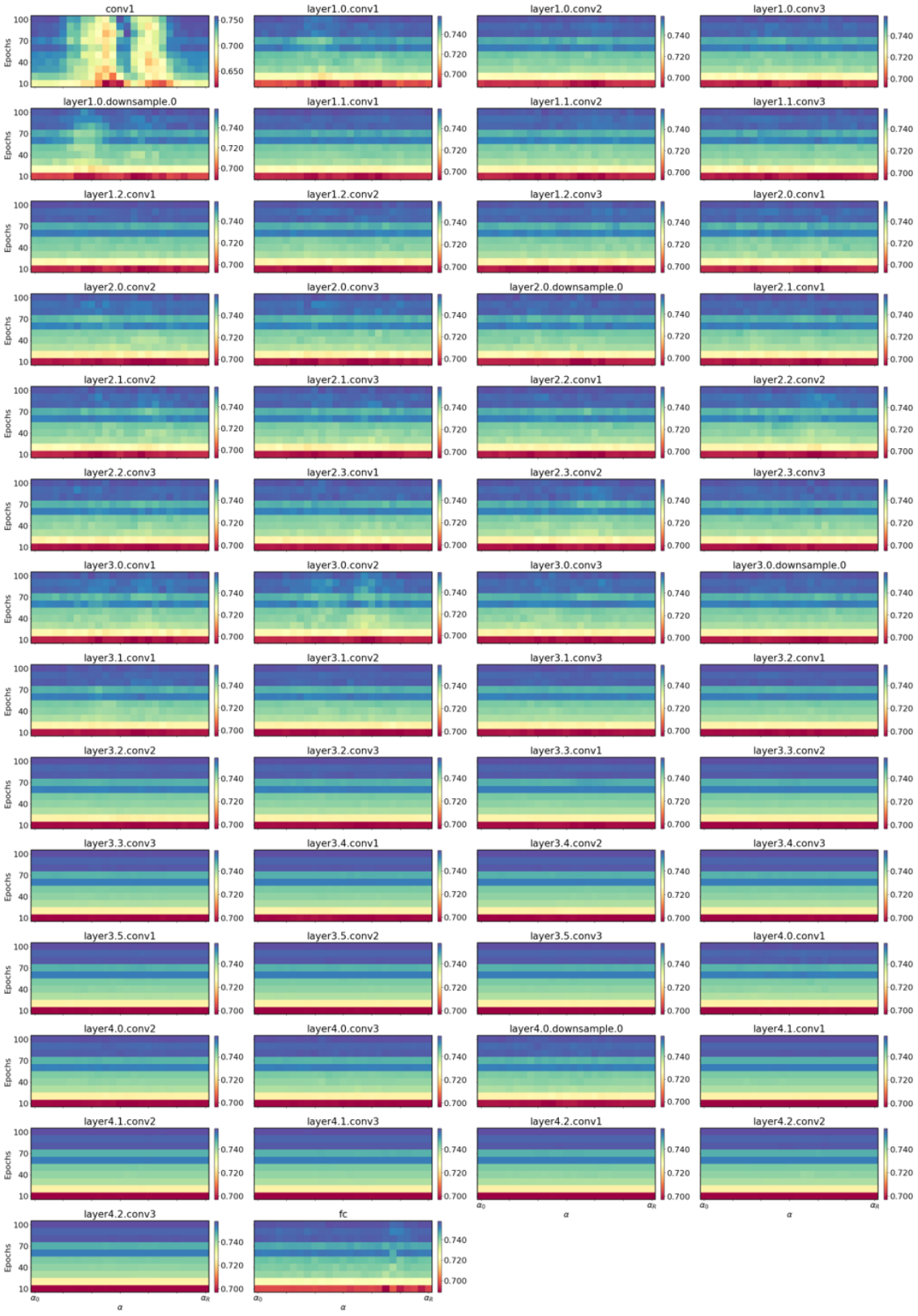


Fig. A.51 ResNet-50 response to clean CIFAR10 dataset, for synaptic filter h_3.

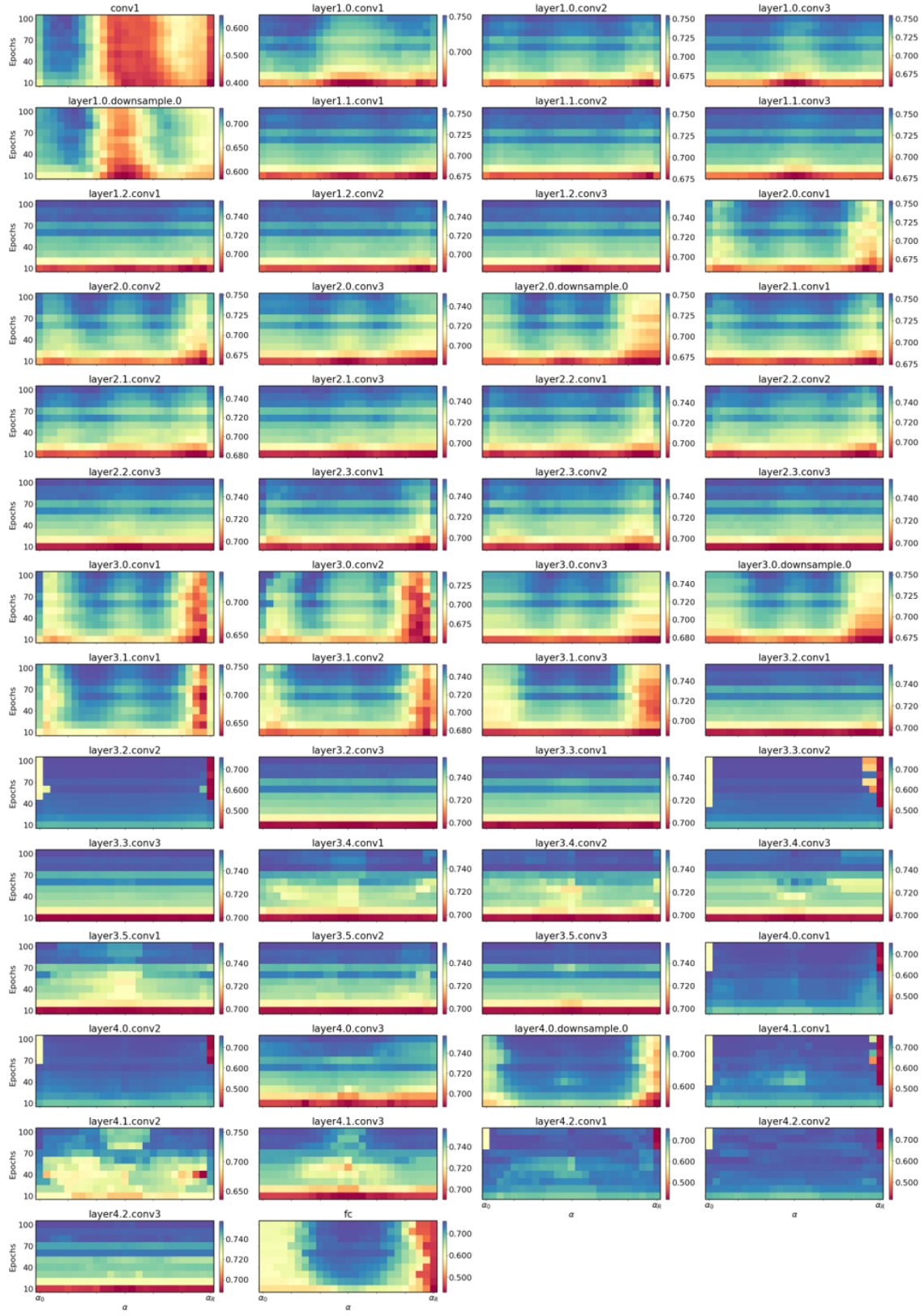


Fig. A.52 Combined ResNet-50 response to clean CIFAR10 dataset, for all synaptic filters in h.

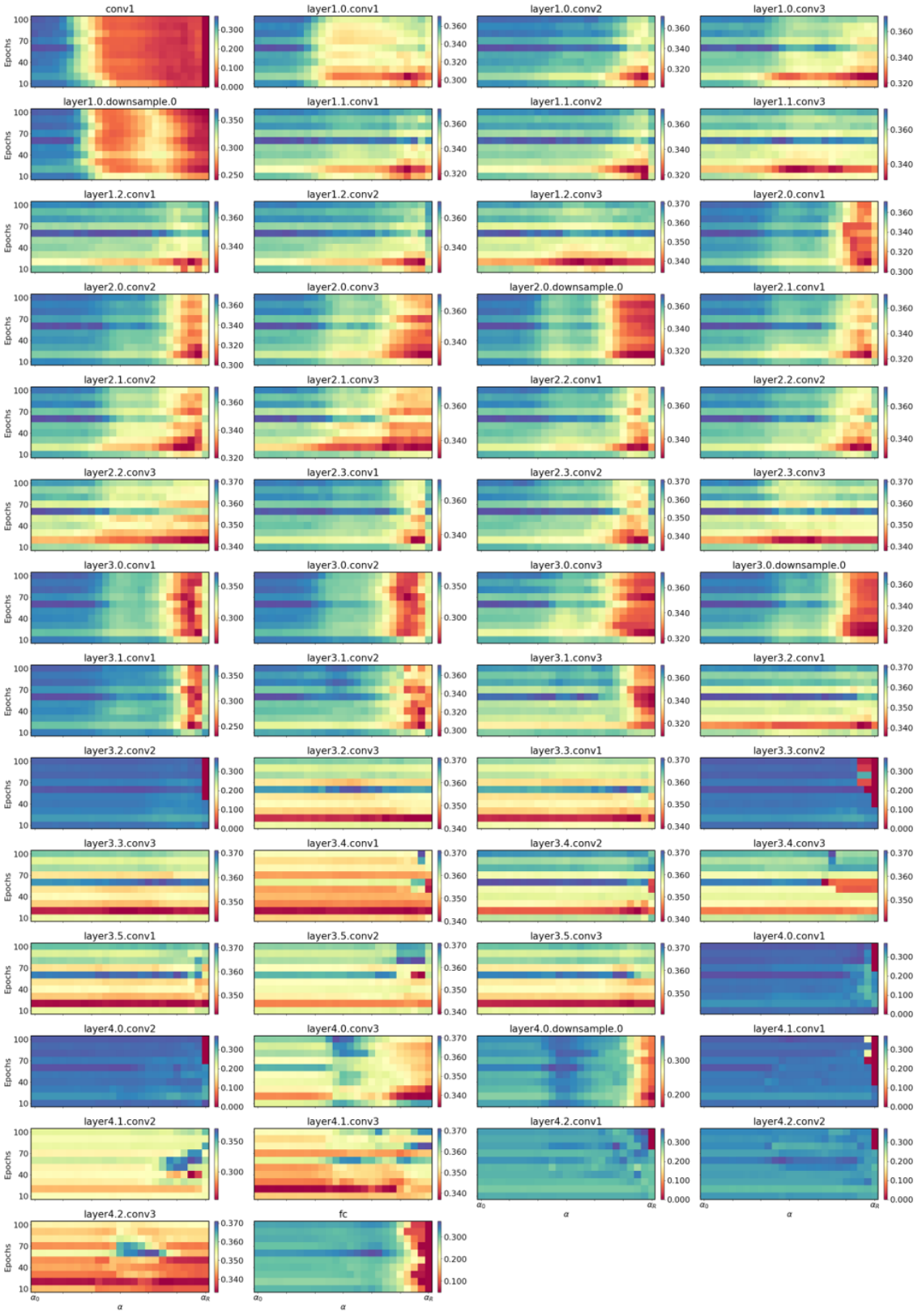


Fig. A.53 ResNet-50 response to adversarial CIFAR10 dataset, for synaptic filter h_1.

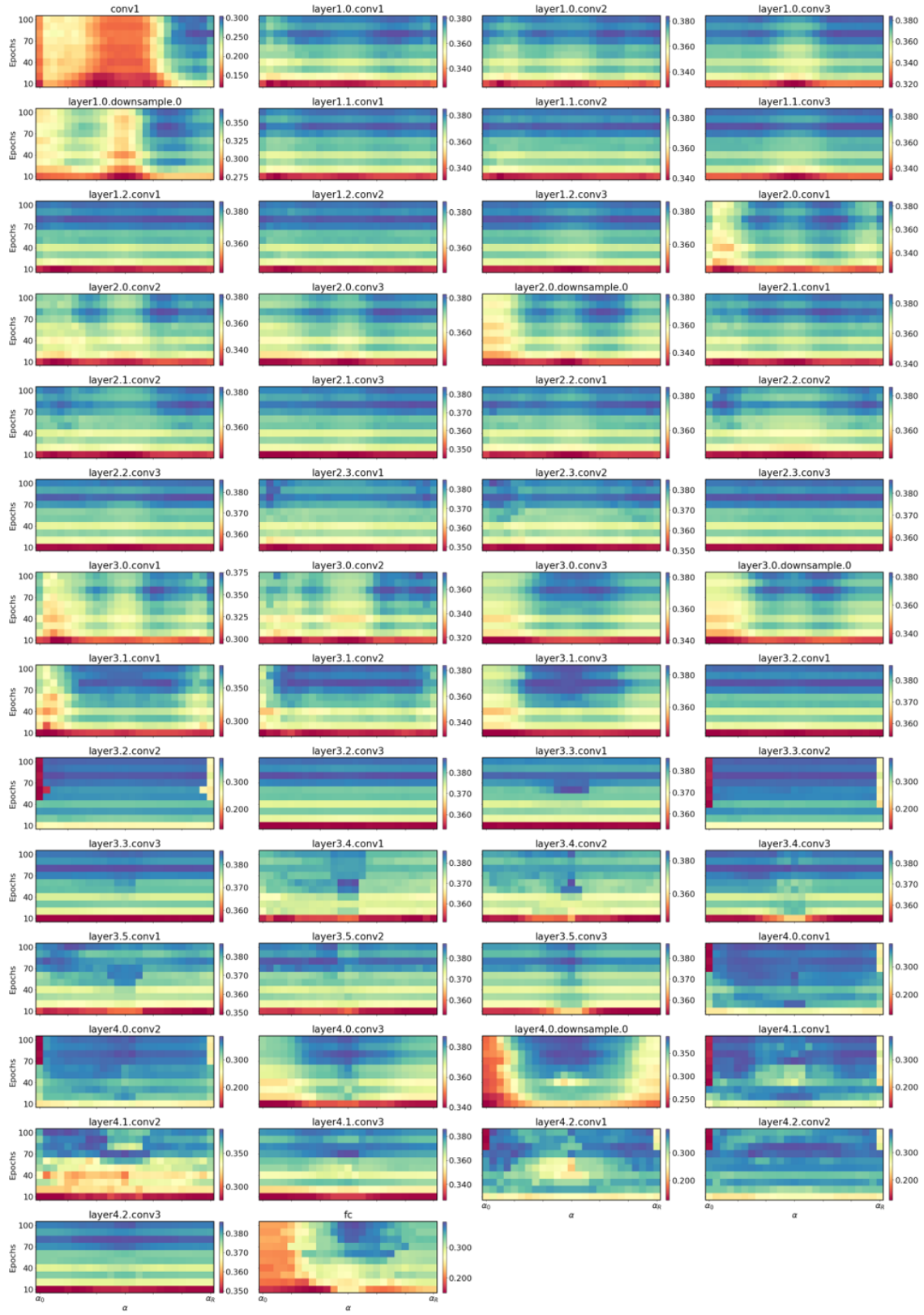


Fig. A.54 ResNet-50 response to adversarial CIFAR10 dataset, for synaptic filter h_2.

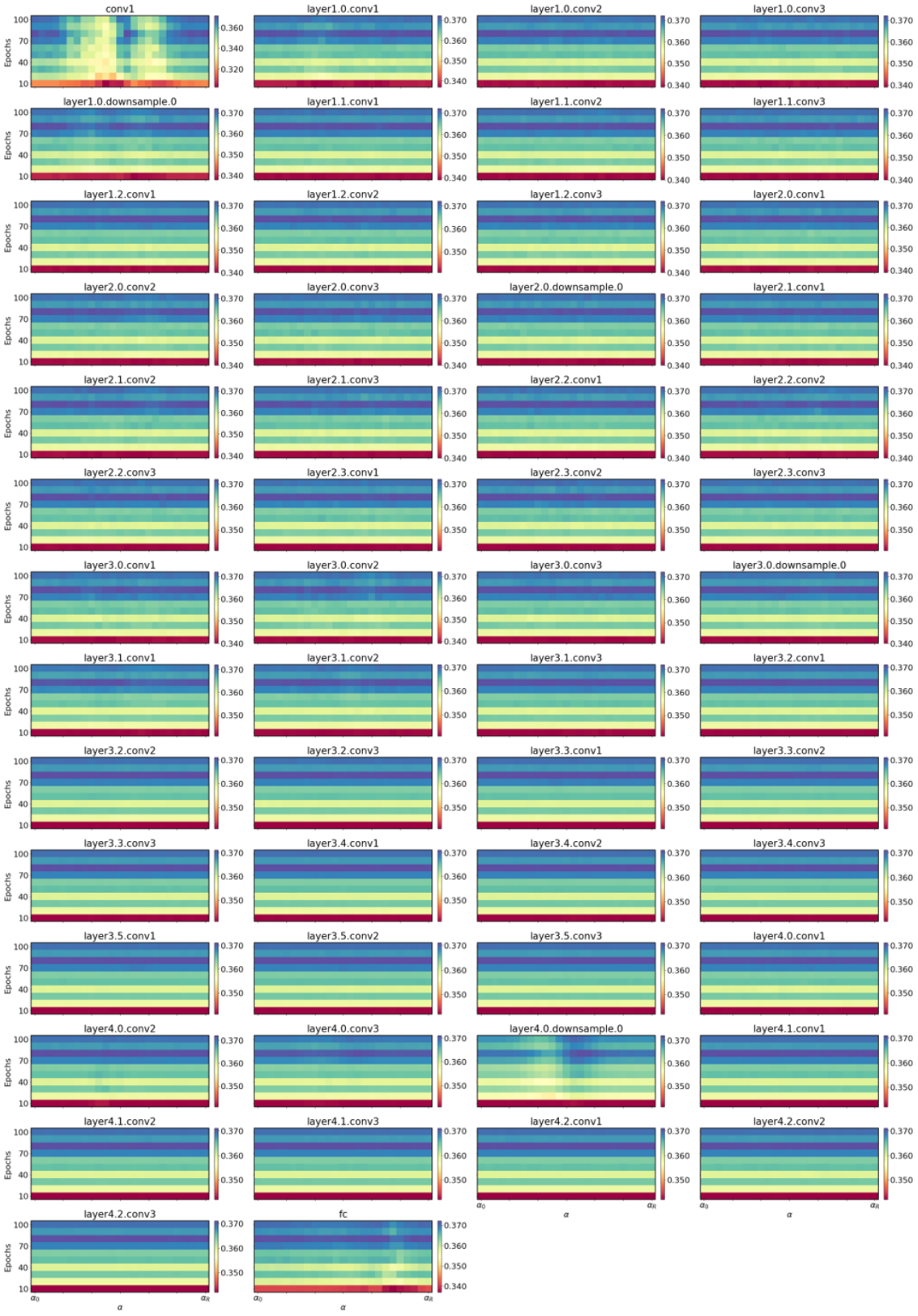


Fig. A.55 ResNet-50 response to adversarial CIFAR10 dataset, for synaptic filter h_3.

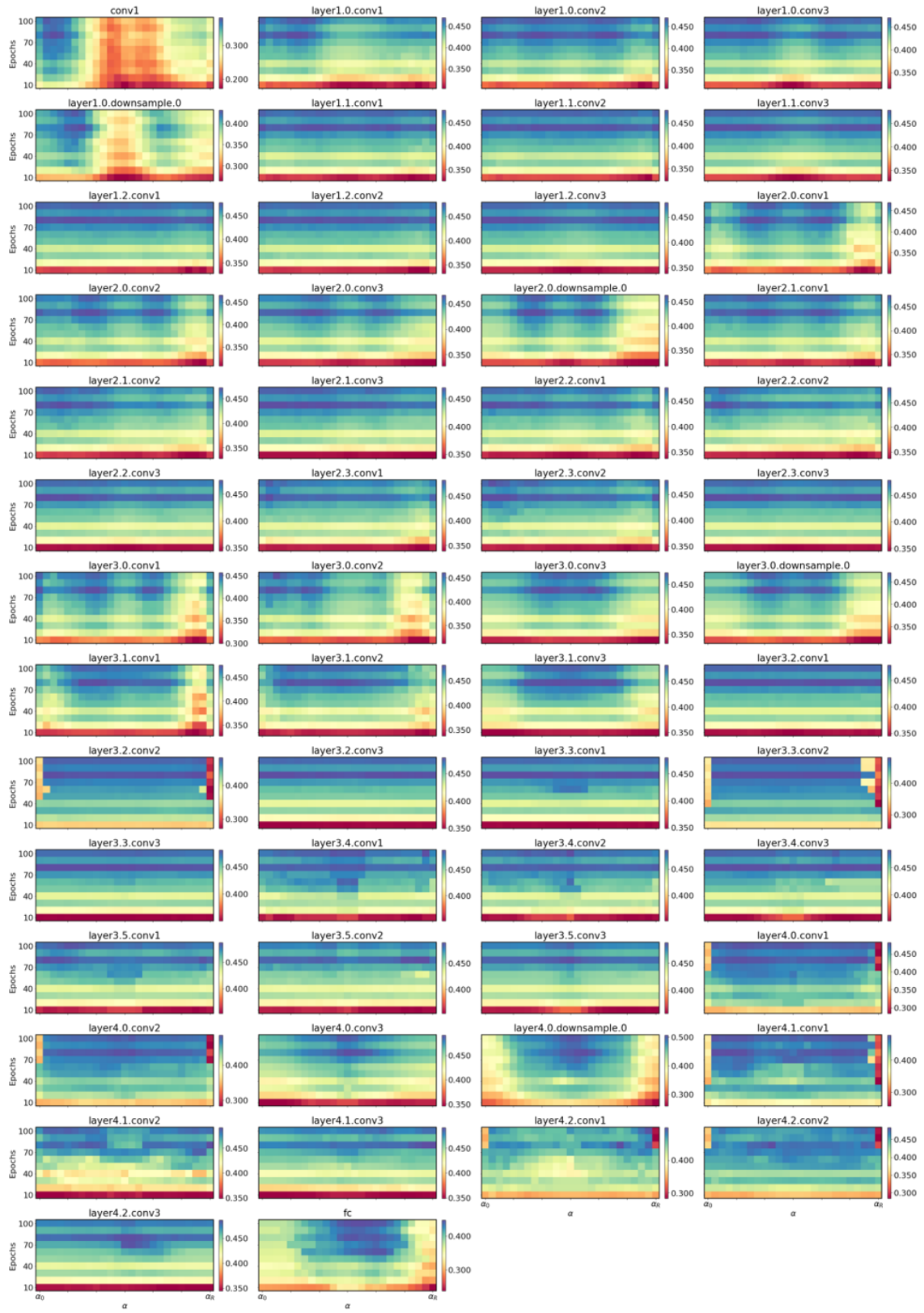


Fig. A.56 Combined ResNet-50 response to adversarial CIFAR10 dataset, for all synaptic filters in h.

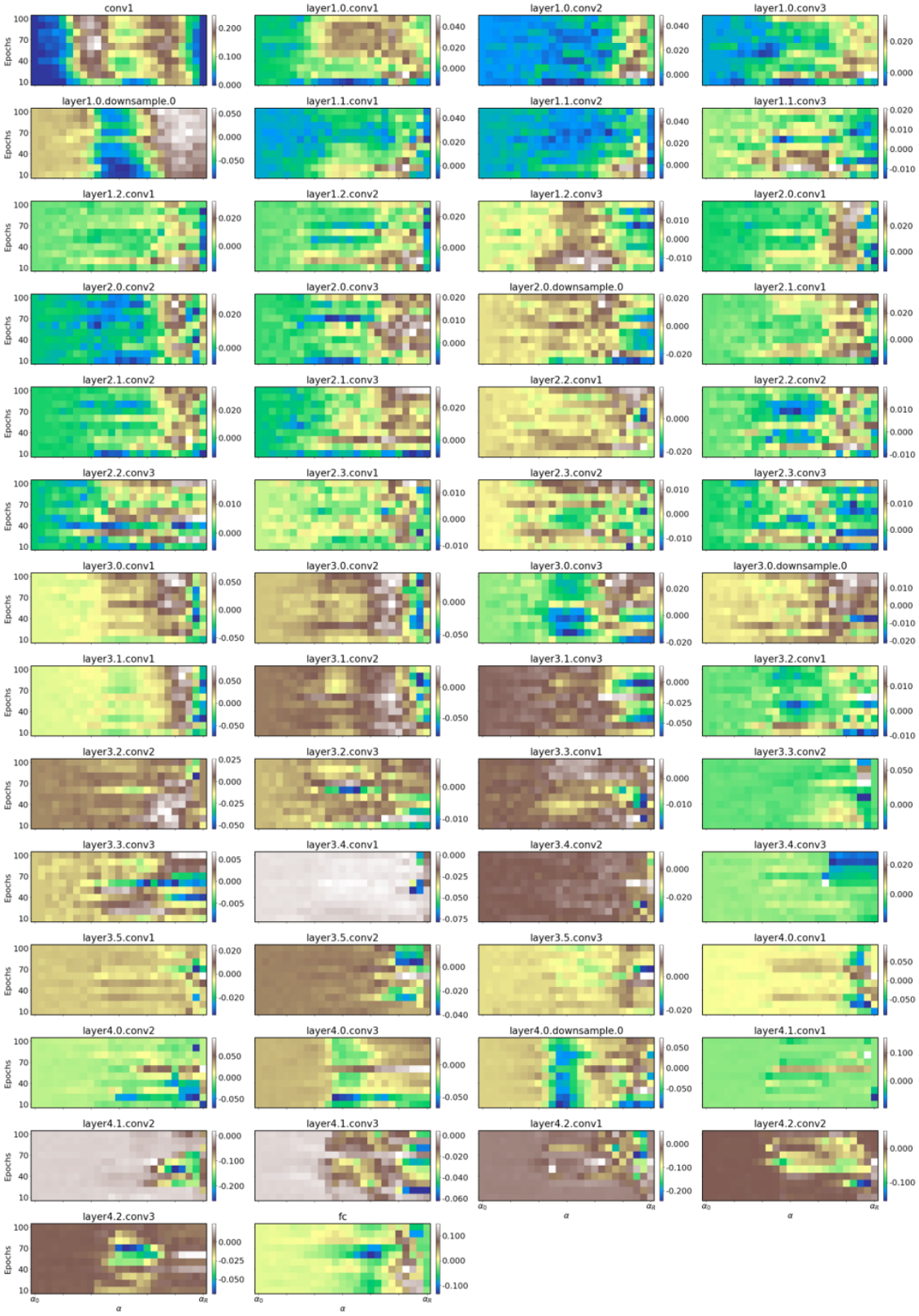


Fig. A.57 Difference in ResNet-50 responses to Clean and adversarial CIFAR10 datasets, for synaptic filter h_1.

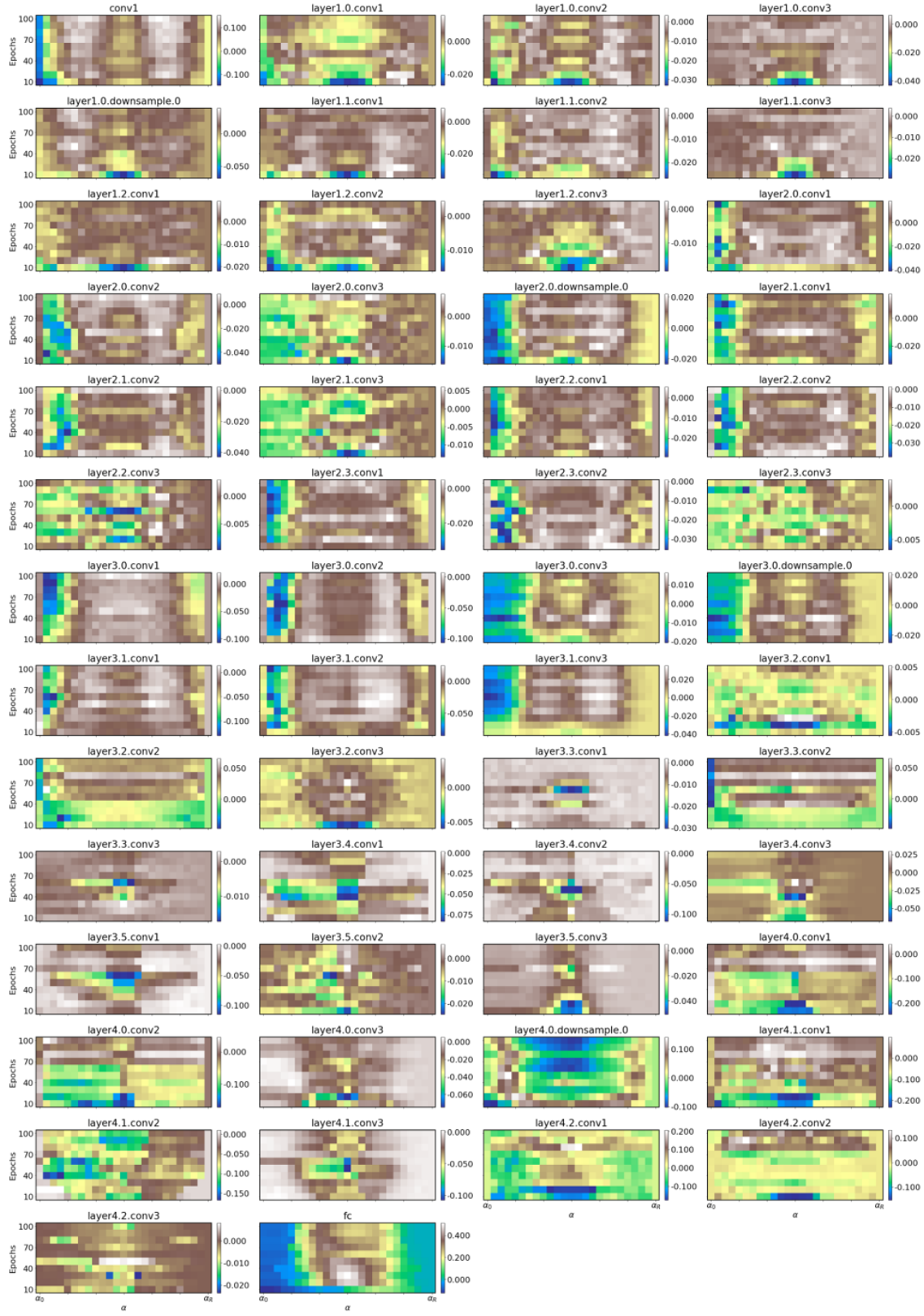


Fig. A.58 Difference in ResNet-50 responses to Clean and adversarial CIFAR10 datasets, for synaptic filter h_2.

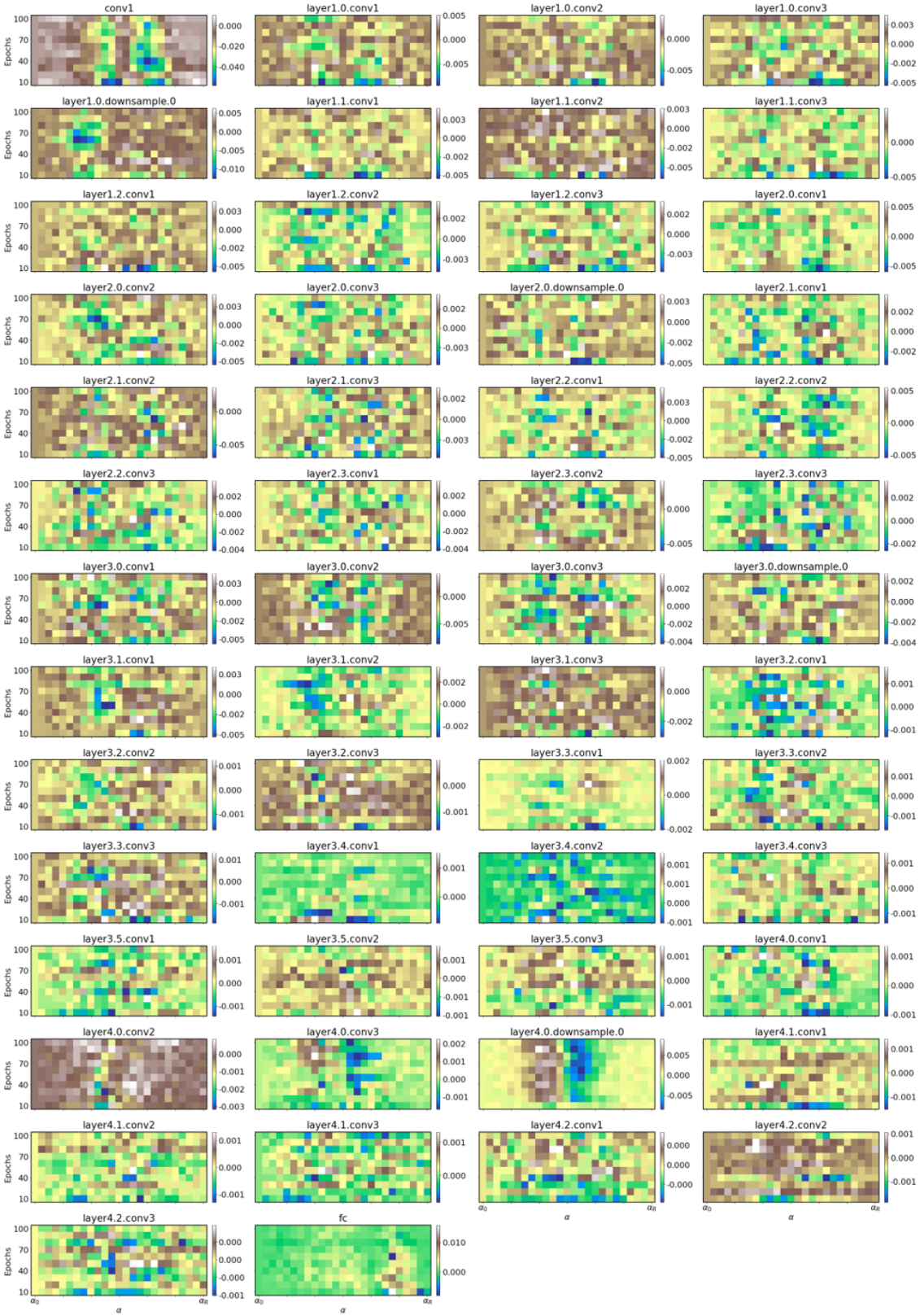


Fig. A.59 Difference in ResNet-50 responses to Clean and adversarial CIFAR10 datasets, for synaptic filter h_3.

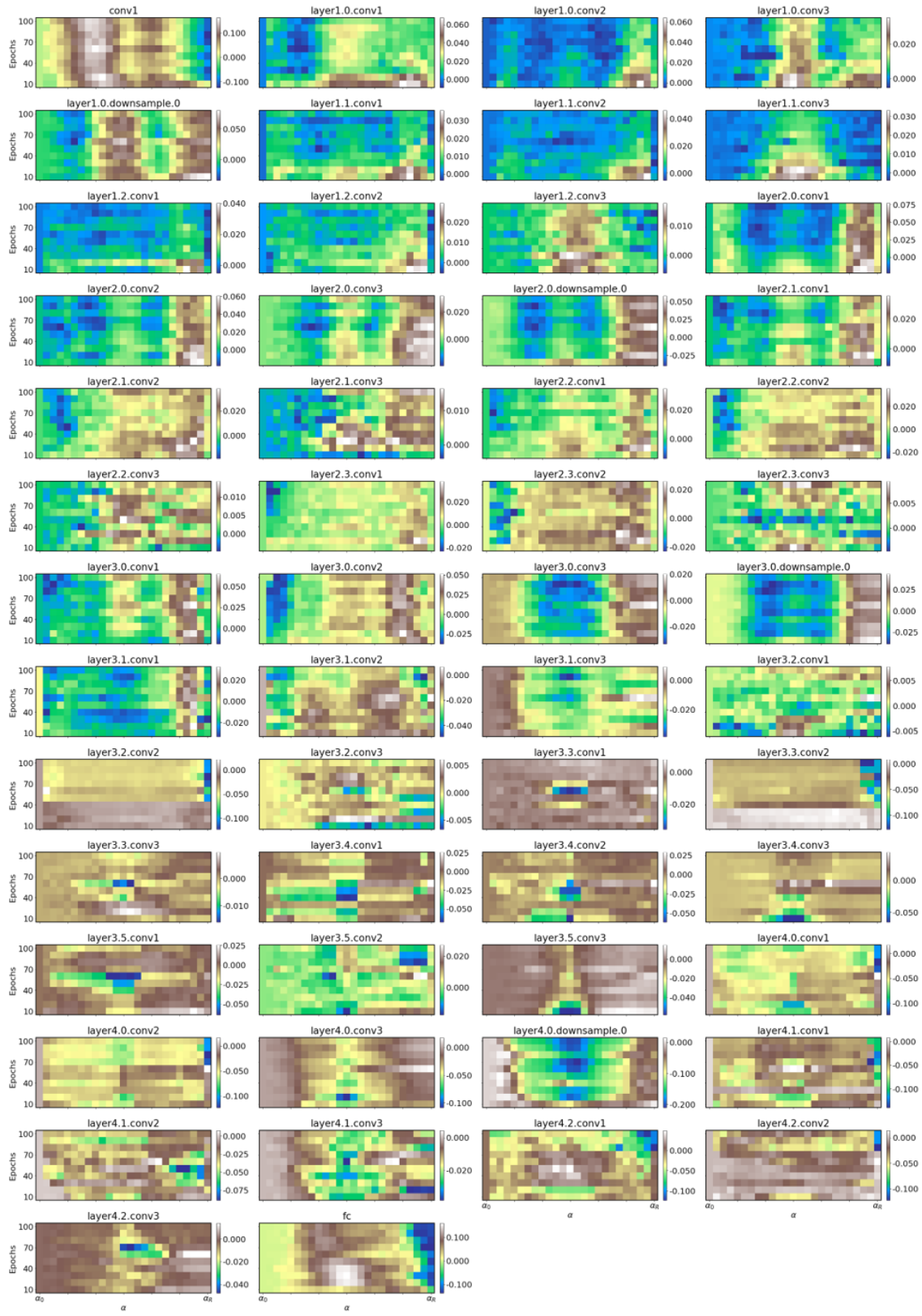


Fig. A.60 Combined difference in ResNet-50 responses to Clean and adversarial CIFAR10 datasets, for all synaptic filters in h.

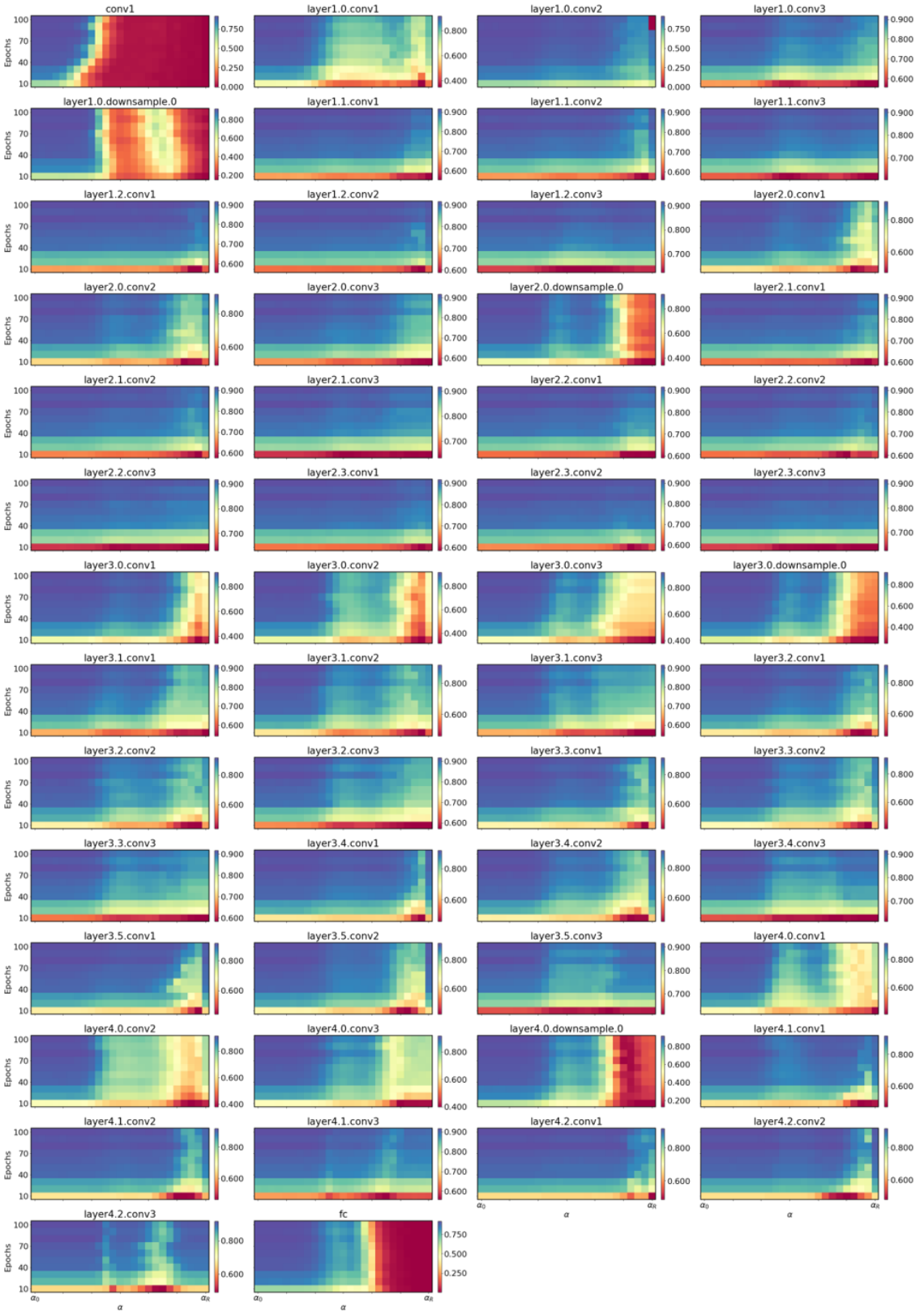


Fig. A.61 ResNet-50 response to clean ImageNet Tiny dataset, for synaptic filter h_1.

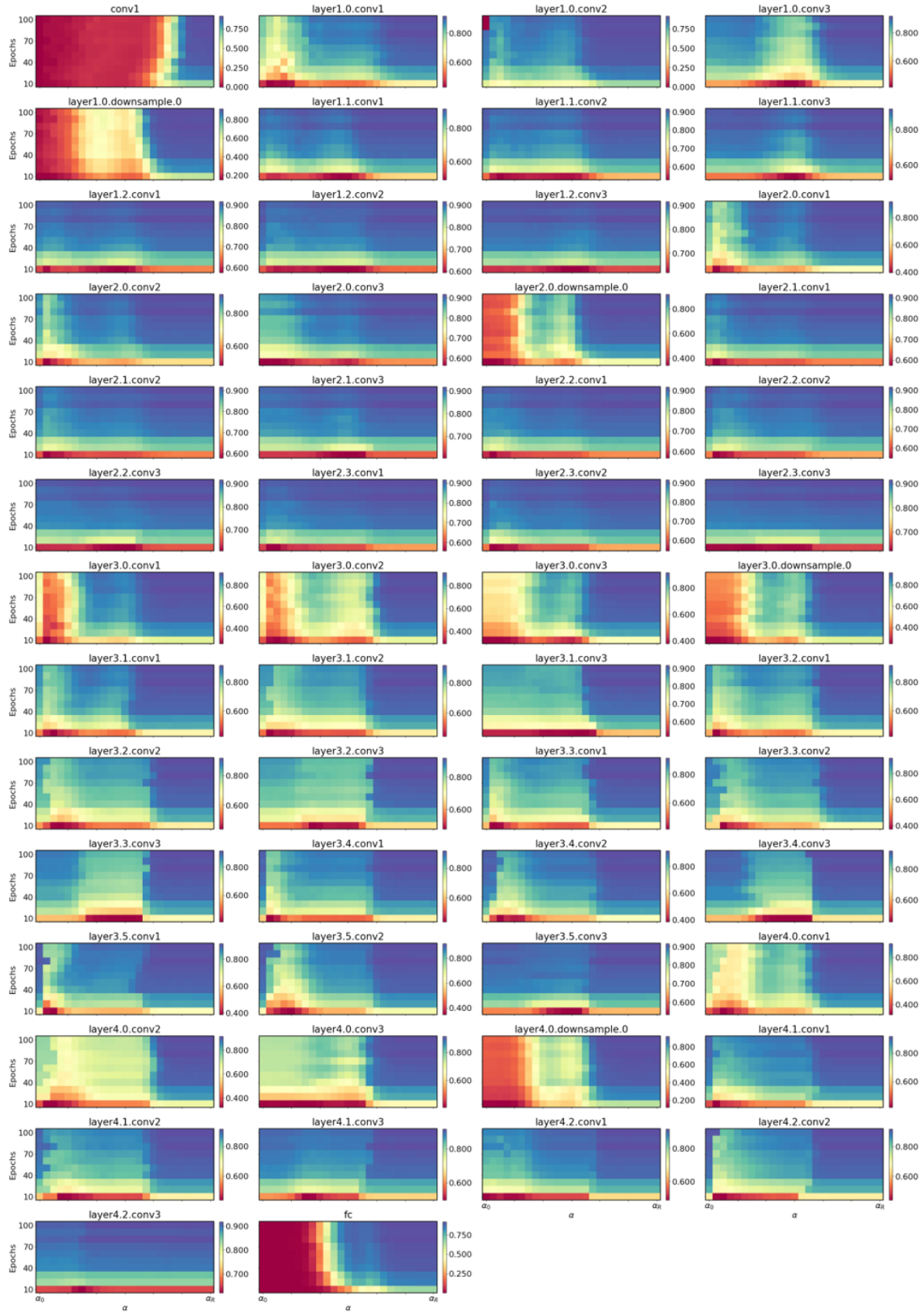


Fig. A.62 ResNet-50 response to clean ImageNet Tiny dataset, for synaptic filter h_2.

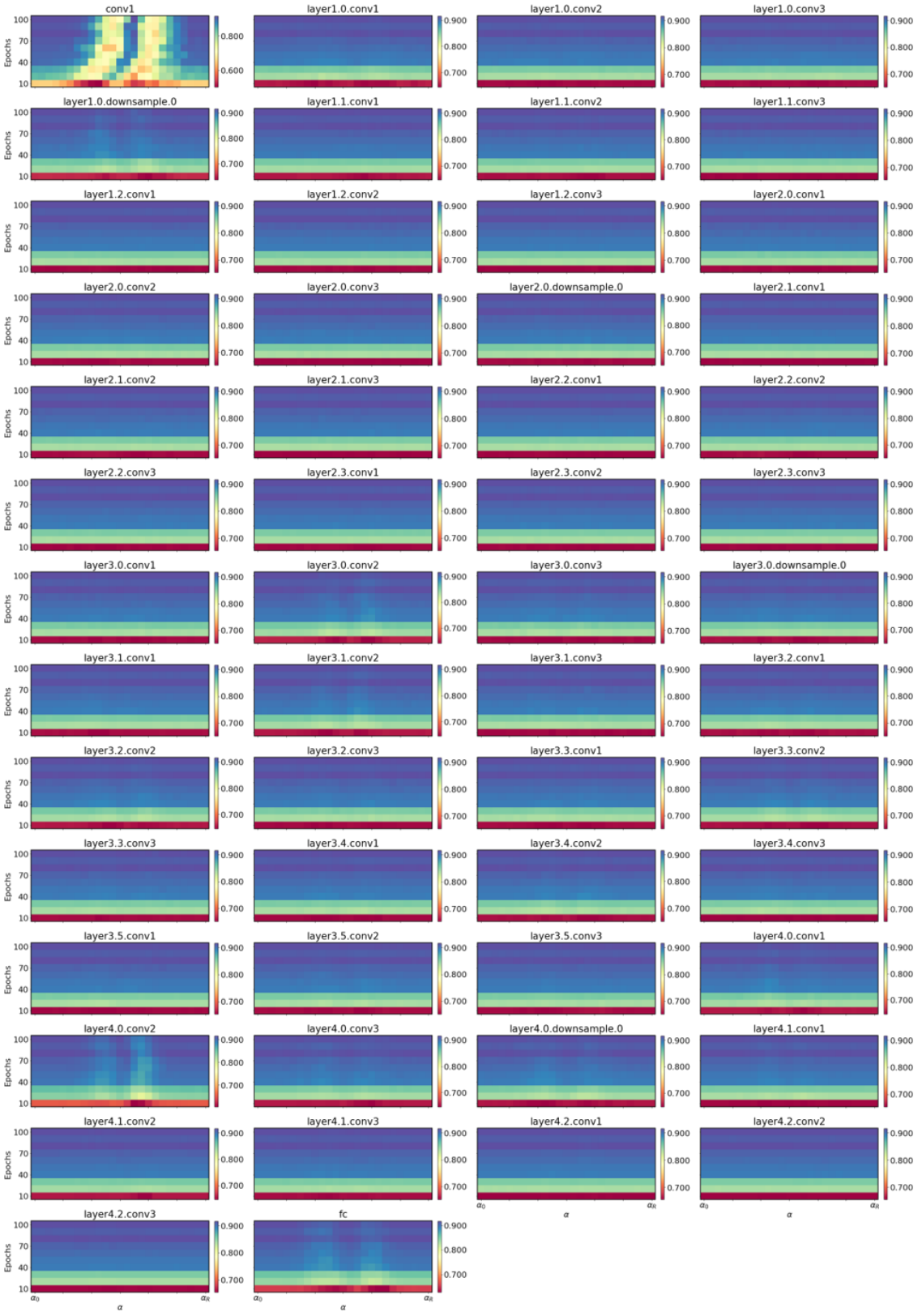


Fig. A.63 ResNet-50 response to clean ImageNet Tiny dataset, for synaptic filter h_3.

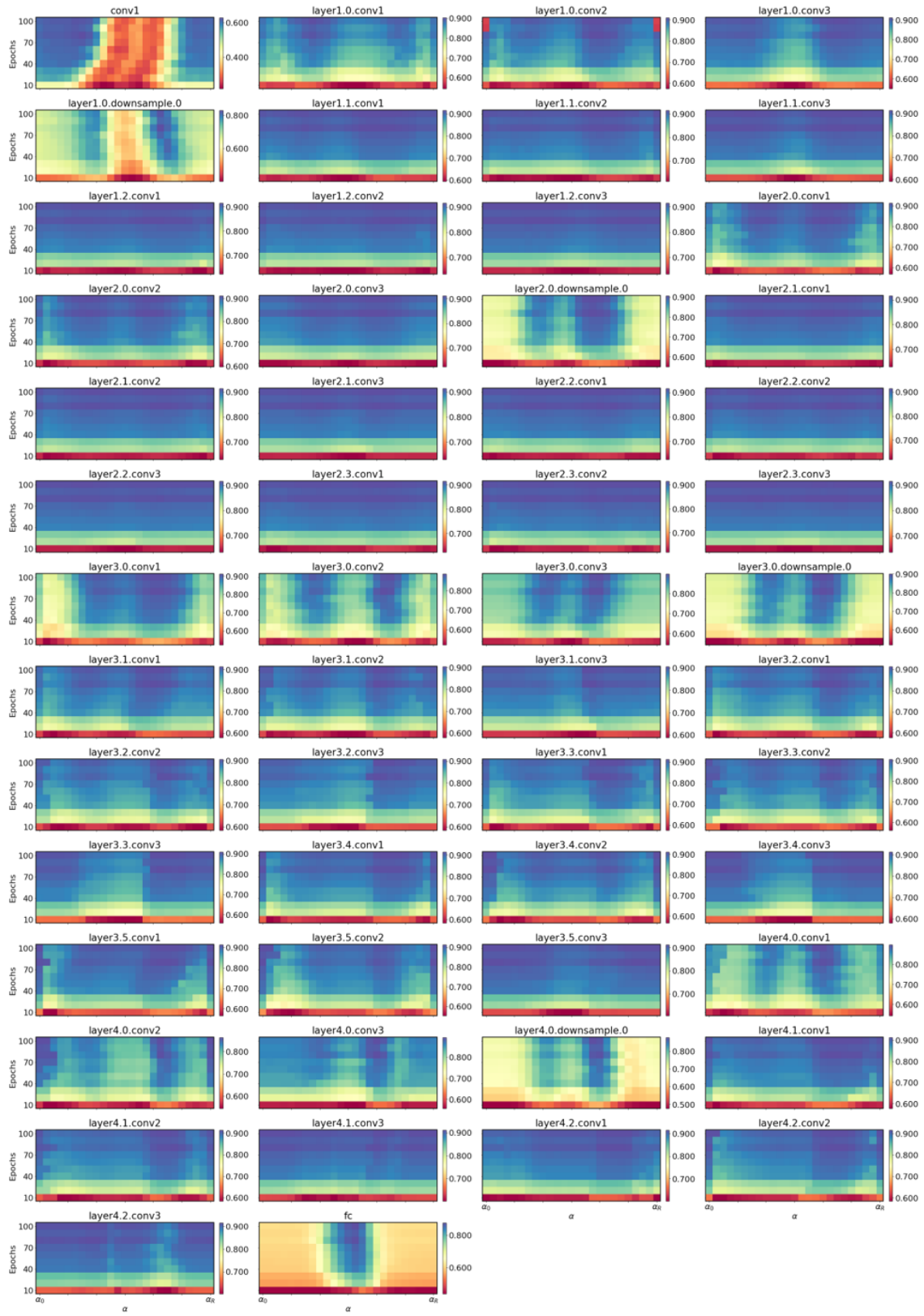


Fig. A.64 Combined ResNet-50 response to clean ImageNet Tiny dataset, for all synaptic filters in h.

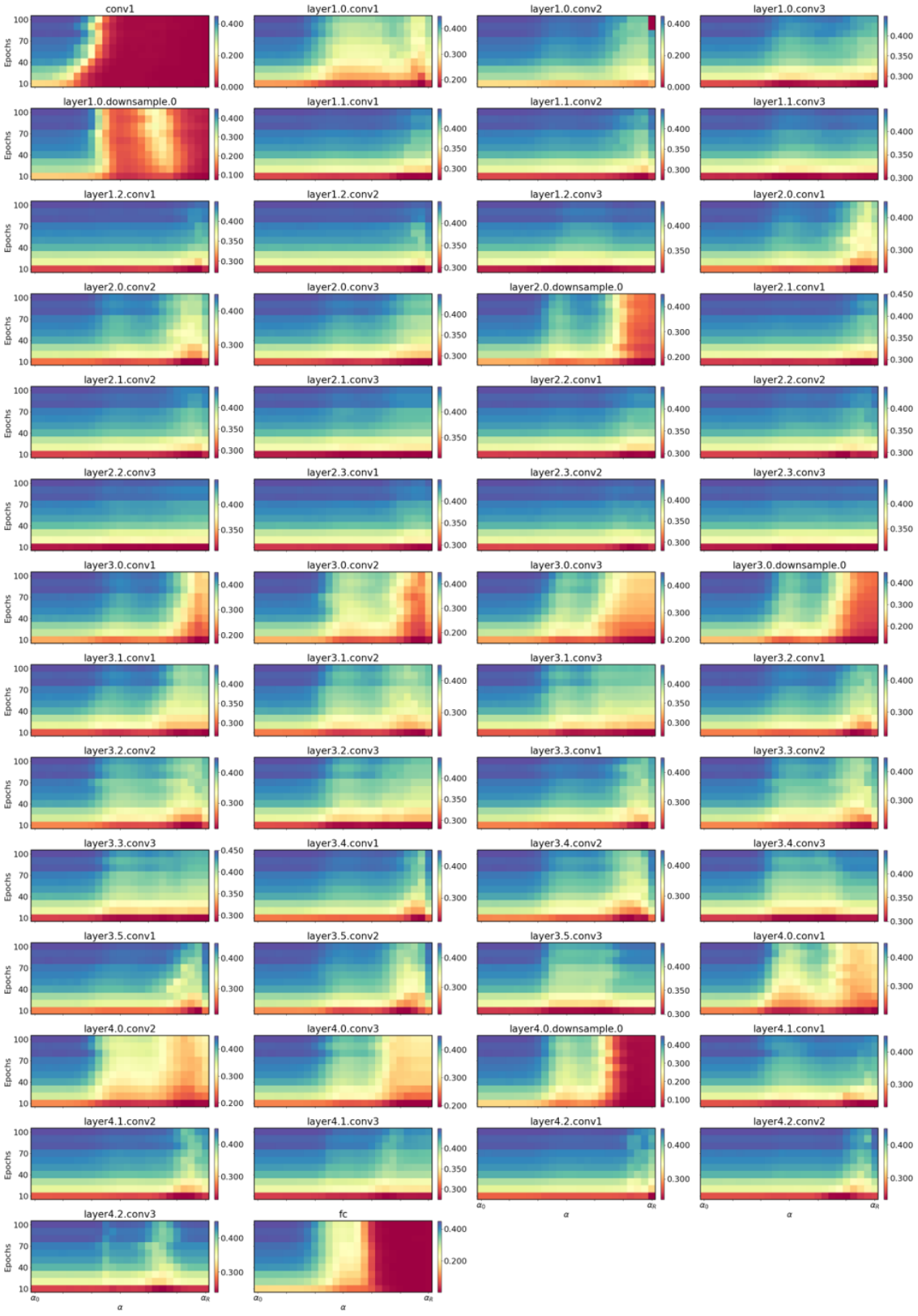


Fig. A.65 ResNet-50 response to adversarial ImageNet Tiny dataset, for synaptic filter h_1.

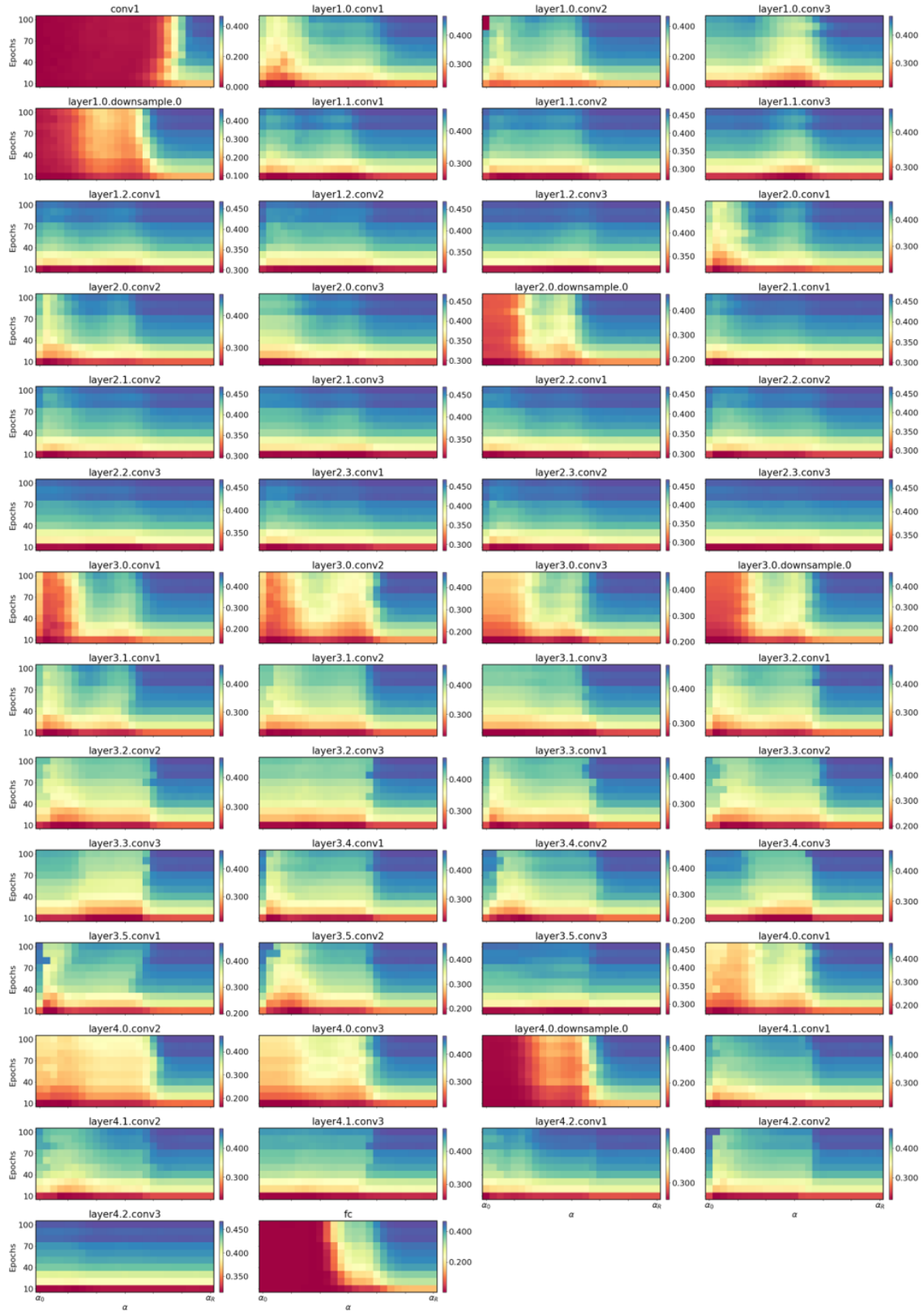


Fig. A.66 ResNet-50 response to adversarial ImageNet Tiny dataset, for synaptic filter h_2.

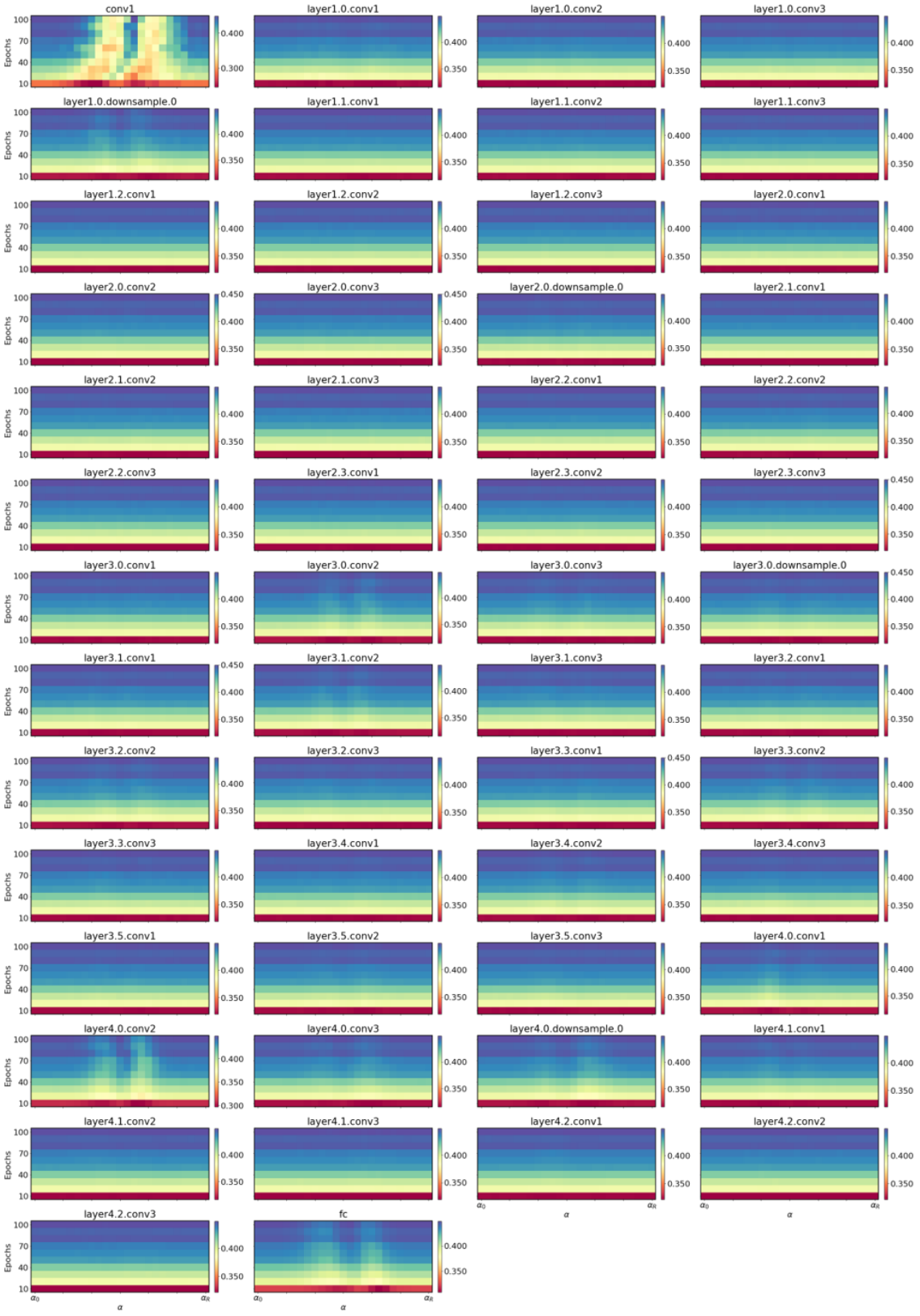


Fig. A.67 ResNet-50 response to adversarial ImageNet Tiny dataset, for synaptic filter h_3.

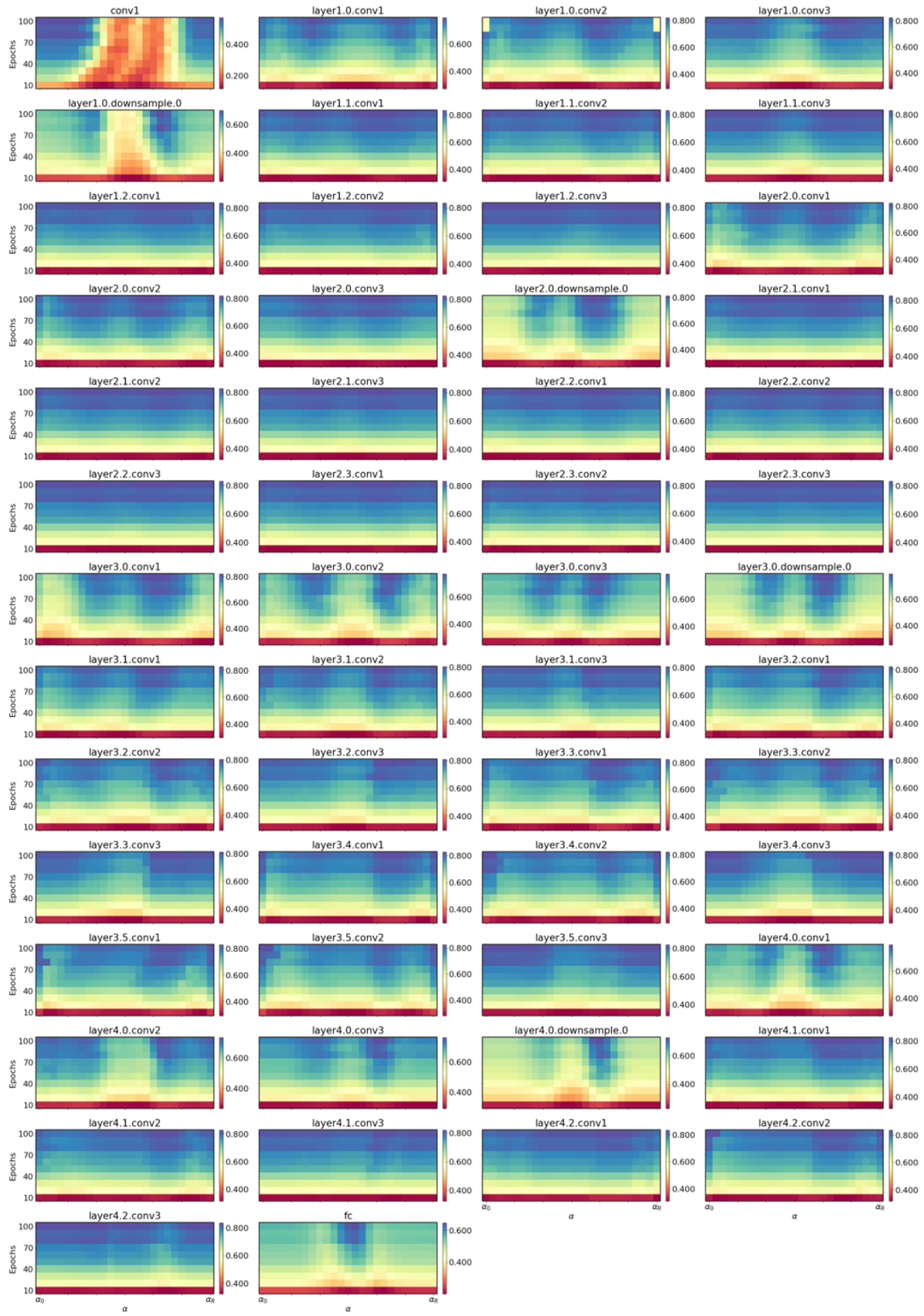


Fig. A.68 Combined ResNet-50 response to adversarial ImageNet Tiny dataset, for all synaptic filters in h.

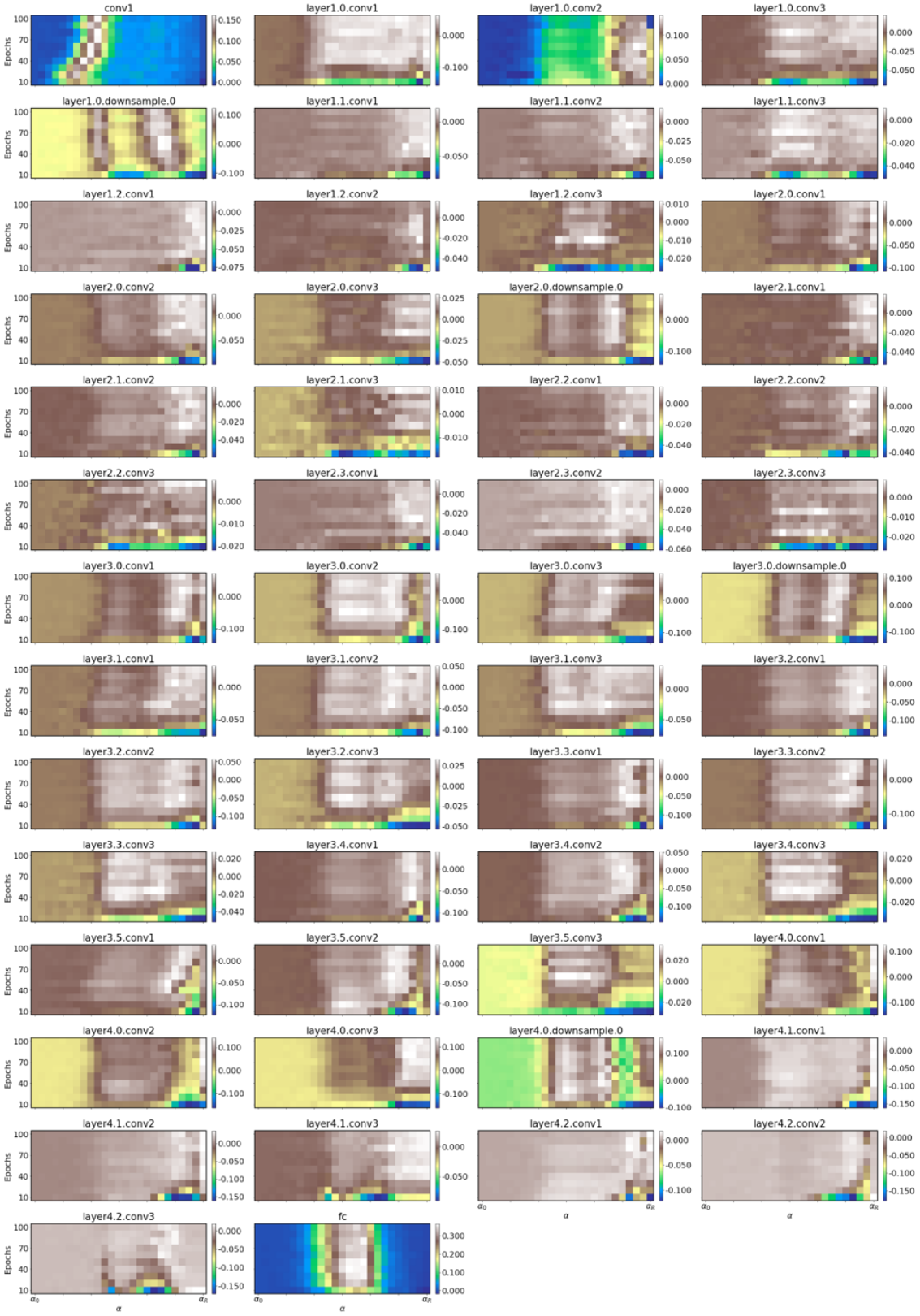


Fig. A.69 Difference in ResNet-50 responses to Clean and adversarial ImageNet Tiny datasets, for synaptic filter h_1.

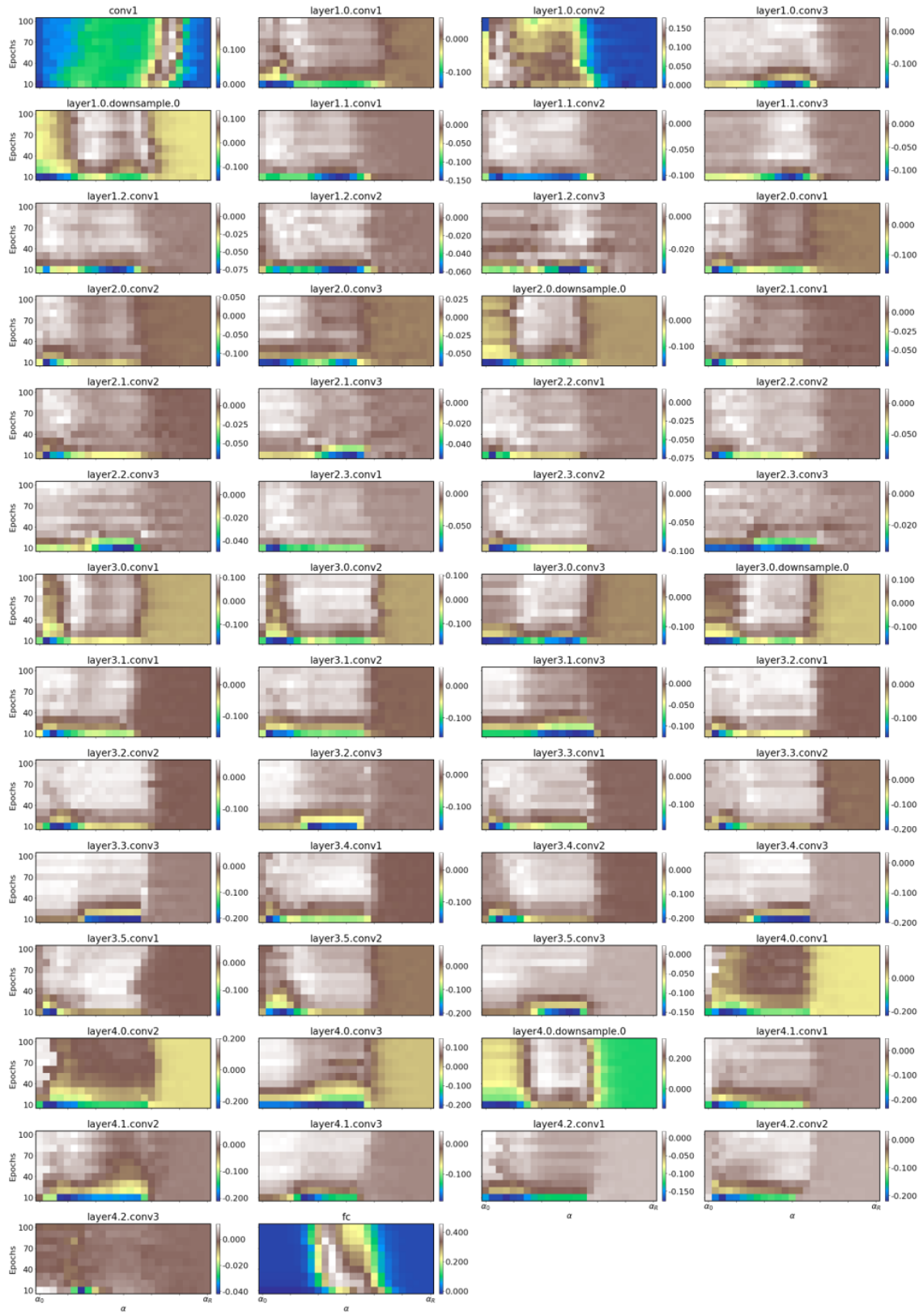


Fig. A.70 Difference in ResNet-50 responses to Clean and adversarial ImageNet Tiny datasets, for synaptic filter h_2.

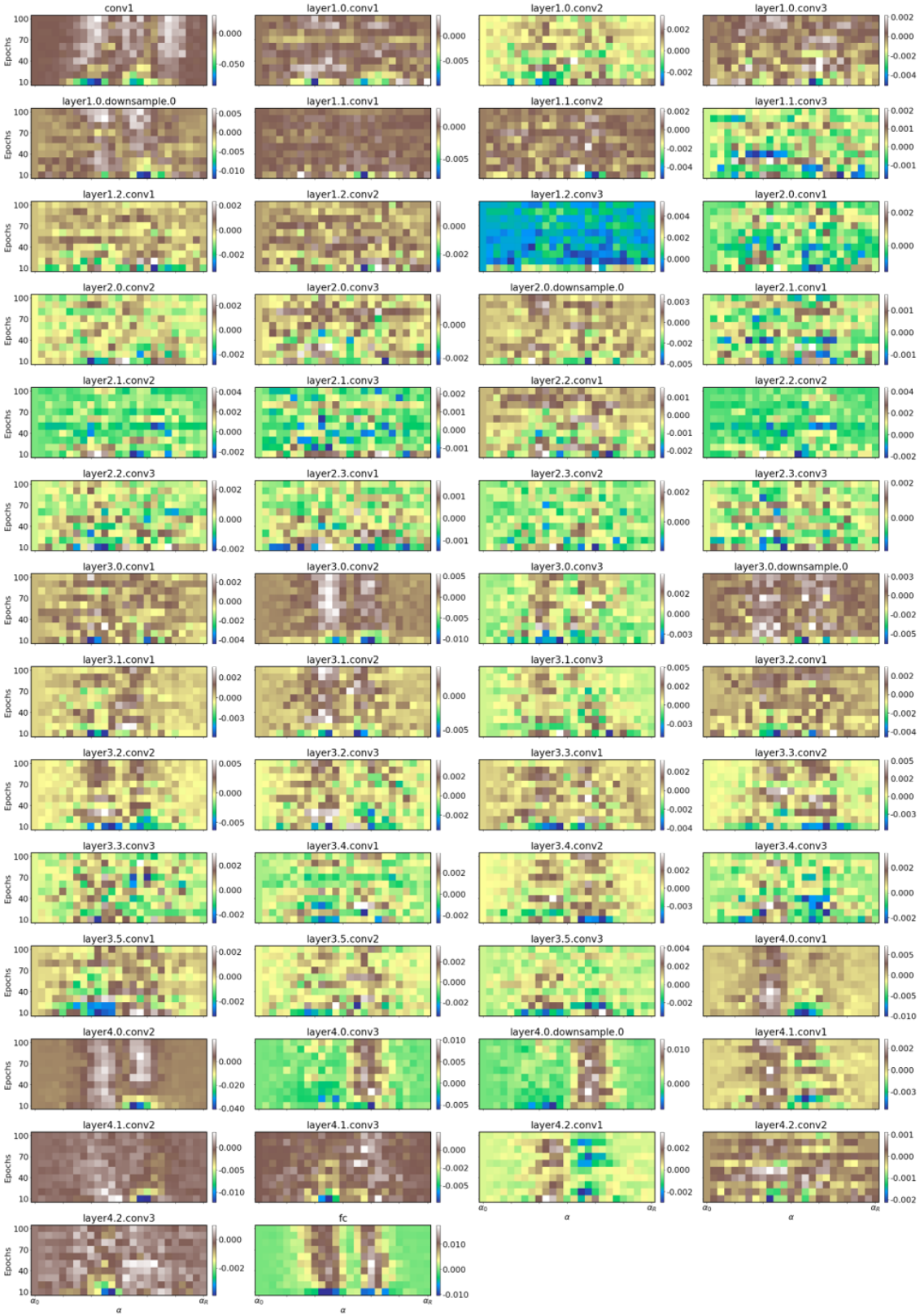


Fig. A.71 Difference in ResNet-50 responses to Clean and adversarial ImageNet Tiny datasets, for synaptic filter h_3.

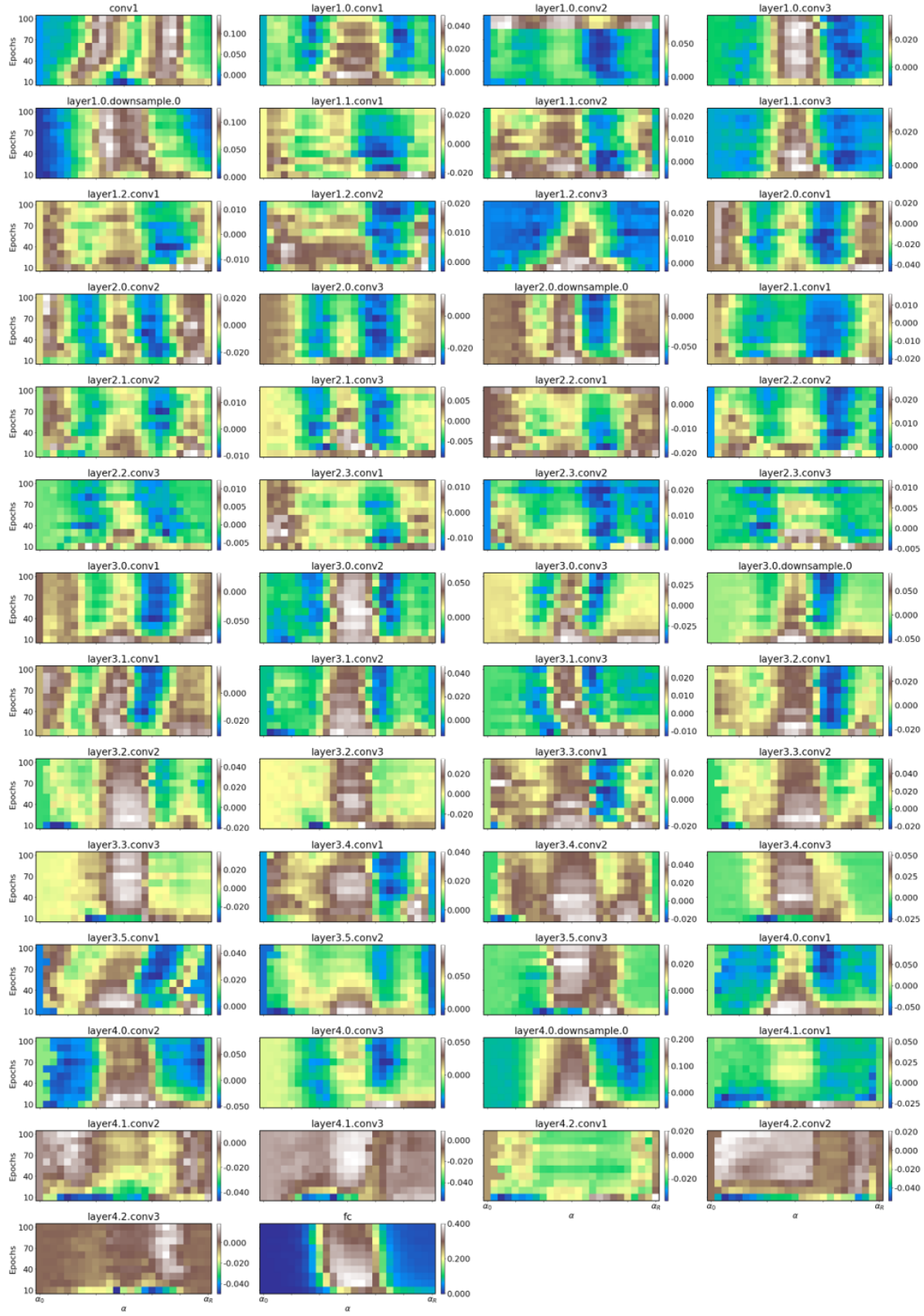


Fig. A.72 Combined difference in ResNet-50 responses to Clean and adversarial ImageNet Tiny datasets, for all synaptic filters in h.

A.3.2 Adversarial Dataset Responses

The SqueezeNet-v1.1 network test accuracy responses to the different synaptic filters (see Sec. 3) on the adversarial datasets is given for all layers of the network. We present the SqueezeNet-v1.1 responses to the synaptic filter h_1, for the adversarially perturbed MNIST dataset in Fig. A.74. The constraints for the perturbation magnitude of the attack is given in Sec. 3. The Pixel intensities in the presented figures show the measured network accuracy for all threshold values α ranging from α_0 to α_A, as per Sec. 3, measured every 10 epochs from 10 to 100 epochs during network training. Similarly, the network test accuracy responses for SqueezeNet-v1.1 on the adversarially perturbed CIFAR10 dataset, are presented in Fig. A.77. The SqueezeNet-v1.1 network responses for the adversarially perturbed ImageNet Tiny dataset are presented in Fig. A.80. Using the results presented in Fig. A.74, Fig. A.77, and Fig. A.80 we find links and layers, similar to those highlighted for the clean dataset responses, where there are invariant response characteristics to different synaptic filters for the three evaluated adversarial datasets on the SqueezeNet-v1.1 network. Further analysis of results is presented in Sec. 5.

A.3.3 Scaled Response Difference

The SqueezeNet-v1.1 network scaled test accuracy responses difference (see Sec. 3.2) between the clean and adversarial datastes, to the different synaptic filters is given for all layers of the network. We present the SqueezeNet-v1.1 response differences to the synaptic filter h_1 for the MNIST dataset in Figs. A.75. The constraints for the perturbation magnitude of the attack is given in Sec. 3. The Pixel intensities in the presented figures show the measured network accuracy for all threshold values α ranging from α_0 to α_A, as per Sec. 3, measured every 10 epochs from 10 to 100 epochs during network training. The network test accuracy response differences, for SqueezeNet-v1.1 on the CIFAR10 dataset are presented in Fig. A.78. The SqueezeNet-v1.1 network response differences for ImageNet Tiny dataset are presented in Fig. A.81. Using the results presented in Fig. A.75, Fig. A.78, and Fig. A.81 we find links and layers where the adversary is targeting the SqueezeNet-v1.1 network. We also find layer response differences that show different characteristics for different datasets, further analysis of results is given in Sec. 5.

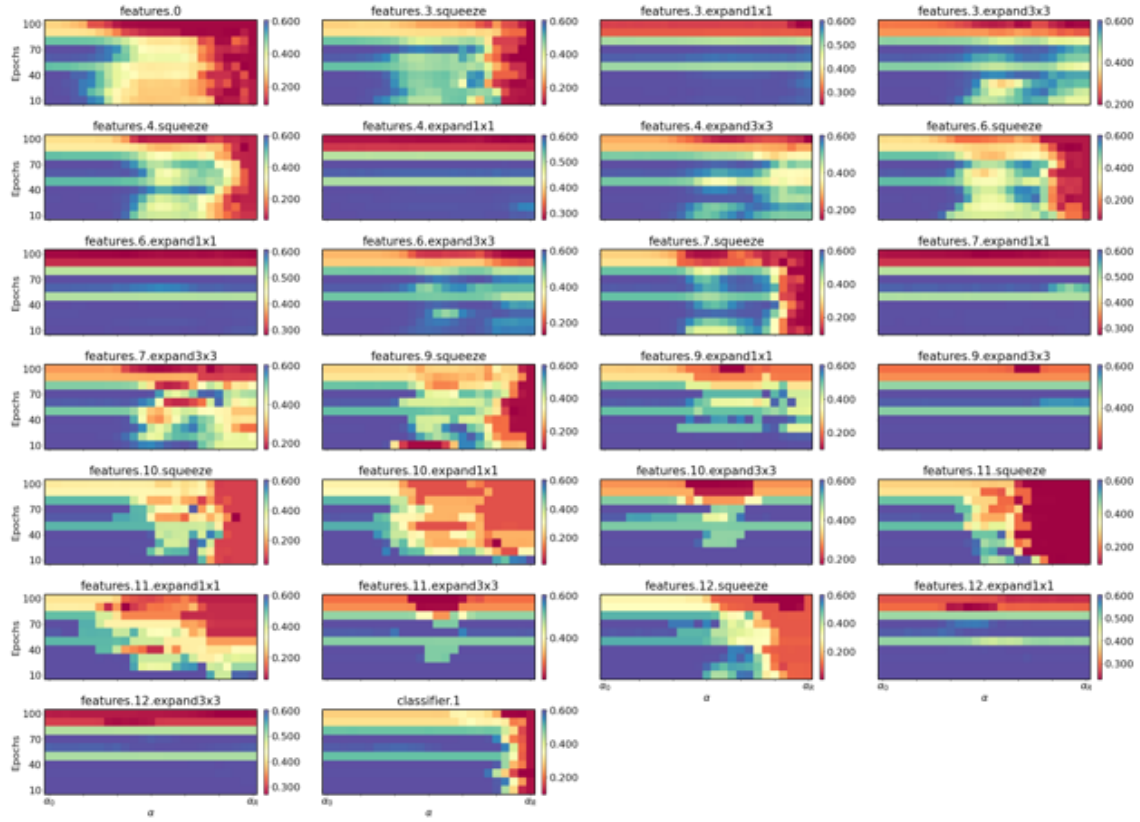


Fig. A.73 SqueezeNet-v1.1 response to clean MNIST dataset, for synaptic filter h_1.

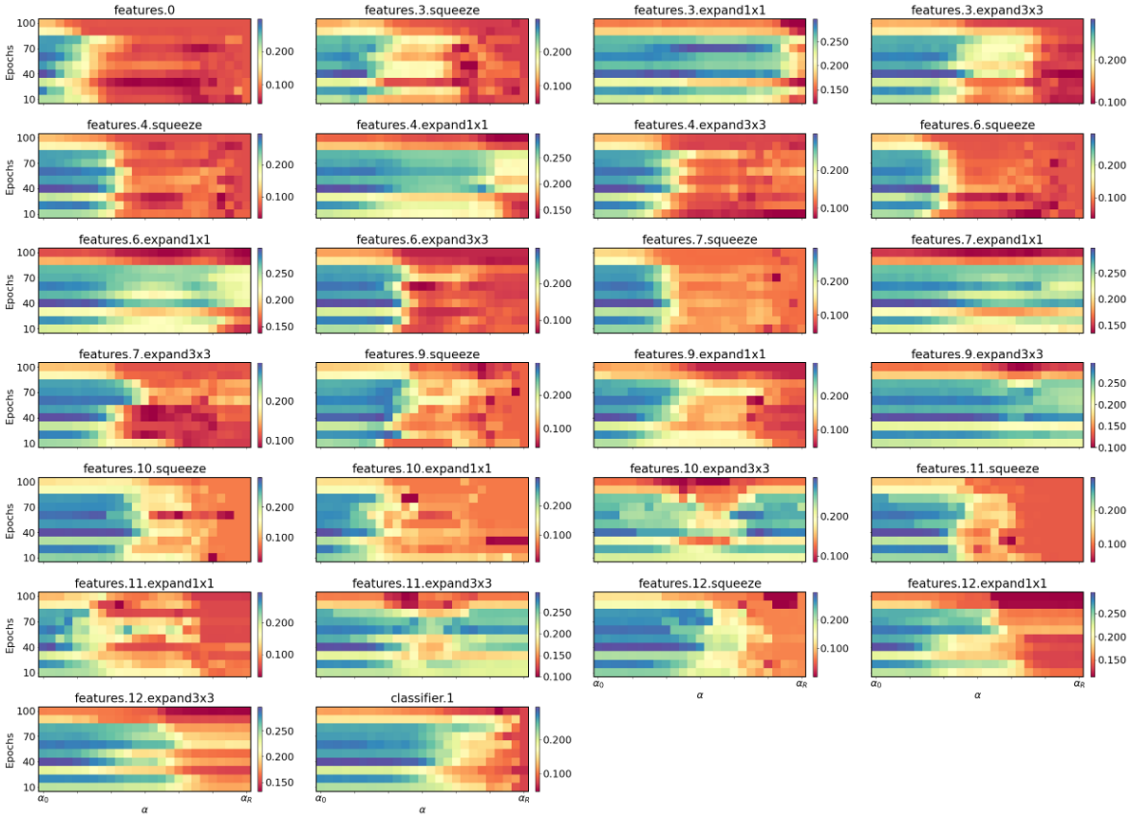


Fig. A.74 SqueezeNet-v1.1 response to adversarial MNIST dataset, for synaptic filter h_1.

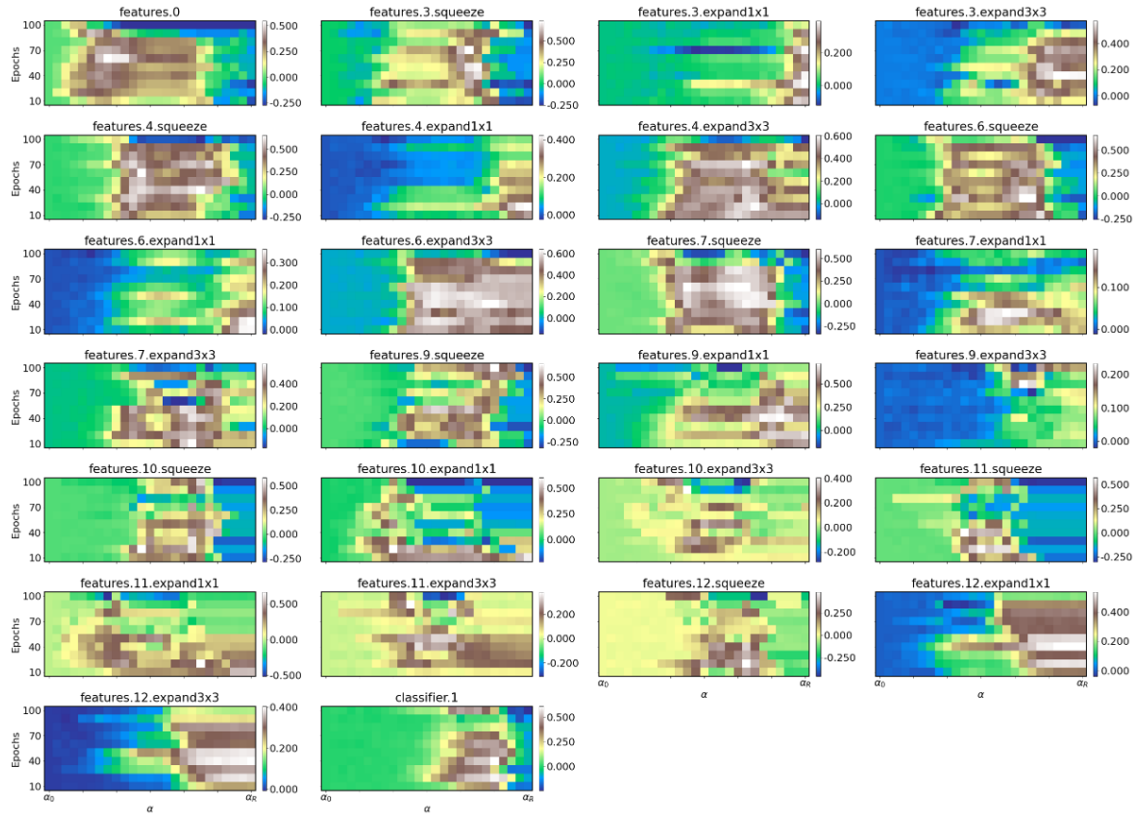


Fig. A.75 Difference in SqueezeNet-v1.1 responses to Clean and adversarial MNIST datasets, for synaptic filter h_1.

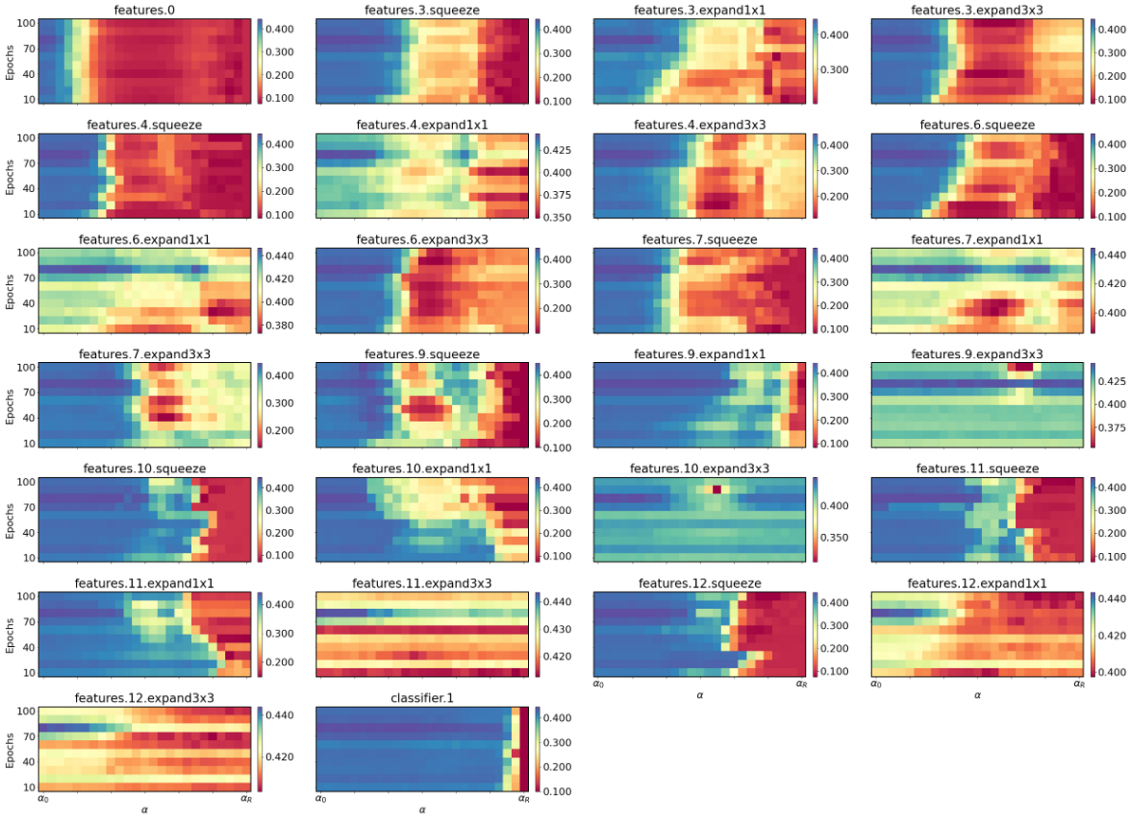


Fig. A.76 SqueezeNet-v1.1 response to clean CIFAR10 dataset, for synaptic filter h_1.

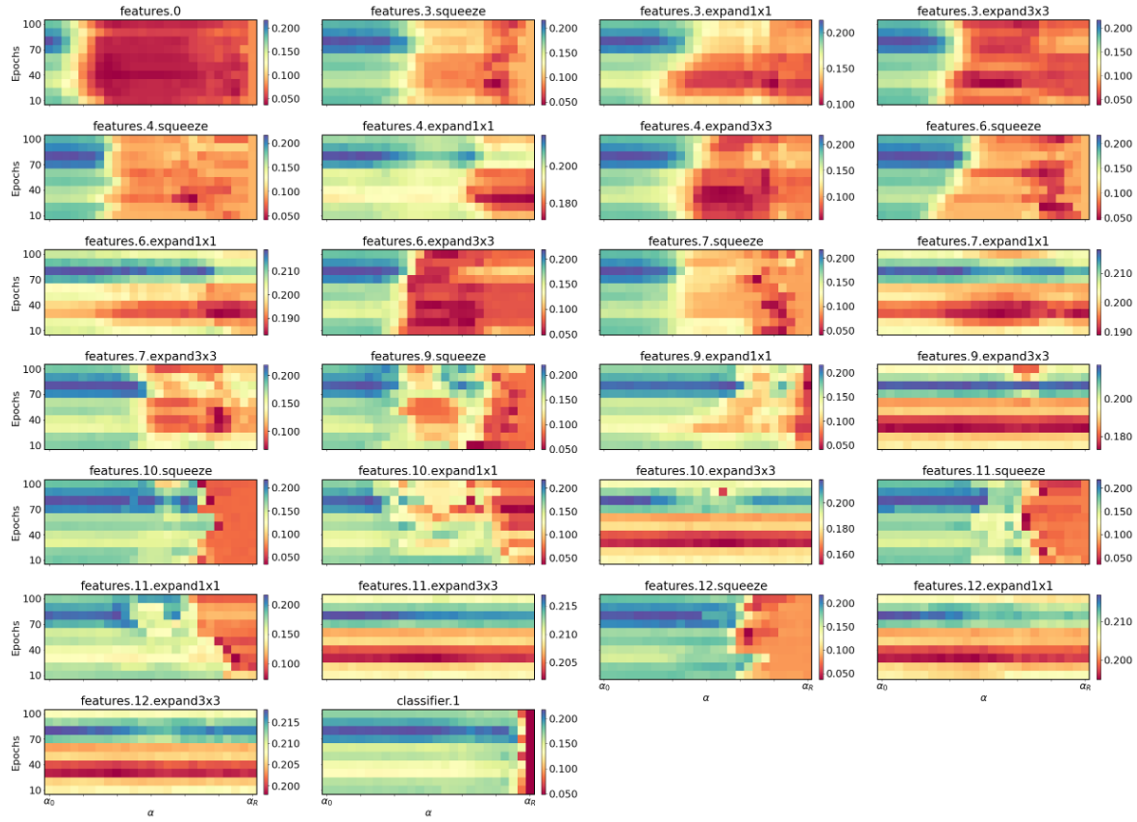


Fig. A.77 SqueezeNet-v1.1 response to adversarial CIFAR10 dataset, for synaptic filter h_1.

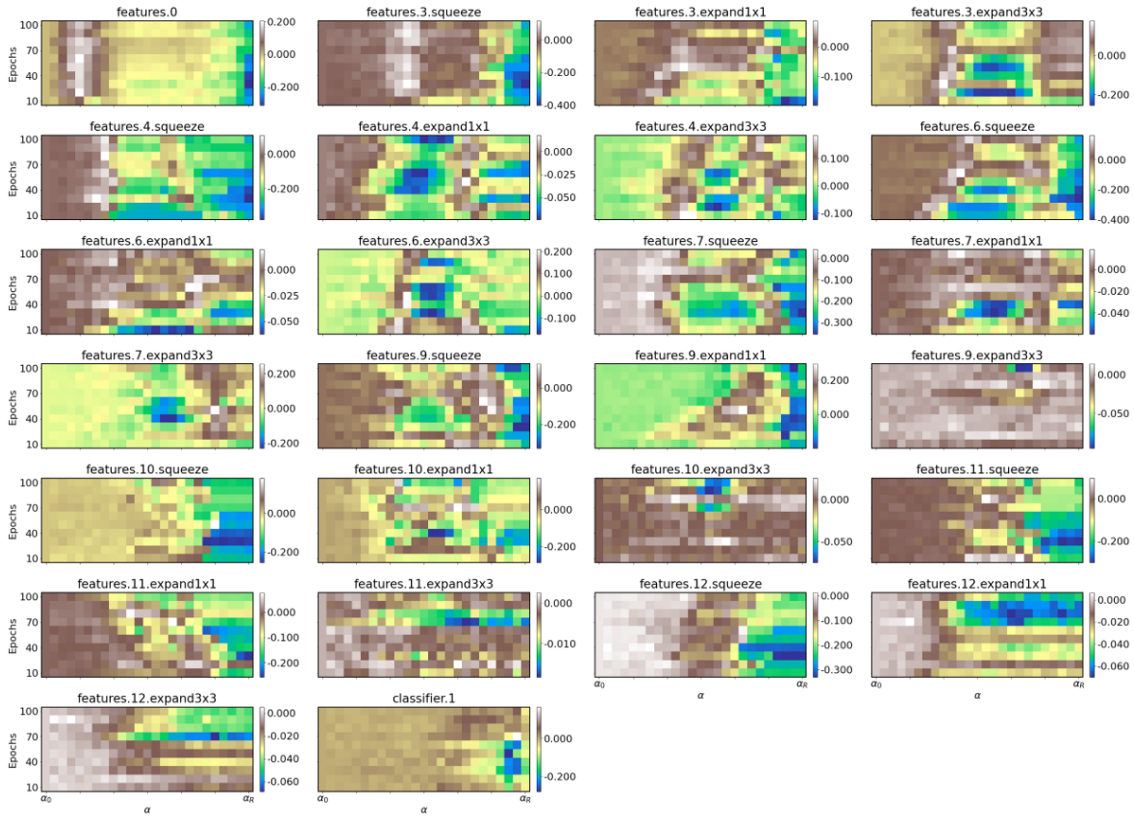


Fig. A.78 Difference in SqueezeNet-v1.1 responses to Clean and adversarial CIFAR10 datasets, for synaptic filter h_1.

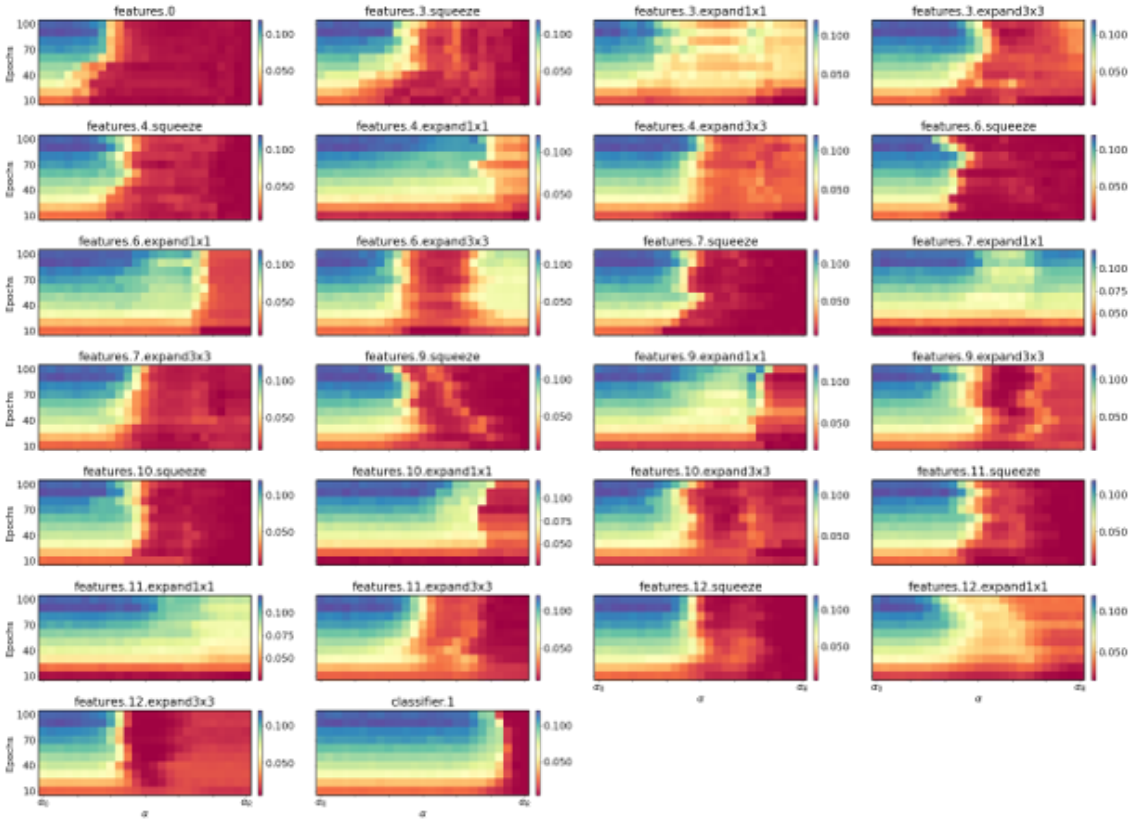


Fig. A.79 SqueezeNet-v1.1 response to clean ImageNet Tiny dataset, for synaptic filter h_1.

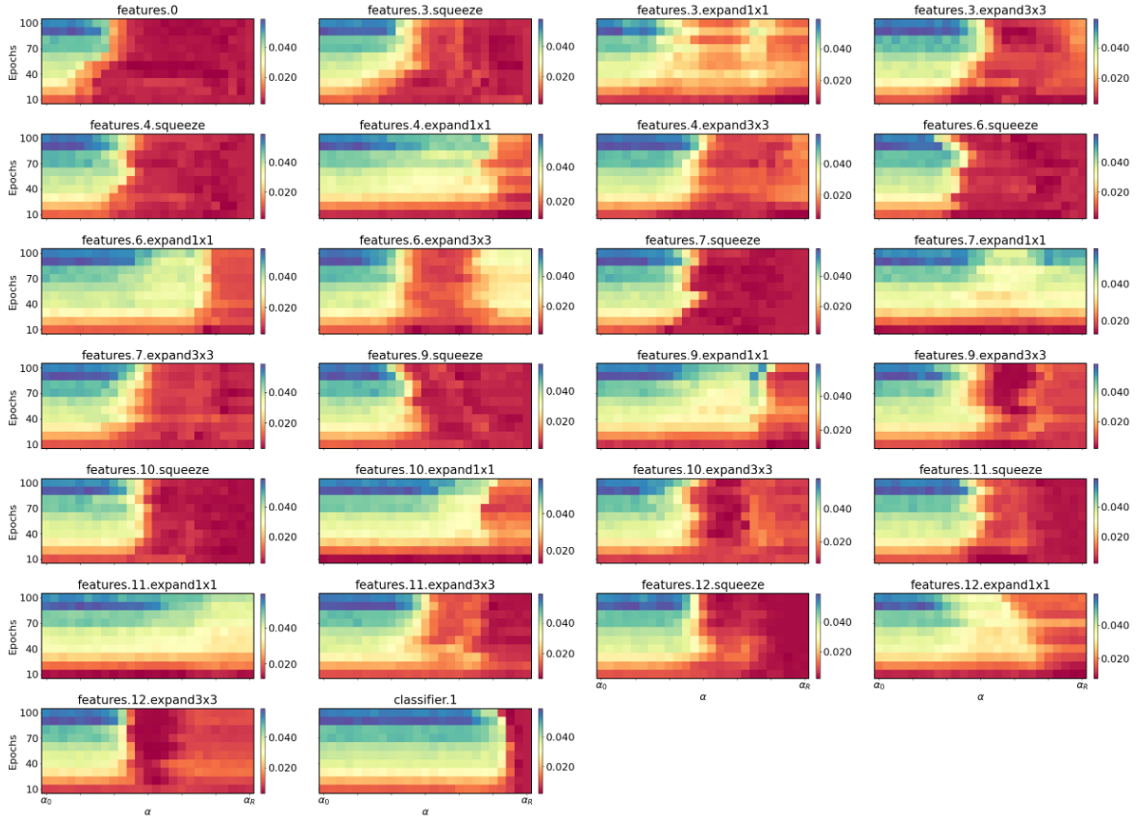


Fig. A.80 SqueezeNet-v1.1 response to adversarial ImageNet Tiny dataset, for synaptic filter h_1.

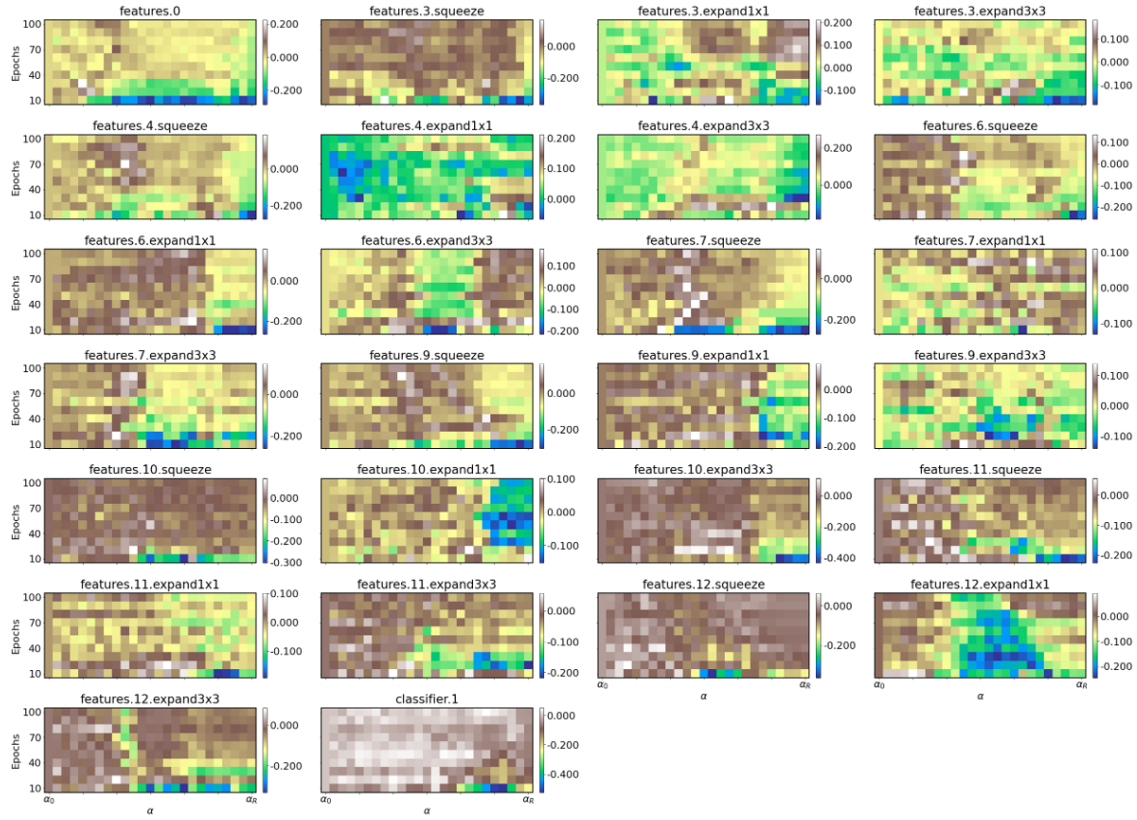


Fig. A.81 Difference in SqueezeNet-v1.1 responses to Clean and adversarial ImageNet Tiny datasets, for synaptic filter h_1.

A.4 ShuffleNet V2x1.0

A.4.1 Clean Dataset Responses

The ShuffleNet V2x1.0 network test accuracy responses to the ideal high-pass synaptic filter (see 3) on the clean, unperturbed, datasets is given for all layers of the network. We present the SqueezeNet-v1.1 responses to the synaptic filter h_1, for the MNIST dataset in Fig. A.82. The Pixel intensities in the presented figures show the measured network accuracy for all threshold values α ranging from α_0 to α_A, as per Sec. 3, measured every 10 epochs from 10 to 100 epochs during network training. Similarly, the network test accuracy responses for ShuffleNet V2x1.0 on the CIFAR10 dataset, are presented in Fig. A.85. The ShuffleNet V2x1.0 network test accuracy responses, for the ImageNet tiny dataset are presented in Fig. A.88. Using the results presented in Figs. A.82, Fig. A.82, and Fig. A.88 we find layers, such as layers '*stage3.branch1.0*', '*stage4.0.branch1.0*' and '*fc*' for example, where there are invariant response characteristics to different synaptic filters for the three evaluated datasets on the ShuffleNet V2x1.0 network.

Adversarial Dataset Responses

The ShuffleNet V2x1.0 network test accuracy responses to the different synaptic filters (see Sec. 3) on the adversarial datasets is given for all layers of the network. We present the ShuffleNet V2x1.0 responses for the synaptic filter h_1 for the adversarially perturbed MNIST dataset in Fig. A.83. The constraints for the perturbation magnitude of the attack is given in Sec. 3. The Pixel intensities in the presented figures show the measured network accuracy for all threshold values α ranging from α_0 to α_A, as per Sec. 3, measured every 10 epochs from 10 to 100 epochs during network training. Similarly, the network test accuracy responses for ShuffleNet V2x1.0 on the adversarially perturbed CIFAR10 dataset, are presented in Fig. A.86. The ShuffleNet V2x1.0 network test accuracy responses, for the adversarially perturbed ImageNet Tiny dataset are presented in Fig. A.89. Using the results presented in Fig. A.83, Fig. A.86, and Fig. A.89 we find layers, similar to those highlighted for the clean dataset responses, where there are invariant response characteristics to different synaptic filters for the three evaluated adversarial datasets on the ShuffleNet V2x1.0 network. Further analysis of these results is presented in Sec. 5.

Scaled Response Difference

The ShuffleNet V2x1.0 network scaled test accuracy responses difference (see Sec. 3.2) between the clean and adversarial datasets, to the different synaptic filters is given for all layers of the network. We present the ShuffleNet V2x1.0 response differences for the synaptic filter h_1 for the MNIST dataset in Figs. A.84. The constraints for the perturbation magnitude of the attack is given in Sec. 3. The Pixel intensities in the presented figures show the measured network accuracy for all threshold values α ranging from α_0 to α_A, as per Sec. 3, measured every 10 epochs from 10 to 100 epochs during network training. The network test accuracy response differences, for ShuffleNet V2x1.0 on the CIFAR10 dataset are presented in Fig. A.87. The ShuffleNet V2x1.0 network test accuracy response differences for ImageNet Tiny dataset are presented in Fig. A.90. Using the results presented in Fig. A.84, Fig. A.87, and Fig. A.90 we find links and layers where the adversary is targeting the ShuffleNet V2x1.0 network. We also find layer response differences that show different characteristics for different datasets, further analysis of the presented results is given in Sec. 5.

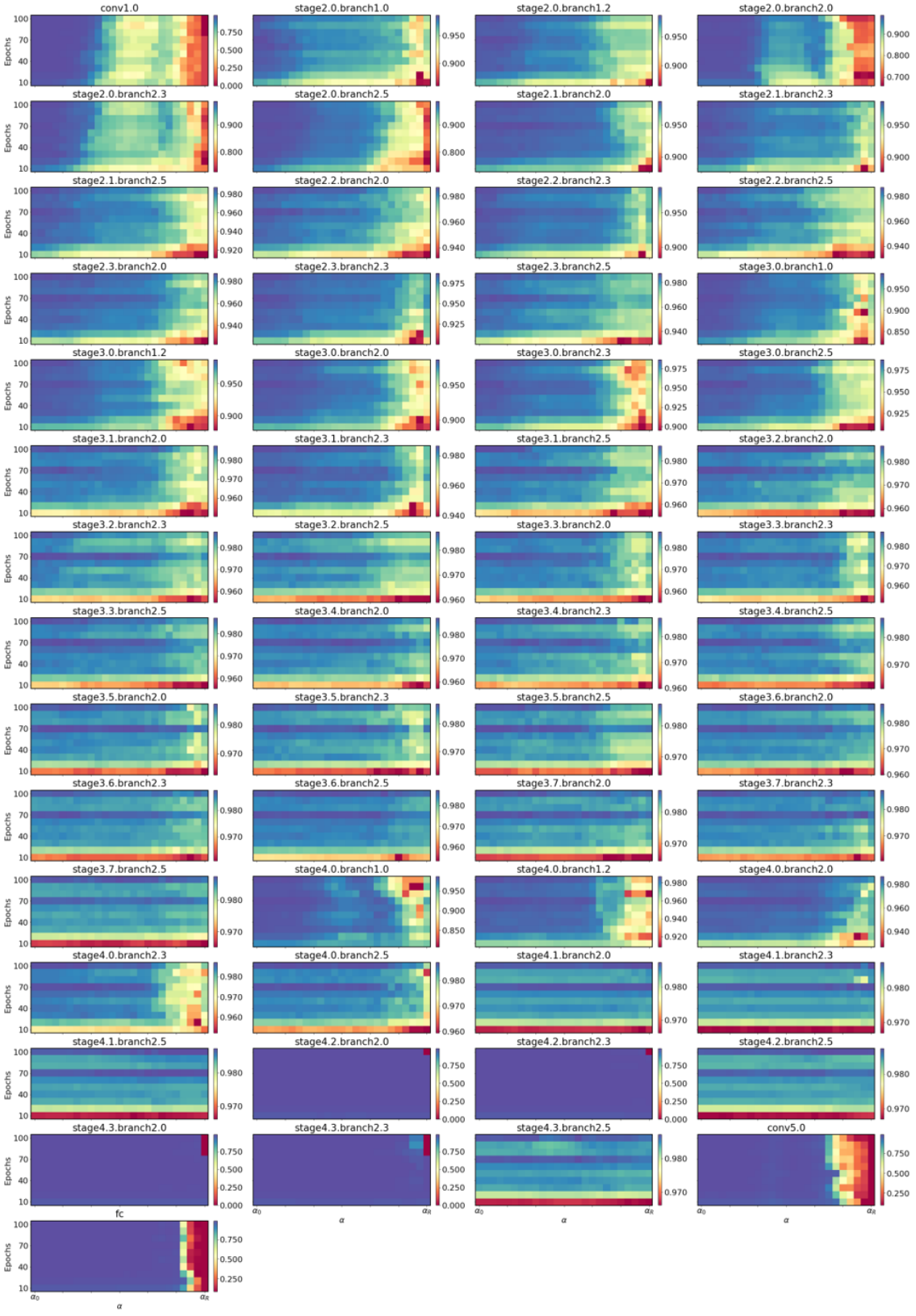


Fig. A.82 ShuffleNet V2x1.0 response to clean MNIST dataset, for synaptic filter h_1.

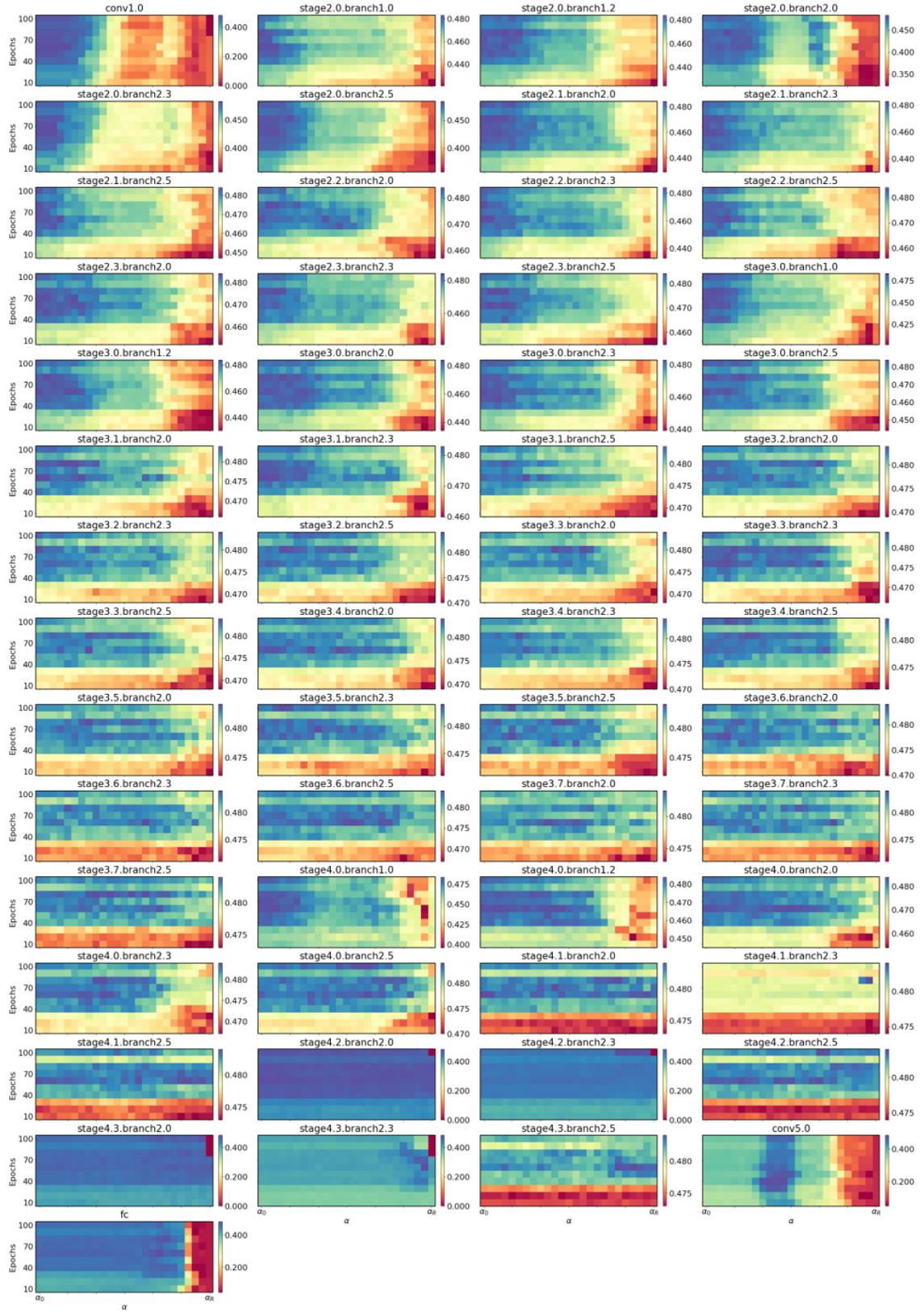


Fig. A.83 ShuffleNet V2x1.0 response to adversarial MNIST dataset, for synaptic filter h_1.

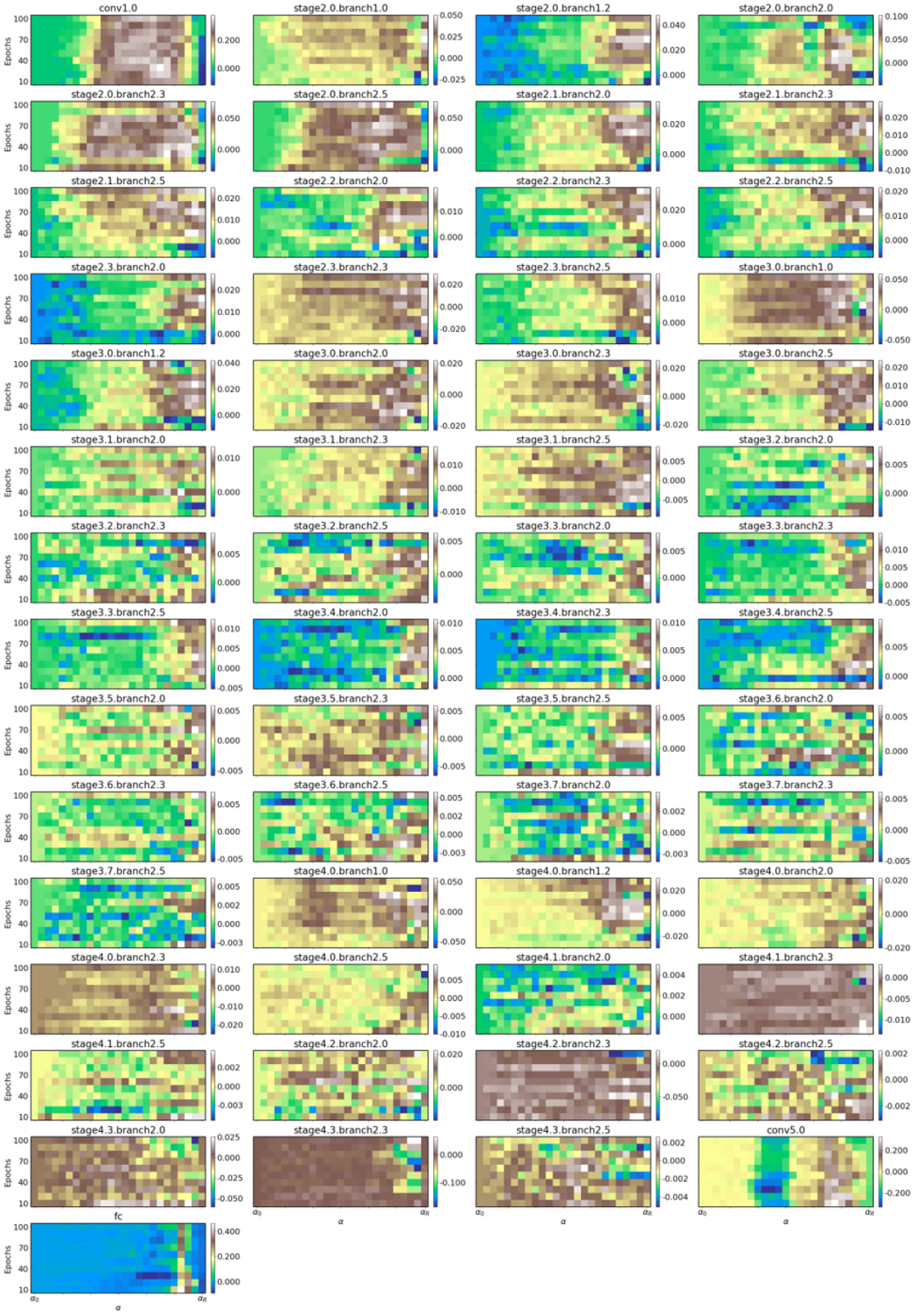


Fig. A.84 Difference in ShuffleNet V2x1.0 responses to Clean and adversarial MNIST datasets, for synaptic filter h_1.

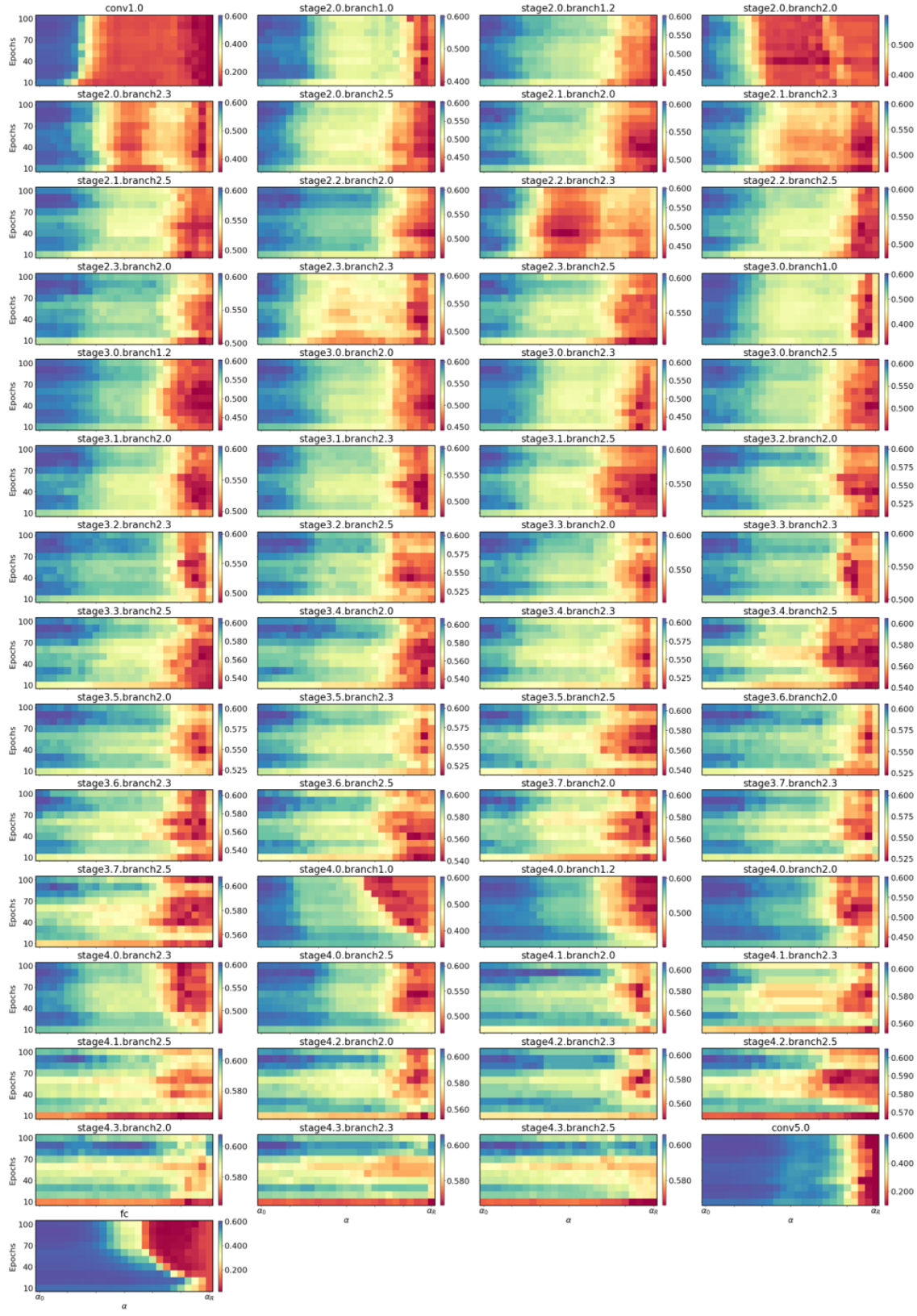


Fig. A.85 ShuffleNet V2x1.0 response to clean CIFAR10 dataset, for synaptic filter h_1.

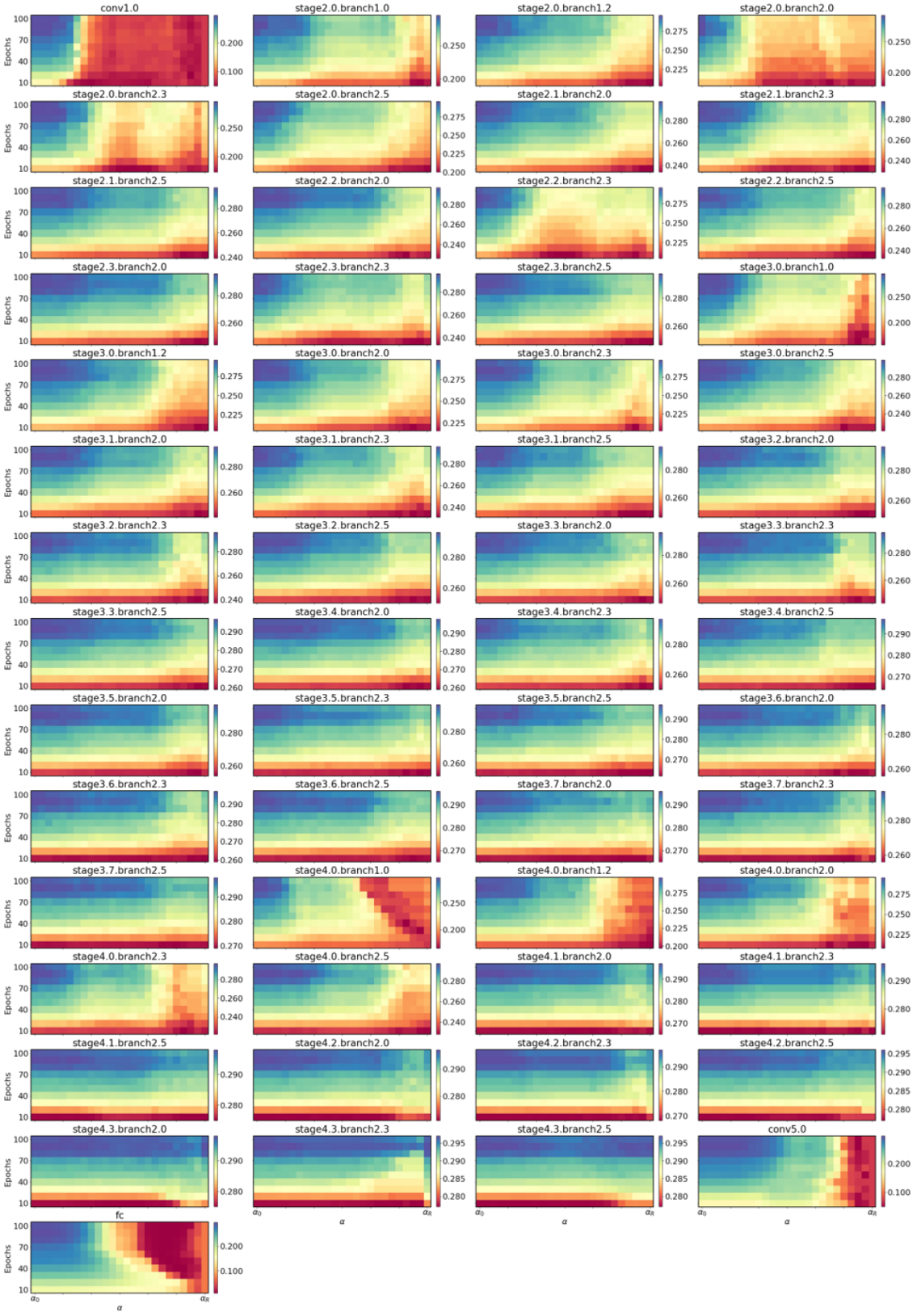


Fig. A.86 ShuffleNet V2x1.0 response to adversarial CIFAR10 dataset, for synaptic filter h_1.

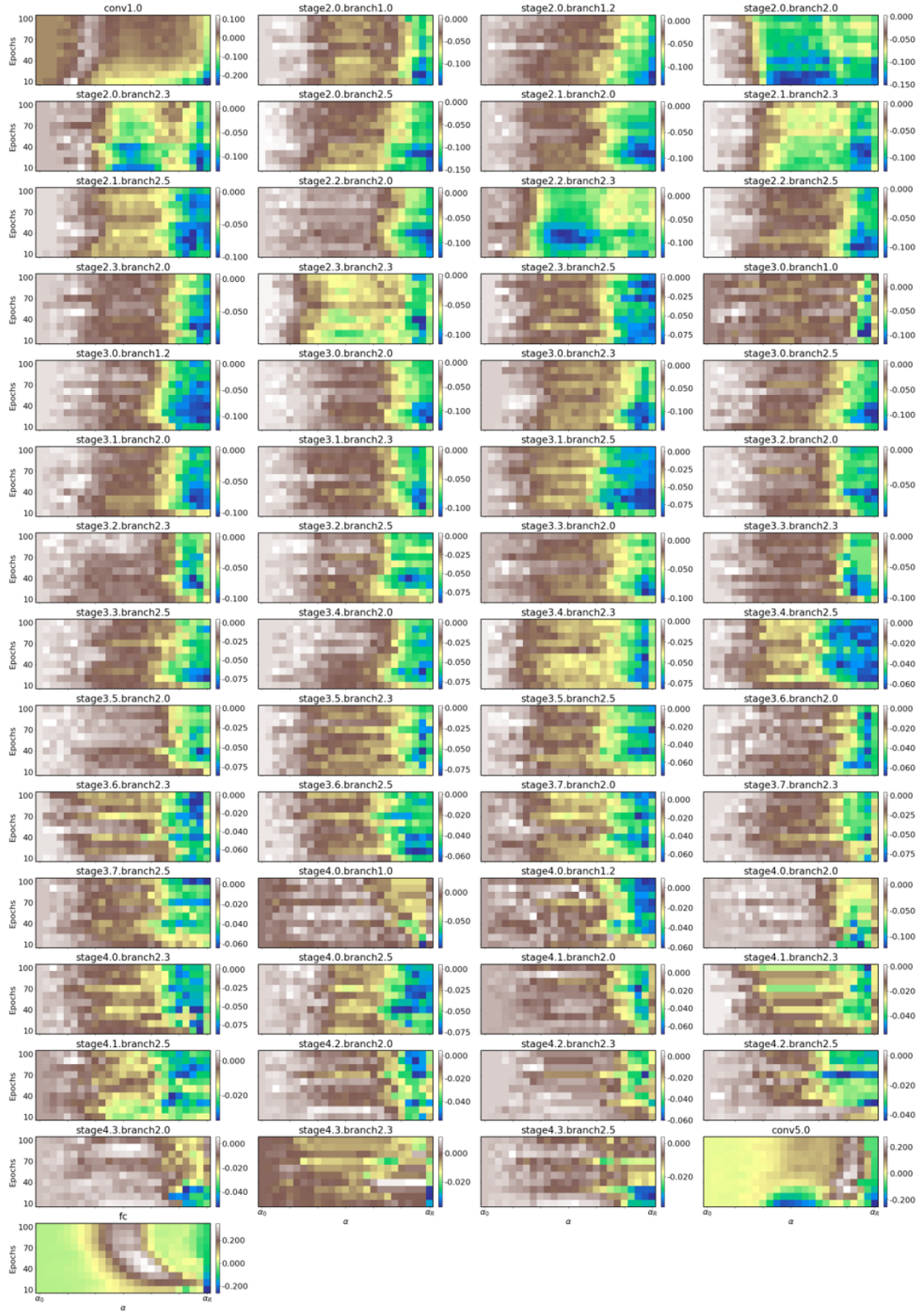


Fig. A.87 Difference in ShuffleNet V2x1.0 responses to Clean and adversarial CIFAR10 datasets, for synaptic filter h_1.

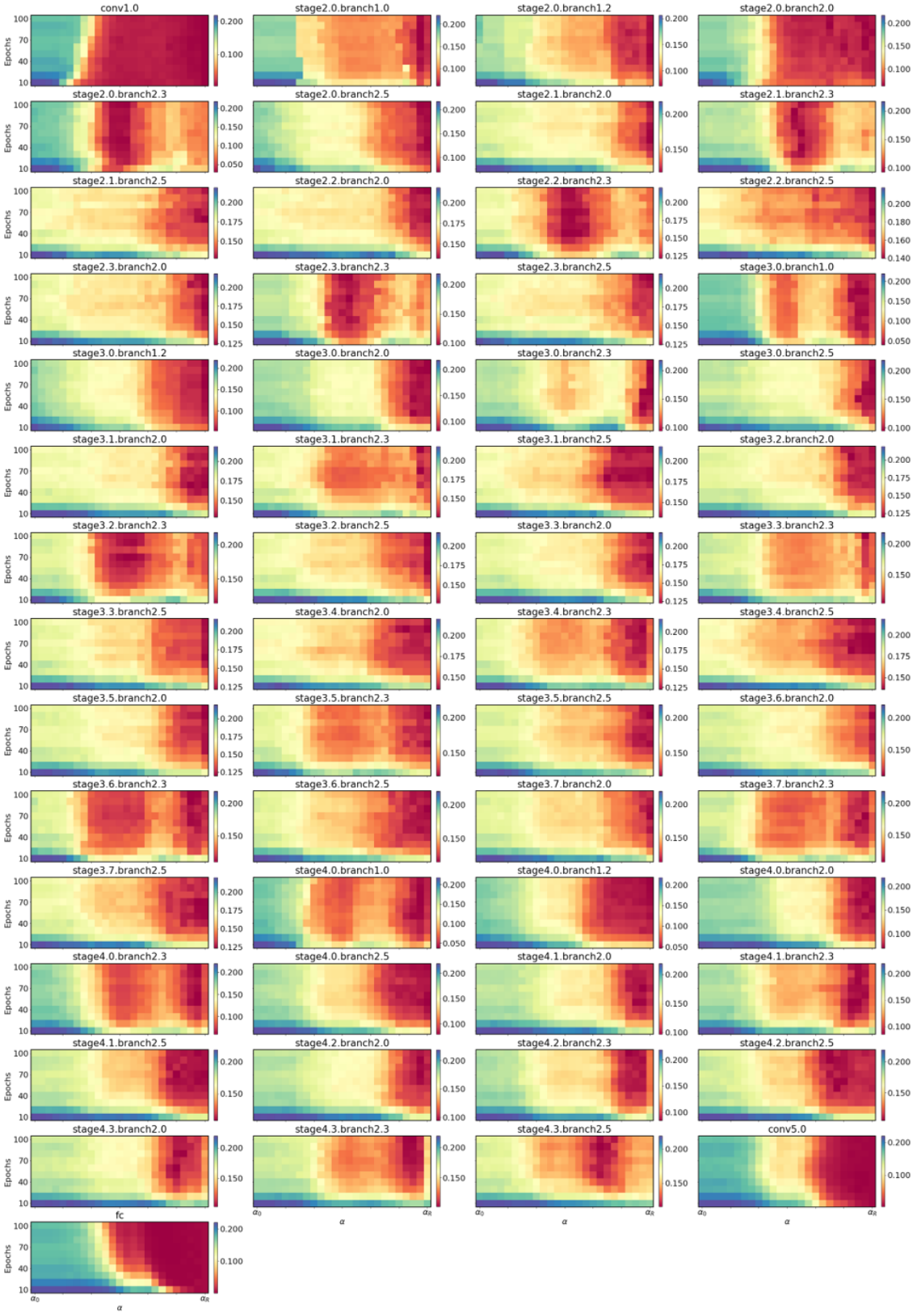


Fig. A.88 ShuffleNet V2x1.0 response to clean ImageNet Tiny dataset, for synaptic filter h_1.

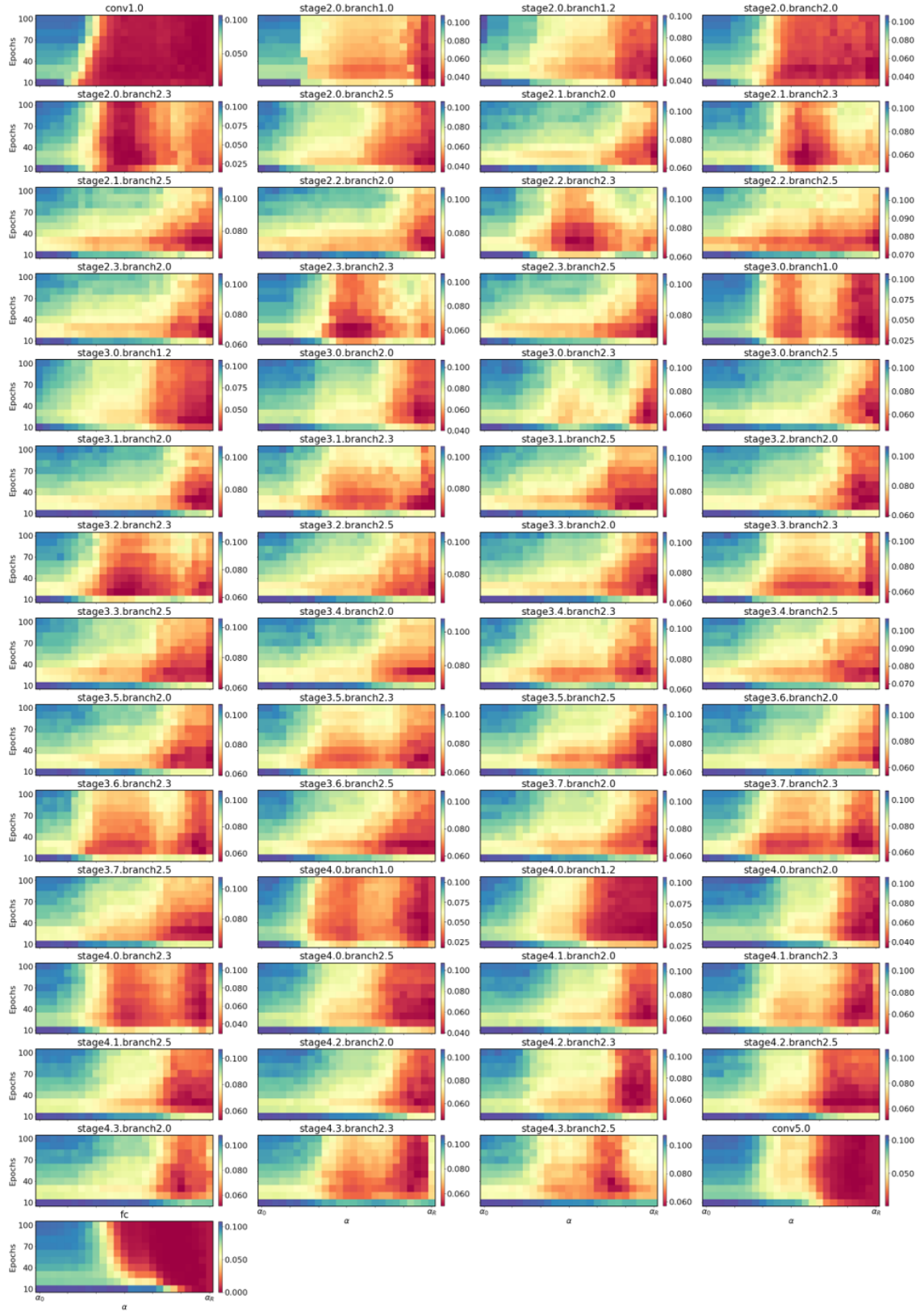


Fig. A.89 ShuffleNet V2x1.0 response to adversarial ImageNet Tiny dataset, for synaptic filter h_1.

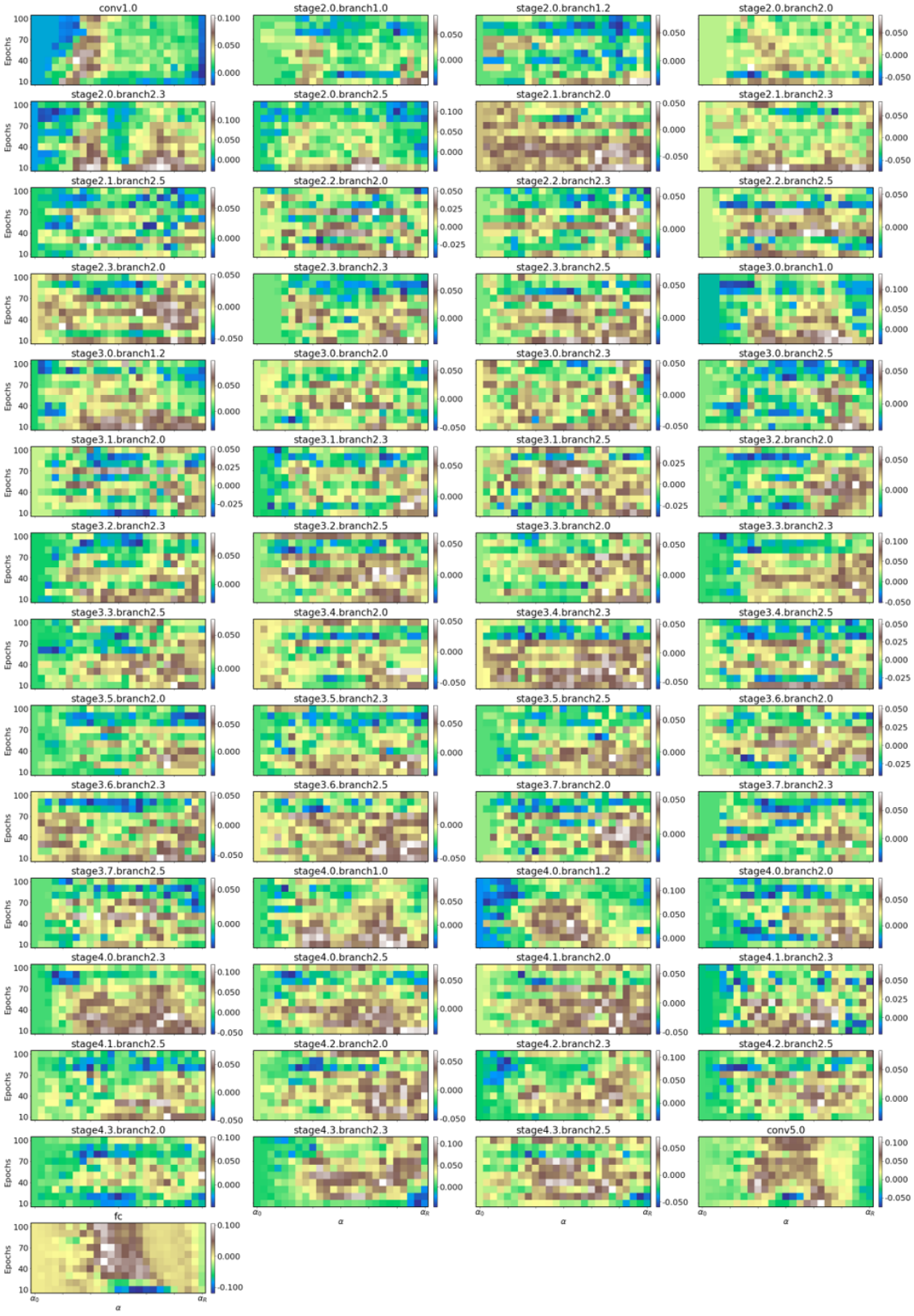


Fig. A.90 Difference in ShuffleNet V2x1.0 responses to clean and adversarial ImageNet datasets, for synaptic filter h_1.